Secure two-party quantum computation against specious adversaries

Frédéric Dupuis
ETH Zürich

Joint work with
Jesper Buus Nielsen (Aarhus Universitet)
Louis Salvail (Université de Montréal)

March 21, 2011

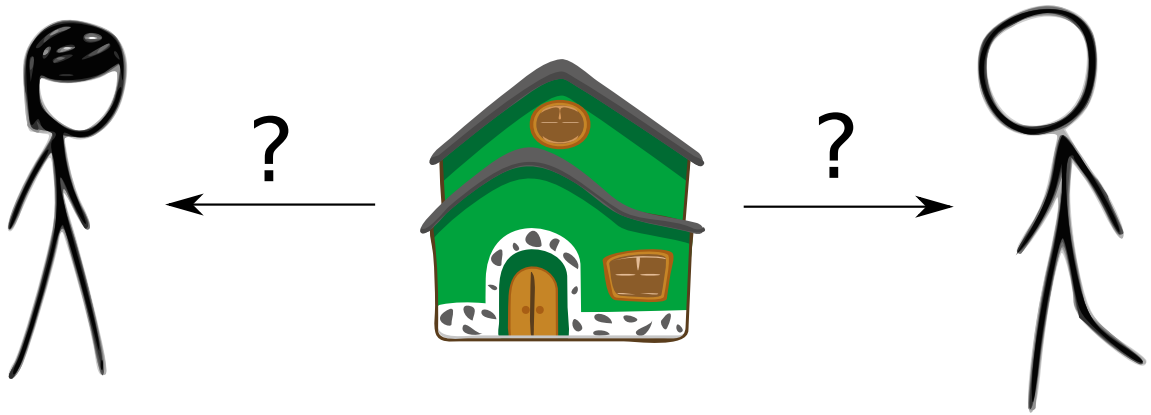
Outline

- Two-party classical computation
- Two-party quantum computation
- Specious adversaries
- Our protocol
- Conclusion and open problems

Introduction: Two-party classical computation

What is two-party computation? Consider the following problem:

- Alice and Bob are getting divorced, and need to figure out who gets the house.

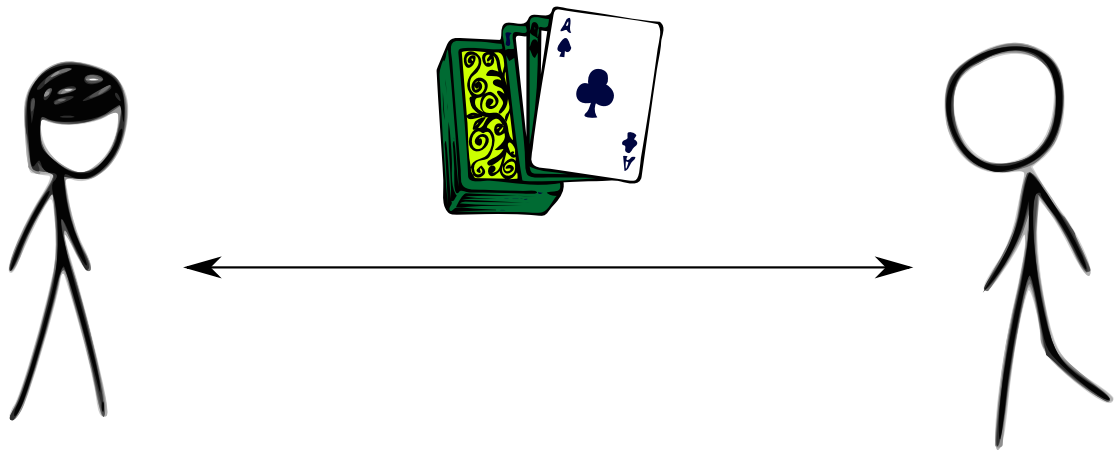


- They could flip a coin, but they live far away now.
- Can they do it fairly over the phone?

Two-party classical computation

A more complicated problem:

- Alice and Bob would like to play poker over the Internet.

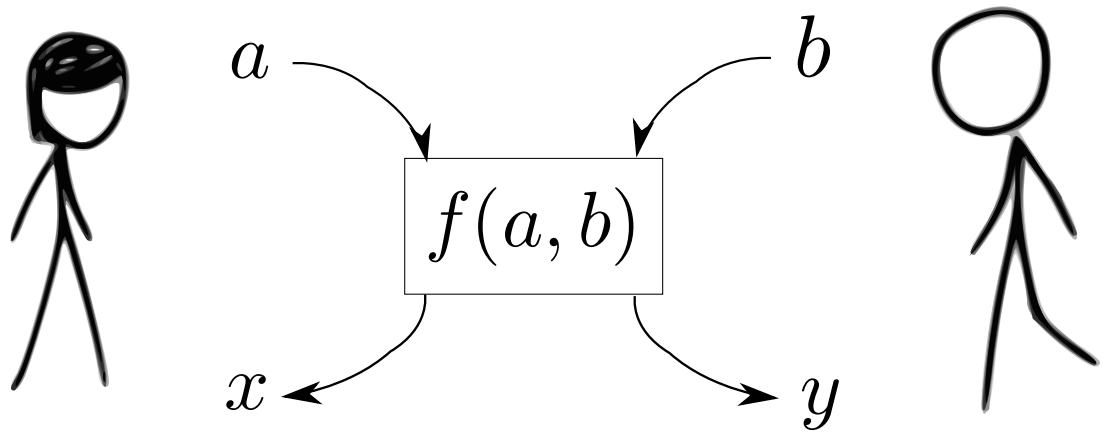


- BUT: Neither of them would really trust a server to deal fairly.

Two-party classical computation

How can we consider these kinds of problems abstractly?

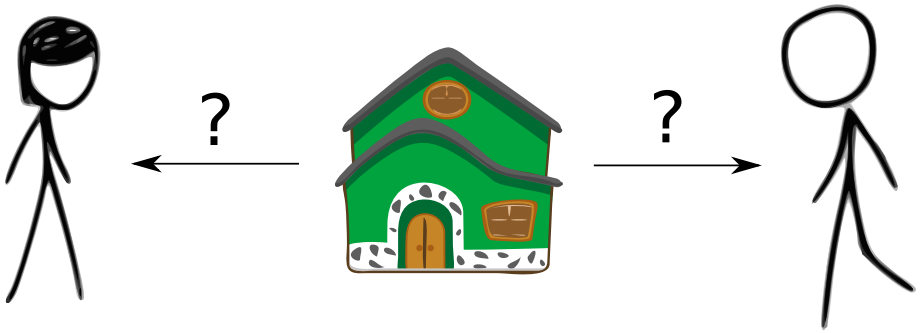
- Alice and Bob want to compute a function $f : (a, b) \mapsto (x, y)$.

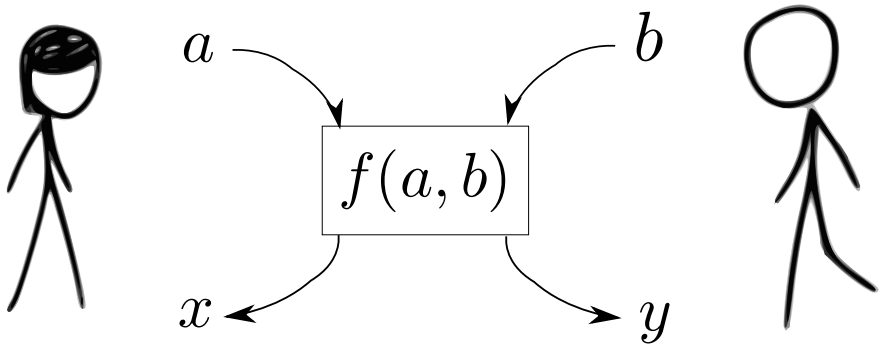


- Alice and Bob have a and b as inputs and x and y as outputs respectively.
- Alice must not get extra information about b and y (and vice-versa).
- They would like to do it only by communicating to each other.

Two-party classical computation

Example: Coin-flipping

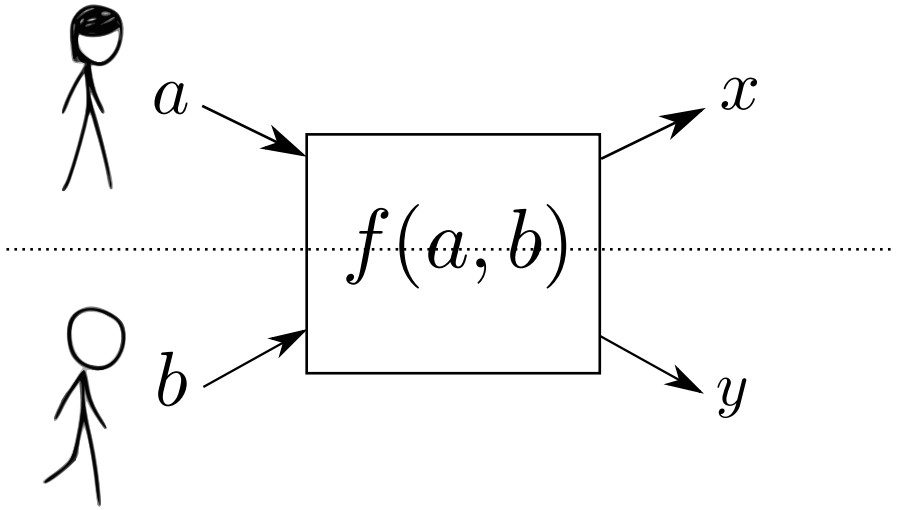




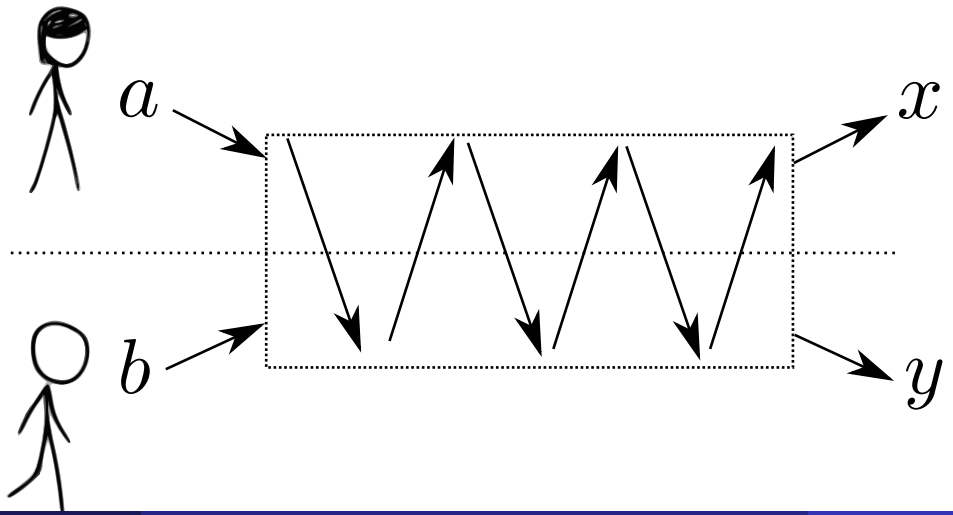
Let's choose $f(a,b) = x \oplus y$. Either Alice or Bob can force the outcome to be random by choosing a random input.

Two-party classical computation

Our goal is then to simulate this:



with this:



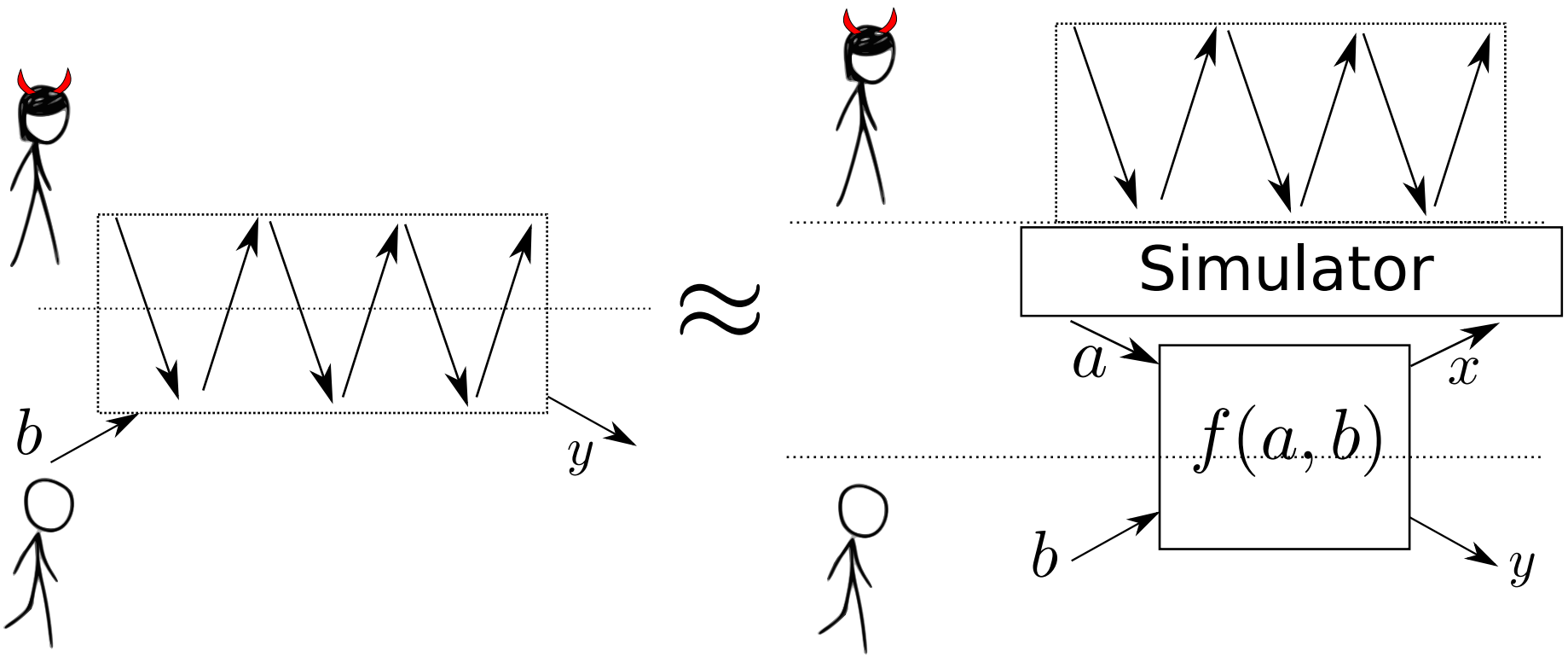
Two-party classical computation: Security

We need to do this “securely”. What does this mean?

- We want to make sure that Alice cannot learn anything about Bob’s input and output other than what she can infer from her output.
- We want to make sure that Alice cannot influence the outputs except through her choice of input.
- And vice-versa!

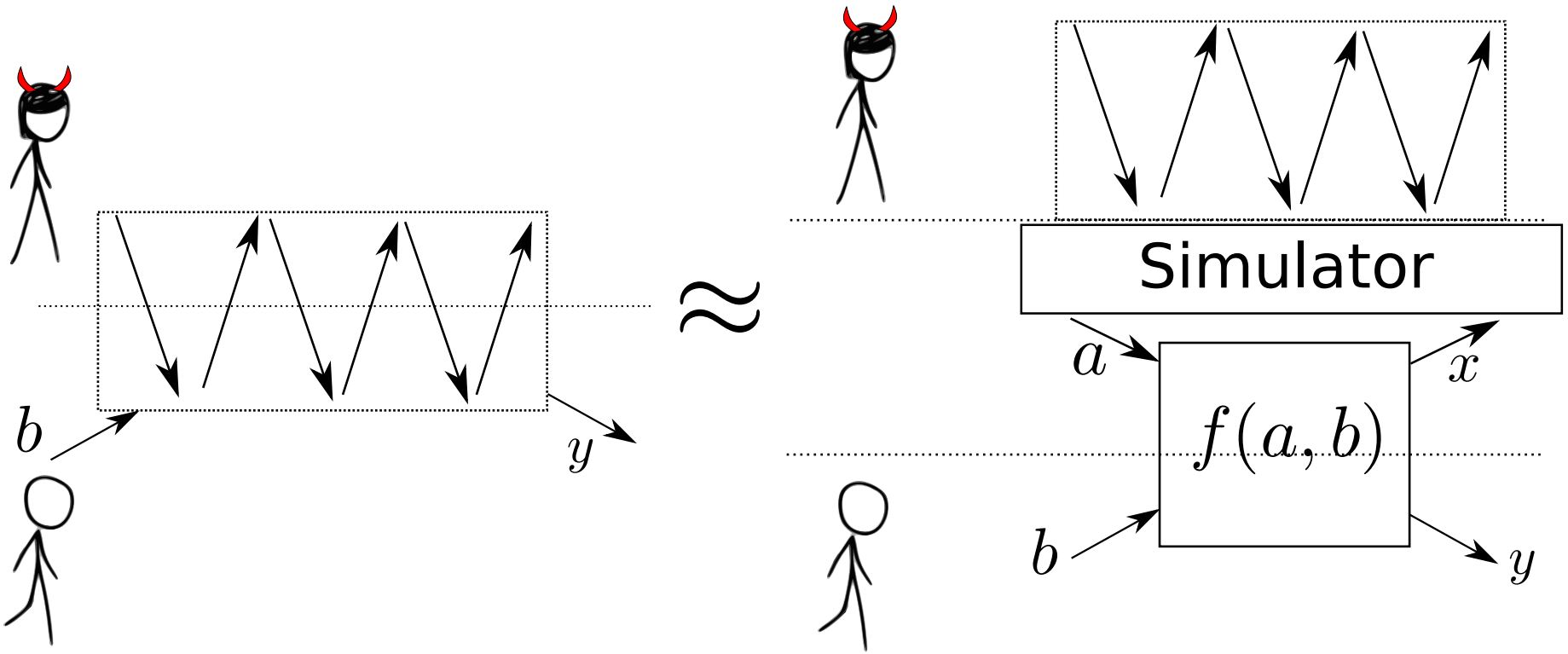
Two-party classical computation: Security

We will say that a protocol is secure if there exists a simulator:



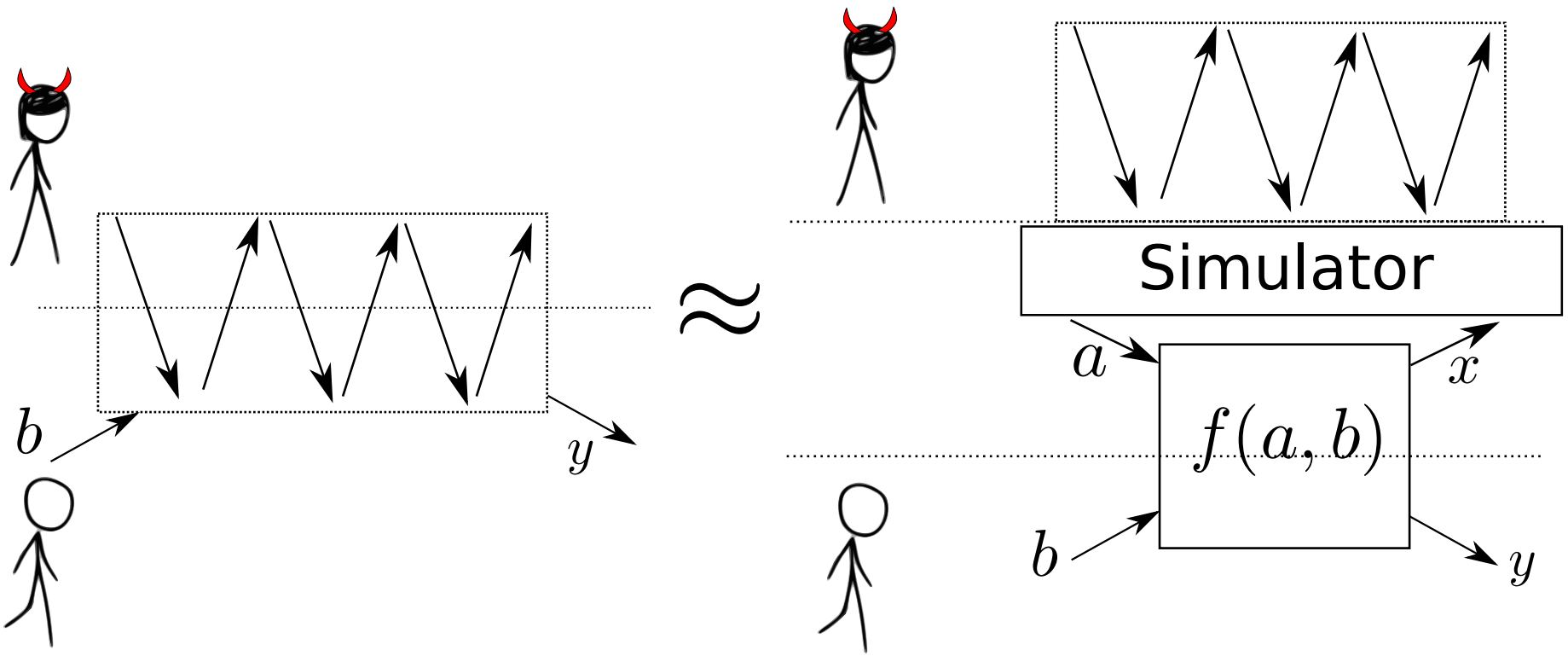
for any cheating Alice.

Two-party classical computation: Security



We require that all information collected by Cheating Alice, together with b and y be indistinguishable from what happens in the real protocol.

Two-party classical computation: Security



In other words, Cheating Alice might as well have attacked the ideal f-box directly.

Two-party classical computation: Security

Besides intuition, why do we define security this way?

- Composition: If a protocol is secure according to this definition, then if we embed it into a bigger protocol, the bigger protocol is also secure.
- Example: If we have a protocol to shuffle cards securely, we want to be able to use it in a poker program.

Two-party classical computation: Implementation?

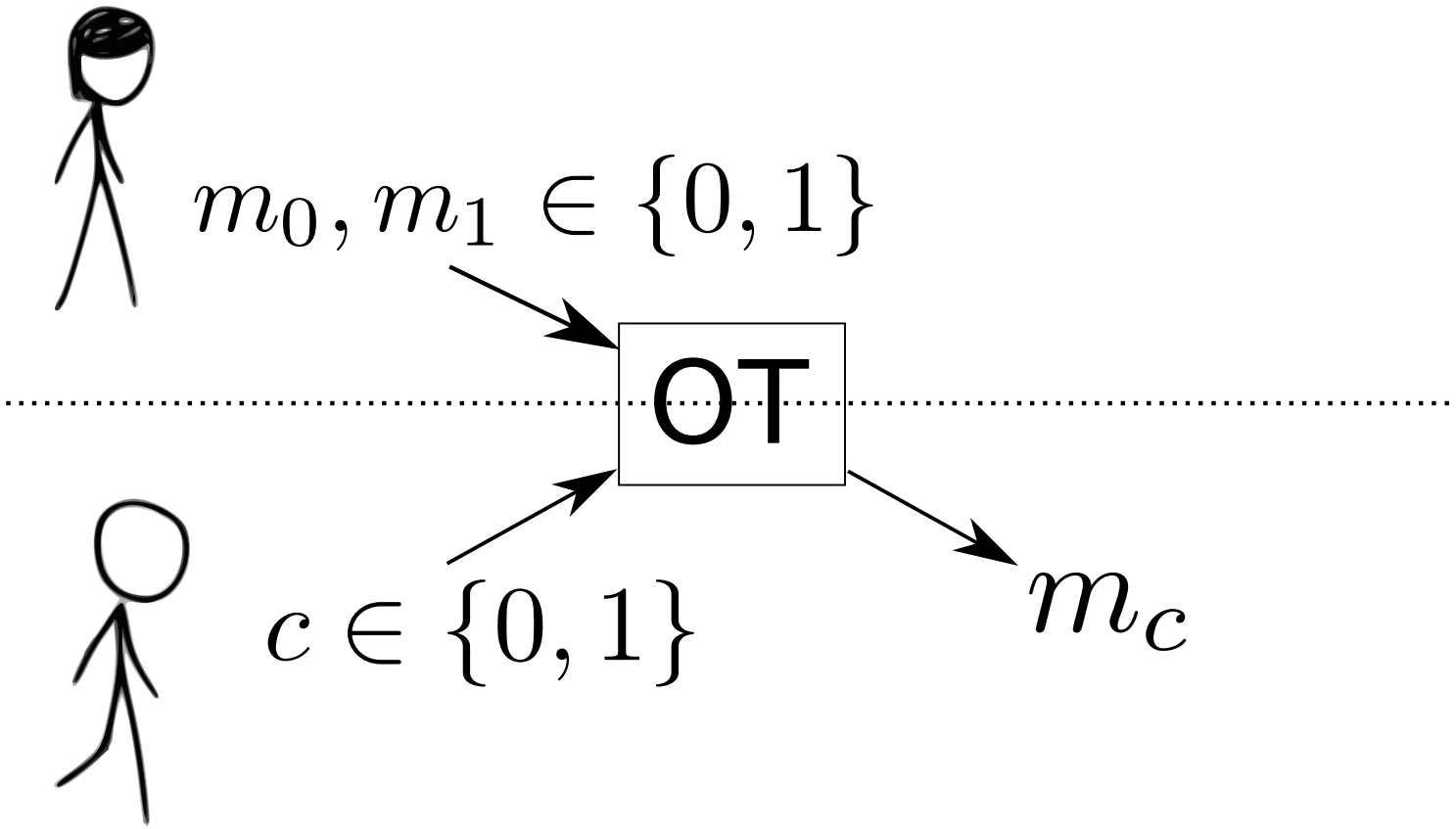
So, given a function f, how do we get a secure protocol for it?

- Bad news: It's impossible to do it perfectly without assumptions.
- Possible assumptions: Computational difficulty, noisy channels whose noise is not controlled by either party
- How do we use the assumption to get a protocol? We "package" the assumption into a *cryptographic primitive*:
  - A simple, fixed f that we can use on top of communication.

Two-party classical computation: Implementation?

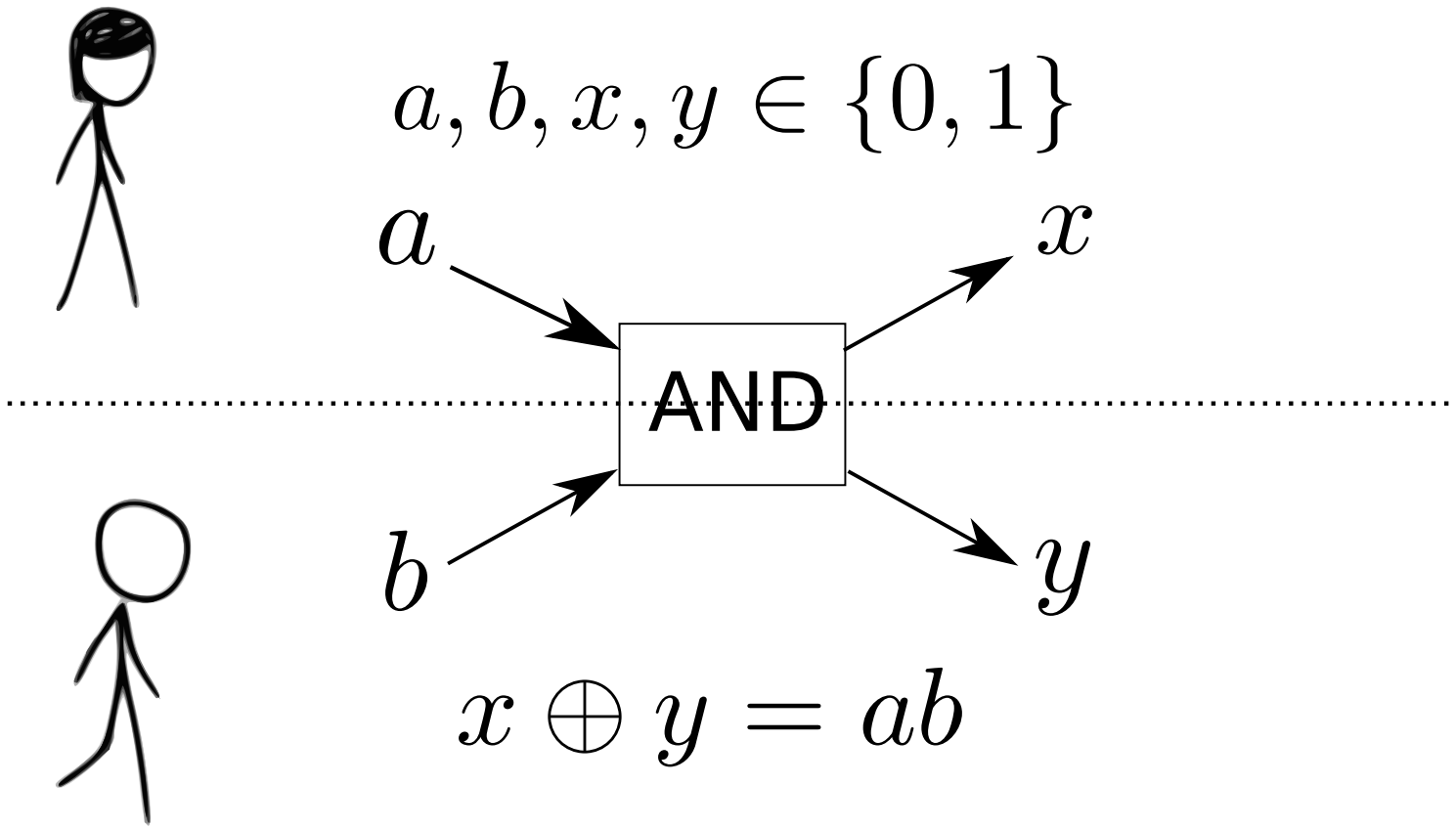
So what primitive can we use?

- One-out-of-two Oblivious Transfer (OT):



Two-party classical computation: Implementation?

Another primitive equivalent to OT: the AND-box [Wolf-Wullschleger 06]:

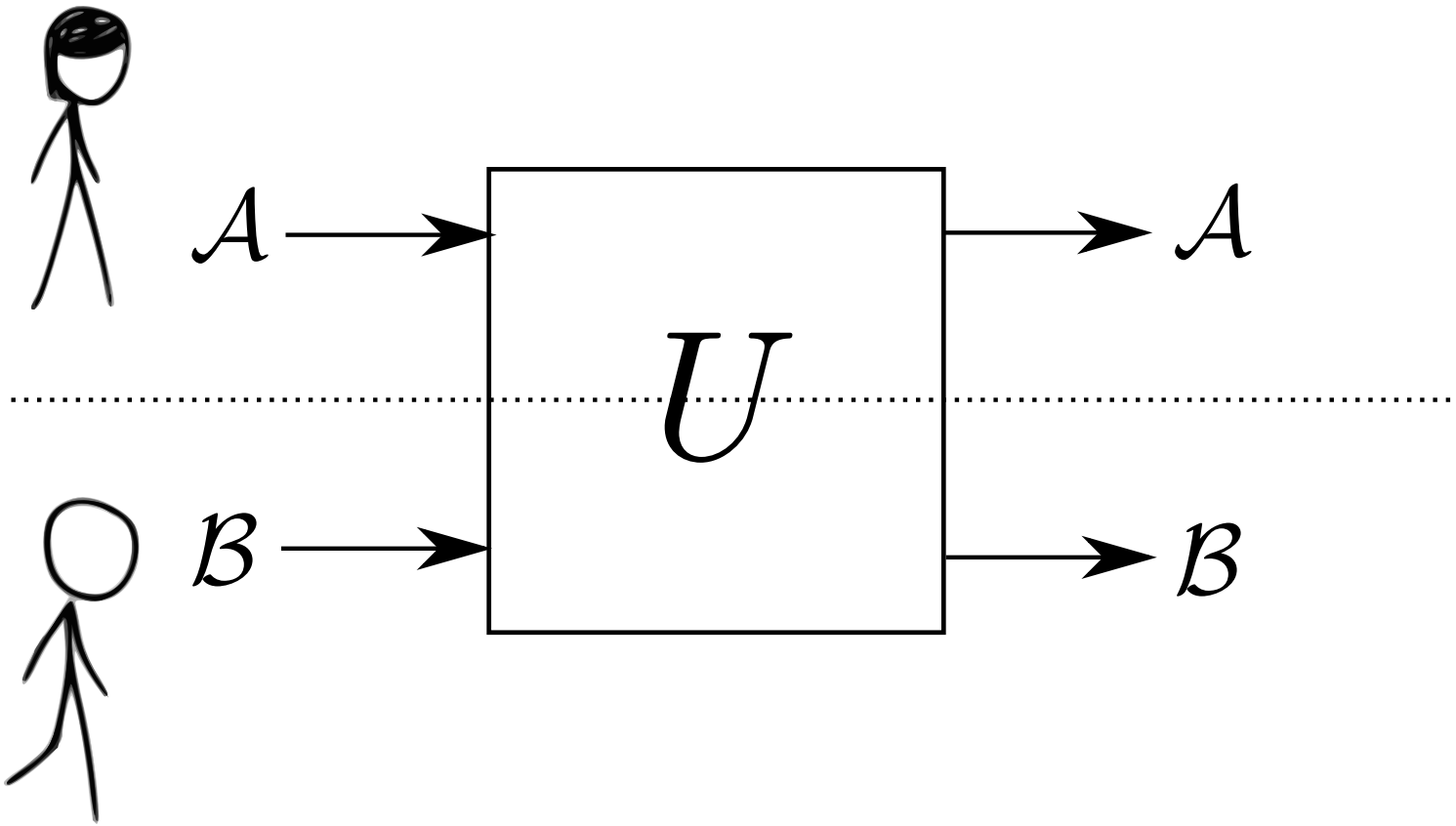


Two-party quantum computation

What about *quantum* circuits?

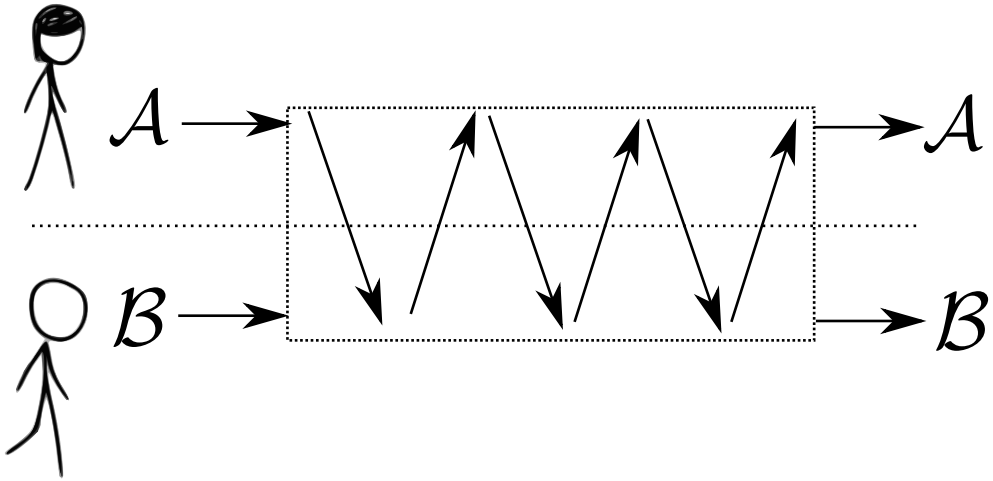
Two-party quantum computation

- Alice and Bob want to securely execute a quantum circuit U on a joint input space $\mathcal{A} \otimes \mathcal{B}$.

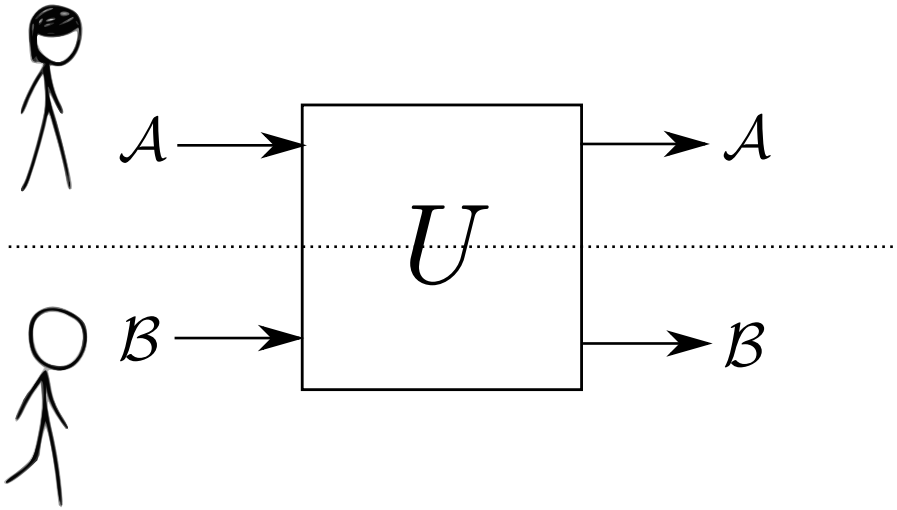


Two-party quantum computation

So they want a protocol



that imitates the ideal box:



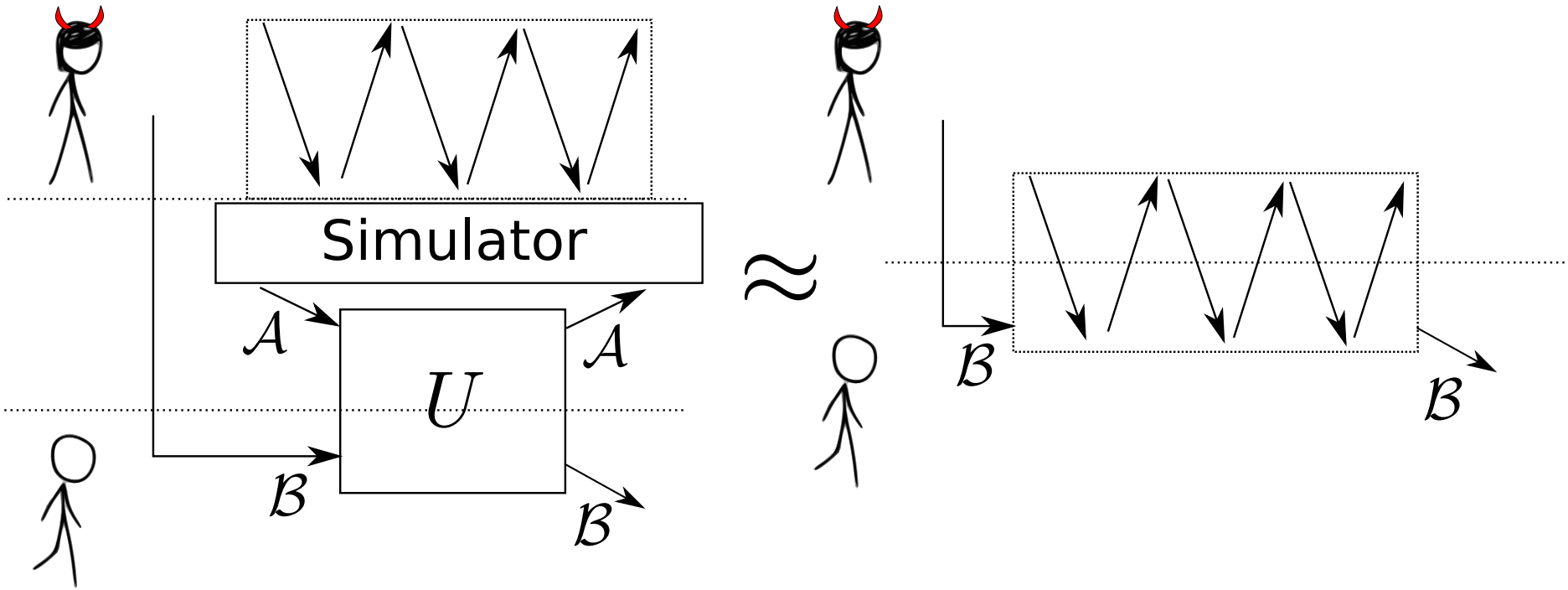
Two-party quantum computation

Simple example: Alice and Bob want to swap two qubits ($U = \mathrm{SWAP}$).

- Can also be used for coin-tossing: Alice and Bob input a random bit, swap them, then XOR the result.

Two-party quantum computation: Security

We can define the security of a quantum protocol in the same way as for the classical case: we need a simulator



for any cheating Alice and any ρ (and likewise for cheating Bobs).

Two-party quantum computation: Implementation?

Can we implement this?

- First surprise: This is *not* a generalization of the classical case: there is no U that can implement OT securely!
- This might actually be *easier* than classical two-party computation—or may be of incomparable difficulty.
- Is OT sufficient to implement this? Can we do it with something weaker than OT?

Our result

We show that:

- OT is sufficient for “specious” adversaries
- If U is in the Clifford group, a quantum SWAP (weaker than OT) is enough

Specious adversaries

Specious adversaries

What is a dishonest player allowed to do?

- Most general case: anything

Specious adversaries

What is a dishonest player allowed to do?

- Most general case: anything
- Semi-honest: Must follow the protocol exactly, but tries to extract any information from data he has.

Specious adversaries

What is a dishonest player allowed to do?

- Most general case: anything
- Semi-honest: Must follow the protocol exactly, but tries to extract any information from data he has.
  - Problematic for quantum protocols: We can't copy quantum information, so if the protocol specifies that we must send something, we can't keep a backup and look at it...

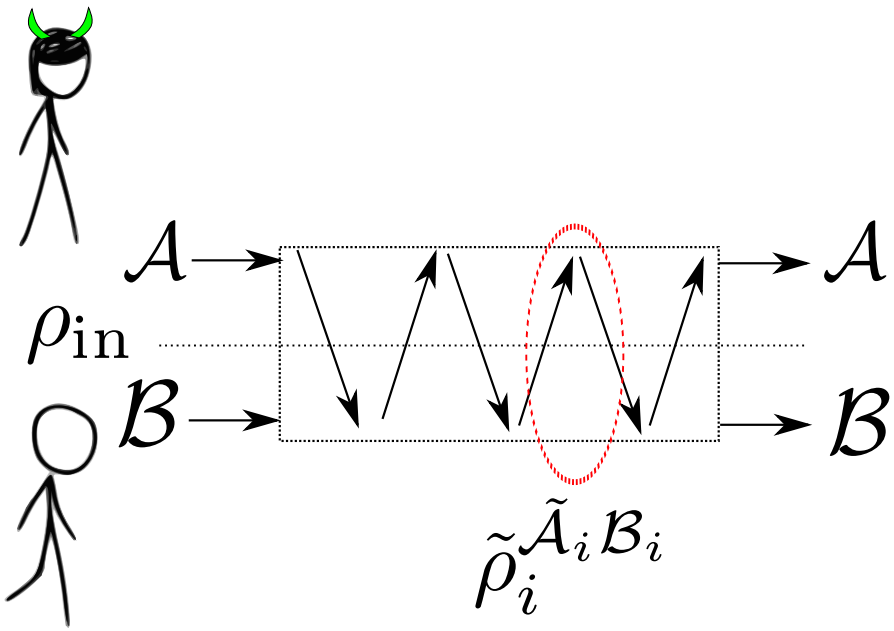
Specious adversaries

What is a dishonest player allowed to do?

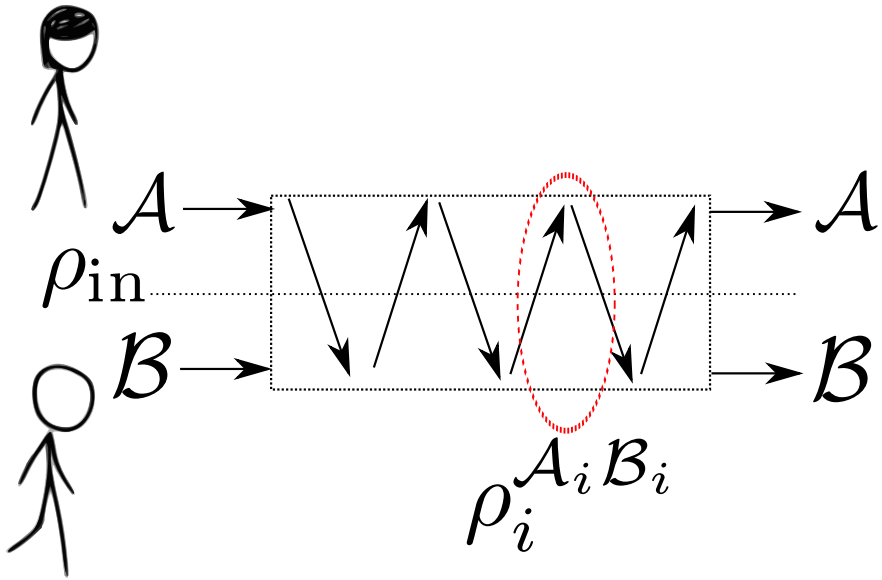
- Most general case: anything
- Semi-honest: Must follow the protocol exactly, but tries to extract any information from data he has.
  - Problematic for quantum protocols: We can't copy quantum information, so if the protocol specifies that we must send something, we can't keep a backup and look at it...
- *Specious*: At every step, the adversary can produce a state that is almost identical to what the honest player would have. Can be viewed as "quantum semi-honest".

Specious adversaries

Dishonest Alice



Honest Alice



- Dishonest Alice is specious if $\exists$ a quantum operation $\mathcal{T} : \mathcal{L}(\tilde{\mathcal{A}}_i) \to \mathcal{L}(\mathcal{A}_i)$ such that for all inputs ρ_{in}, $(\mathcal{T} \otimes \mathbb{1}_{\mathcal{B}_i})(\tilde{\rho}_i) \approx \rho_i^{\mathcal{A}_i\mathcal{B}_i}$.
- Can be viewed as the possibility of an audit at every step.

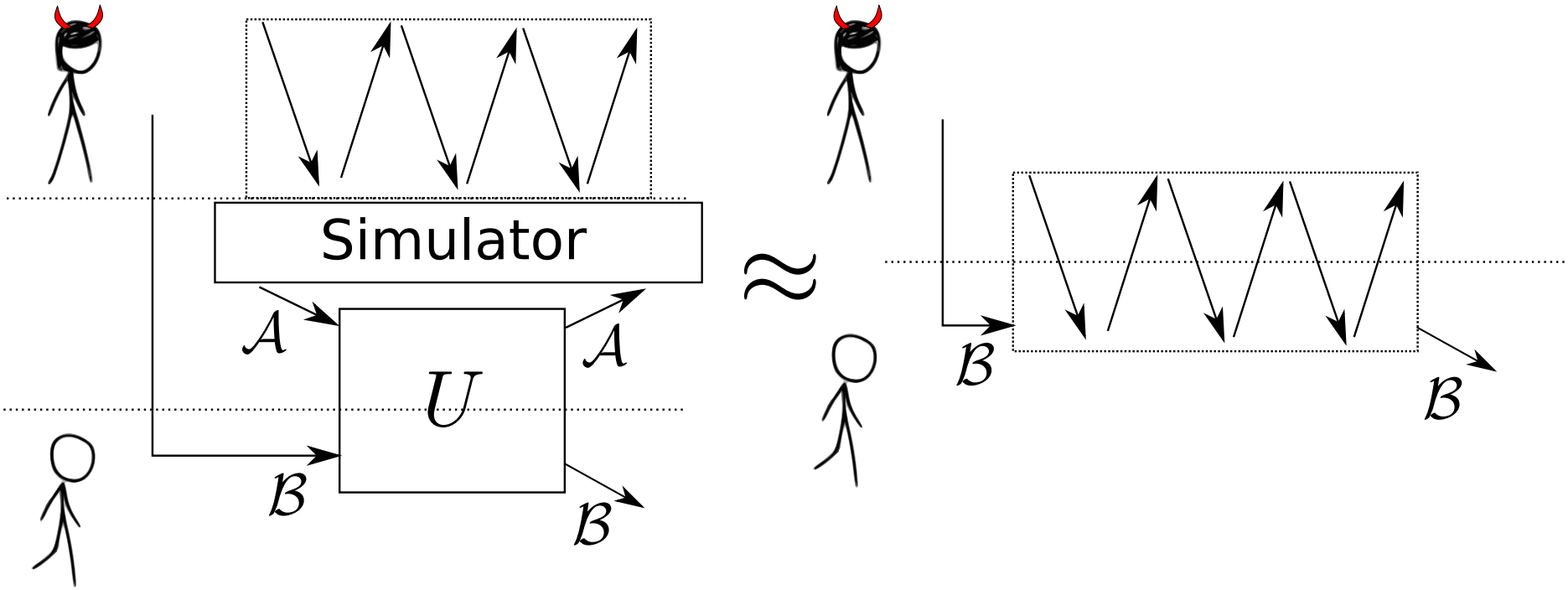
Specious adversaries

Why care about specious adversaries?

- For general security we need two things:
  - Adversary doesn't learn more than they should.
  - Adversary is forced to do what he's supposed to do.
- Security against specious adversaries fulfills the first one.
- This models gets us "halfway there".
- Many results against general adversaries are "compiled" versions of protocols secure against semi-honest adversaries.

Defining security against specious adversaries

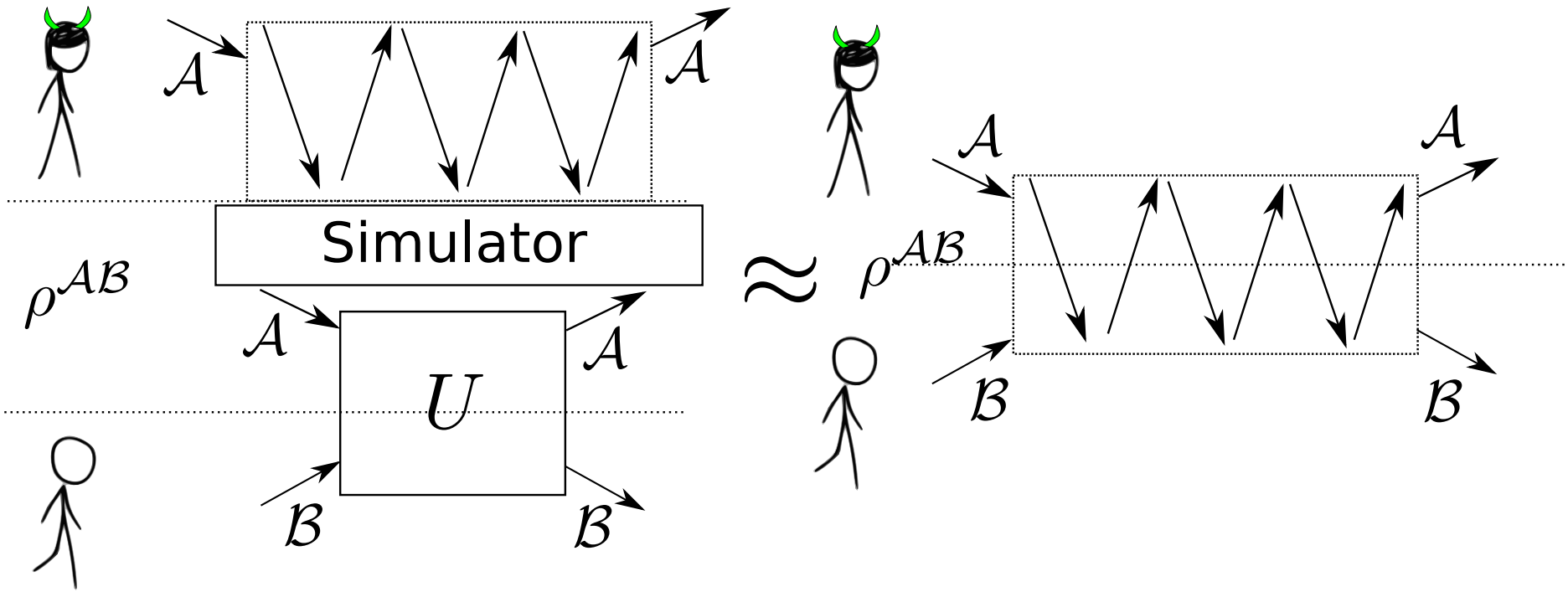
Recall the way we define security against general adversaries:



for all ρ.

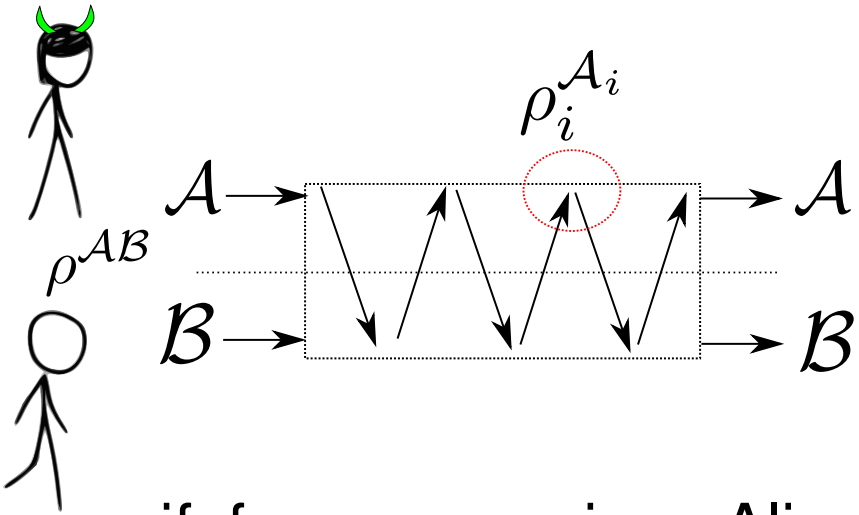
Defining security against specious adversaries

First, now that Alice is specious, she *does* have an input:

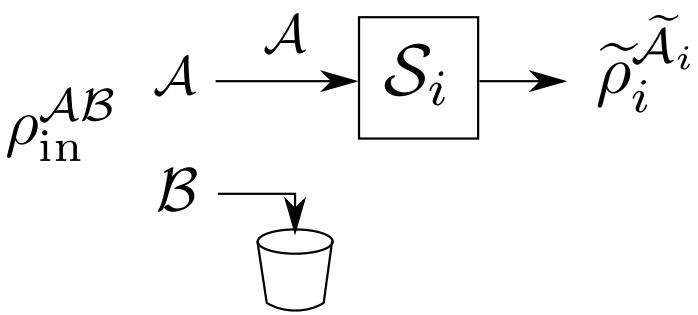


for all $\rho^{\mathcal{AB}}$.

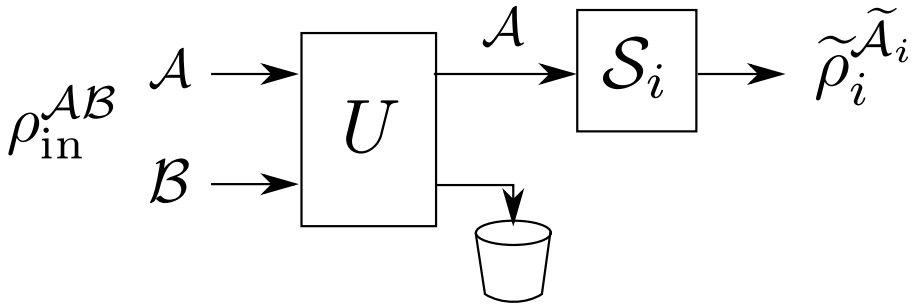
Defining security against specious adversaries



The protocol is *secure* if, for any specious Alice (or Bob), $\exists$ a quantum simulator $\mathcal{S}_i : \mathcal{L}(A) \to \mathcal{L}(A_i)$ such that $\forall$ input states $\rho^{\mathcal{AB}}$, either

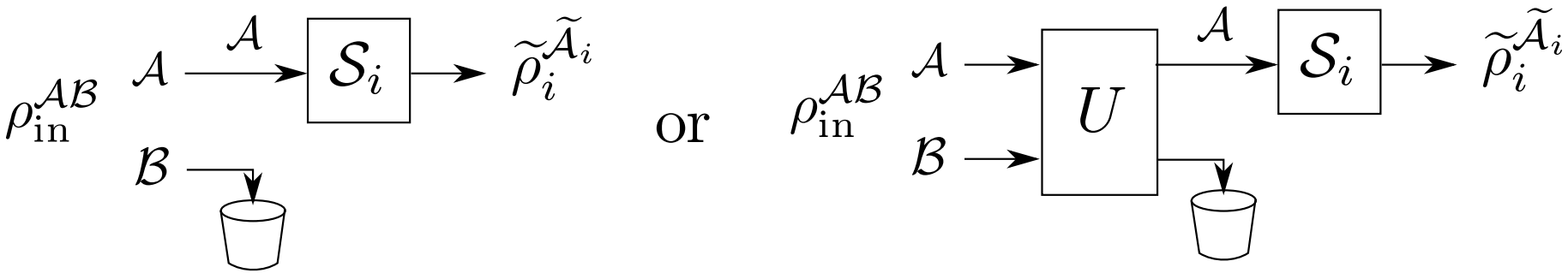


or



Defining security against specious adversaries

The protocol is *secure* if, for any specious Alice (or Bob), $\exists$ a quantum simulator $\mathcal{S}_i : \mathcal{L}(\mathcal{A}) \to \mathcal{L}(\mathcal{A}_i)$ such that $\forall$ input states $\rho_{\text{in}}^{\mathcal{AB}}$, either



In other words, the current state can either be obtained from the legitimate input alone (first case) or from the legitimate output alone (second case).

Defining security against specious adversaries

- This cannot be done for general quantum circuits without any assumptions.
- This *doesn't* follow from the impossibility of quantum bit commitment: most classical primitives (bit commitment, OT, etc) cannot be implemented securely by any bipartite unitary.
- However, we can show that the circuit swapping two qubits cannot be implemented securely in this model.

Impossibility in the bare model

Impossibility in the bare model

Imagine a protocol for secure SWAP: At any point in the protocol, there are two possibilities:

- Alice's state depends only on her input
- Then, since the global state is pure, Bob can recover his own input
- Hence, both Alice's and Bob's simulator work from their respective inputs

- Alice's state depends only on her output ($=$ Bob's input)
- Then, since the global state is pure, Bob can recover his output.
- Hence, both Alice's and Bob's simulator work from their respective outputs

Hence, at all points of the protocol, either Alice and Bob both have their original inputs, or they both have their outputs.

Impossibility in the bare model

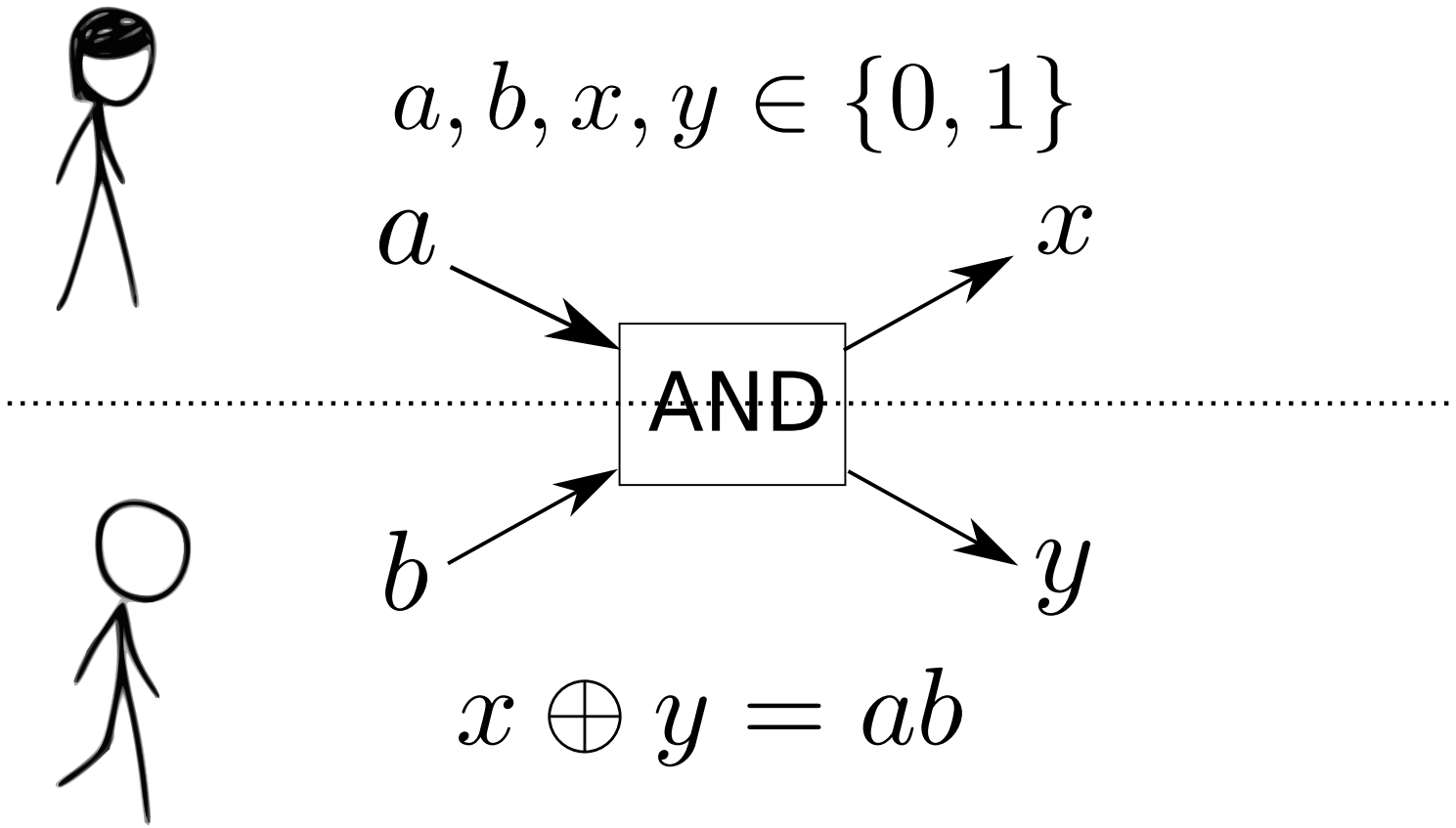
- There must therefore be a step in the protocol when one transitions from the first situation to the second one.
- But during that step, one player must lose the ability to reconstruct his/her state after *receiving* a message!
- This is therefore impossible.

Our protocol

Our protocol

As mentioned earlier, our protocol will require two cryptographic primitives:

- Quantum SWAP
- Distributed AND-box (equivalent to oblivious transfer):



Our protocol

Basic sketch of our protocol:

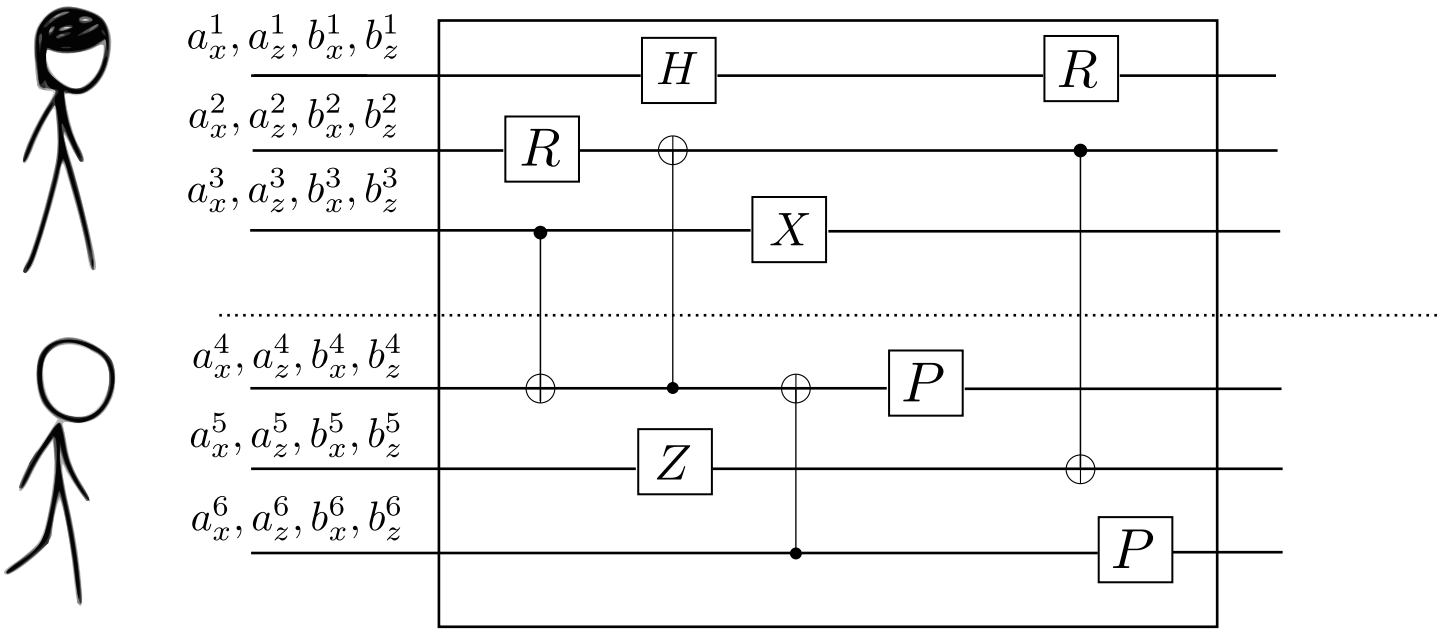
- We represent the circuit U by gates in the universal set $\{X, Y, Z, H, P, R, \mathrm{CNOT}\}$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \qquad P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad R = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

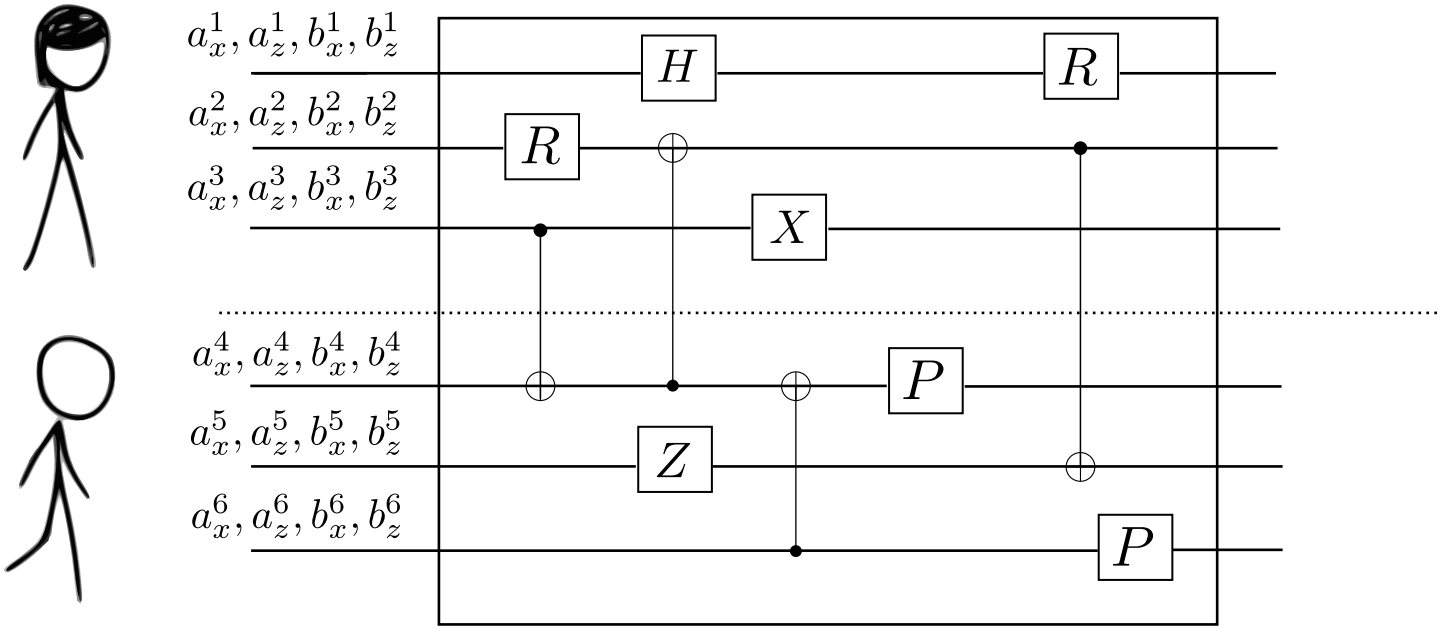
$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Our protocol



- We will keep each wire encrypted with two keys: one for Alice and one for Bob. The wire i is encrypted with the operation $X^{a_x^i \oplus b_x^i} Z^{a_z^i \oplus b_z^i}$.
- We will execute every gate as a subprotocol on encrypted wires.

Our protocol



- We will then use a SWAP gate to exchange keys at the end.

Our protocol

Circuit evaluation phase:

- Start with all keys initialized to 0. (We don't need to encrypt until we have to send things to the other party)
- For Pauli gates (X, Y, Z): Just execute the gates, no need to change the keys.
- For H, P and local CNOT: Execute the gates locally, and update the keys locally; no need for interaction.
- The tough gates: Nonlocal CNOT and R:
  - Nonlocal CNOT can be done with just interaction between Alice and Bob (no primitive required)
  - Each instance of R requires one use of the AND-box to update the keys.

Our protocol

- We then use a quantum SWAP at the end to exchange keys.
- Note that the SWAP at the end can also be implemented by a classical SWAP, but a classical SWAP is *stronger* than a quantum SWAP: we can swap qubits by first encrypting them, sending them directly, and then swapping the classical keys.

Our protocol

Note: If U is Clifford:

- No R gates in the circuit
- Only primitive needed: SWAP at the end.
- Clifford unitaries are “easy”.

Our protocol: Security

How do we prove security? Sketch:

- Simulator for steps before key swap: Just run honest protocol with arbitrary fixed input for the other party.
- After the key swap:
  - Adversary is specious, so she can produce the correct output (for both Alice *and* Bob).
  - This implies that the rest of the adversary's state is uncorrelated with the output.

Open questions

- Can we replace the AND-box by a *unitary* primitive? It would be nice to be able to implement all unitaries with a primitive that is itself a unitary.
- Is there a way to “compile” a protocol secure against specious adversaries to be secure against general adversary? This would greatly simplify the design of quantum protocols.

The End

Thank you!

Brought to you by:

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich