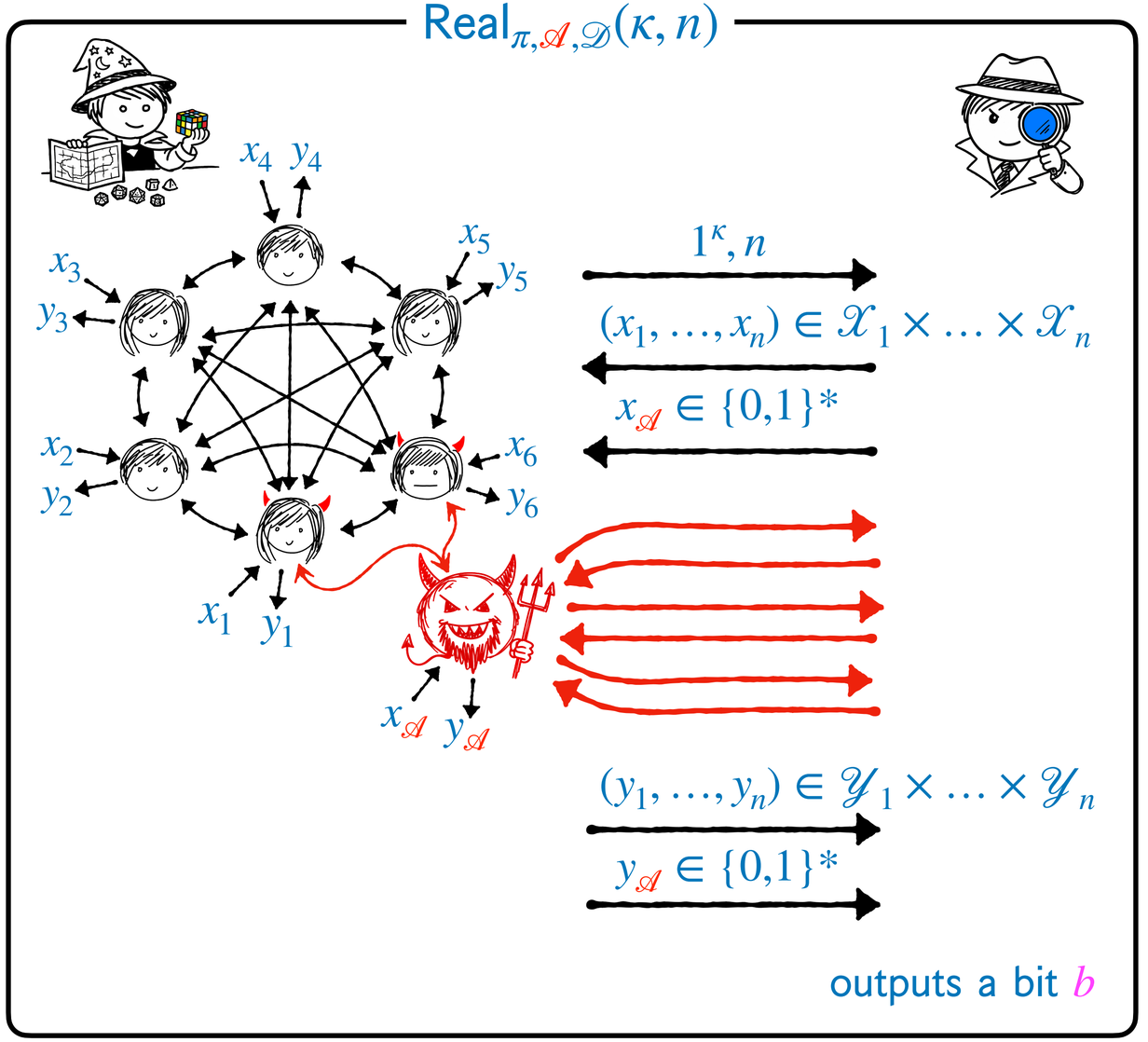
CS4501 Cryptographic Protocols
Lecture 23: Coin Tossing, Cleve's Bound, Commitments, Blum's Protocol

https://jackdoerner.net/teaching/#2026/Spring/CS4501

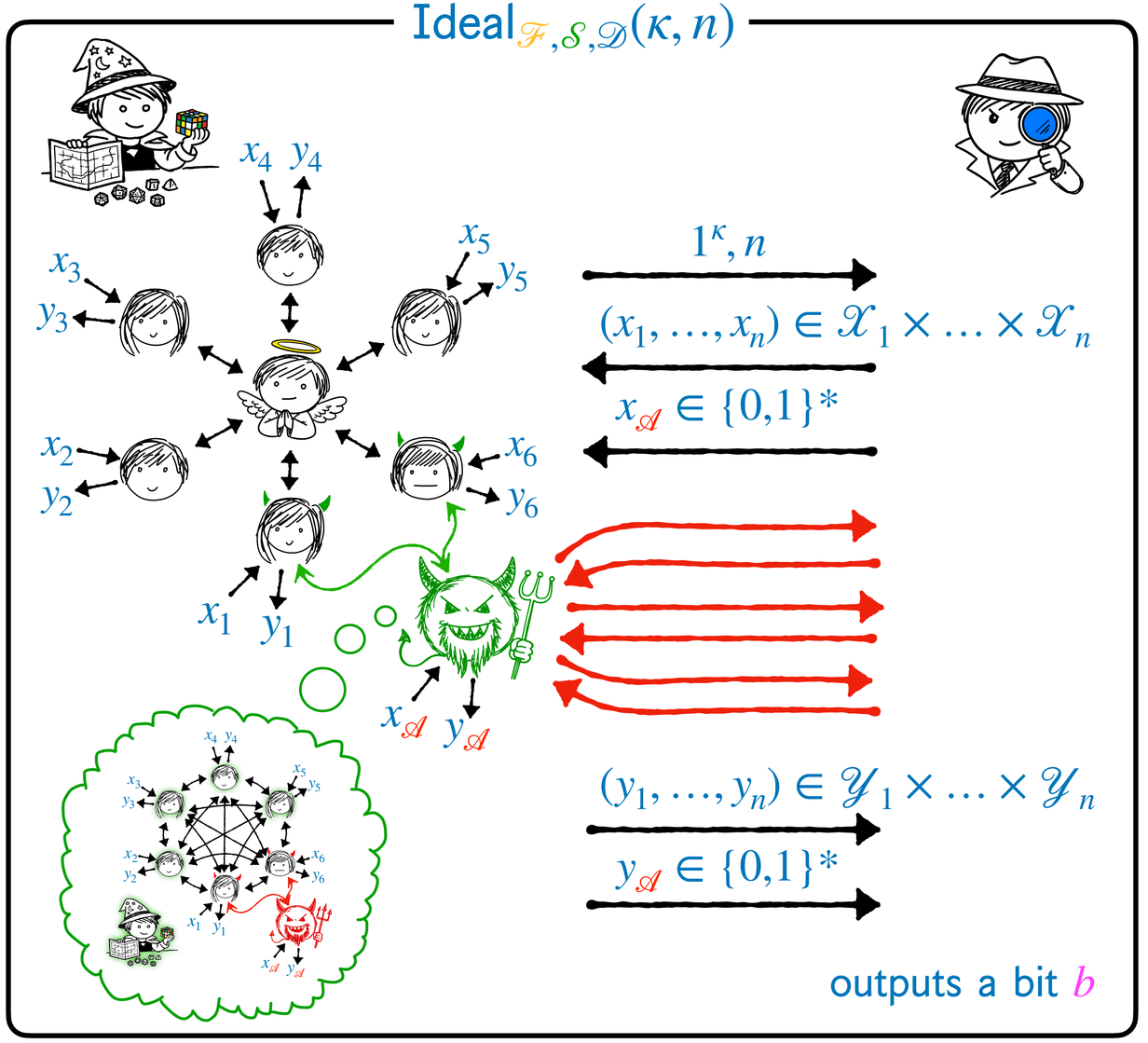
Recap **Def 1**: The Real-World Experiment

1. The challenger communicates the parameters.
2. $\mathscr{D}$ supplies inputs for all parties and $\mathscr{A}$.
3. $\mathscr{A}$ corrupts some parties.
4. The protocol runs. During this time, $\mathscr{A}$ fully controls any corrupted parties.
5. $\mathscr{A}$ may send messages to $\mathscr{D}$, and receive responses.
6. The protocol ends when everyone halts with some output (including $\mathscr{A}$). These outputs are sent to $\mathscr{D}$.
7. $\mathscr{D}$ outputs a bit.

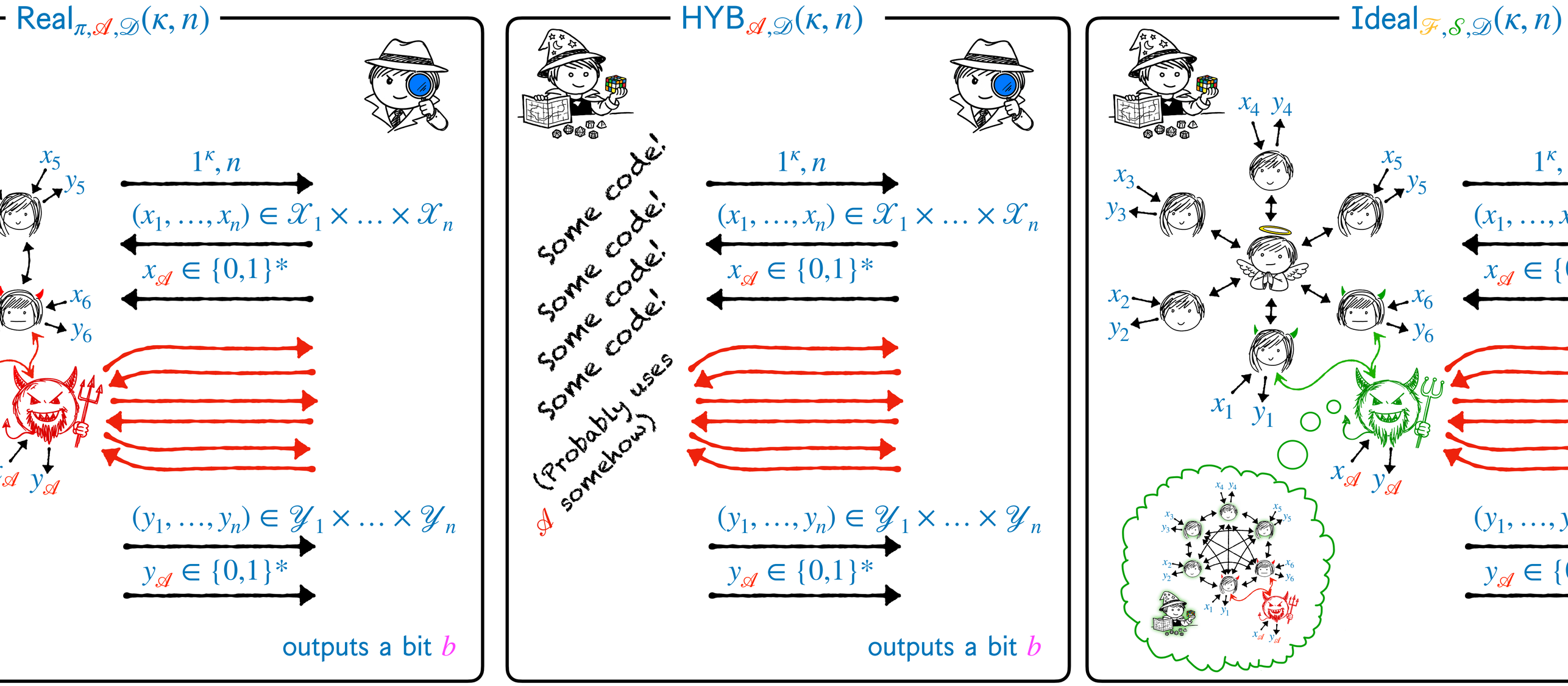


Recap **Def 2**: The Ideal-World Experiment

1. The challenger communicates the parameters.
2. $\mathscr{D}$ supplies inputs for all parties and $\mathscr{A}$ (but it is received by $\mathscr{S}$).
3. $\mathscr{S}$ corrupts some parties.
4. The "dummy protocol" runs. During this time, $\mathscr{S}$ fully controls any corrupted parties (but they only interact with $\mathscr{F}$).
5. $\mathscr{S}$ may send messages to $\mathscr{D}$ (pretending to be $\mathscr{A}$).
6. The protocol ends when everyone halts with some output (including $\mathscr{S}$). These outputs are sent to $\mathscr{D}$.
7. $\mathscr{D}$ outputs a bit.

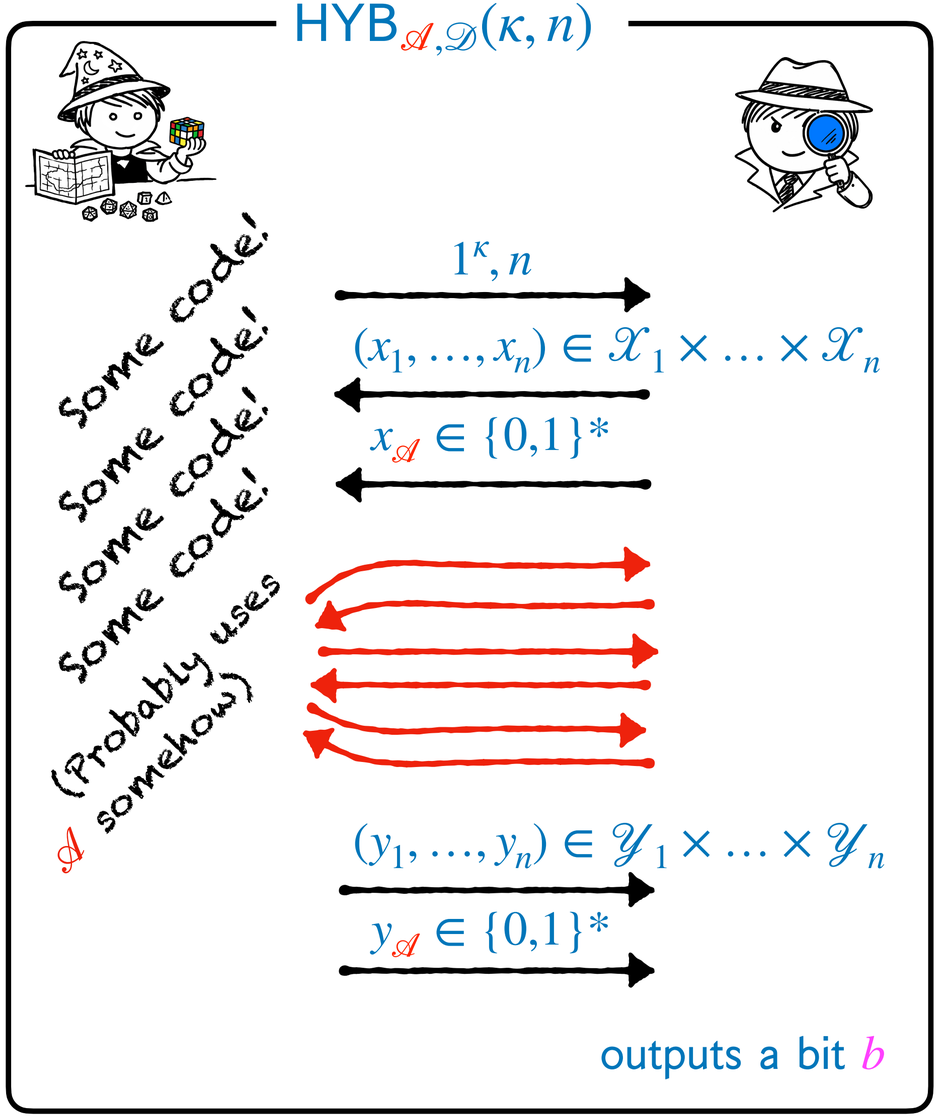


What does a Hybrid Experiment Look Like?



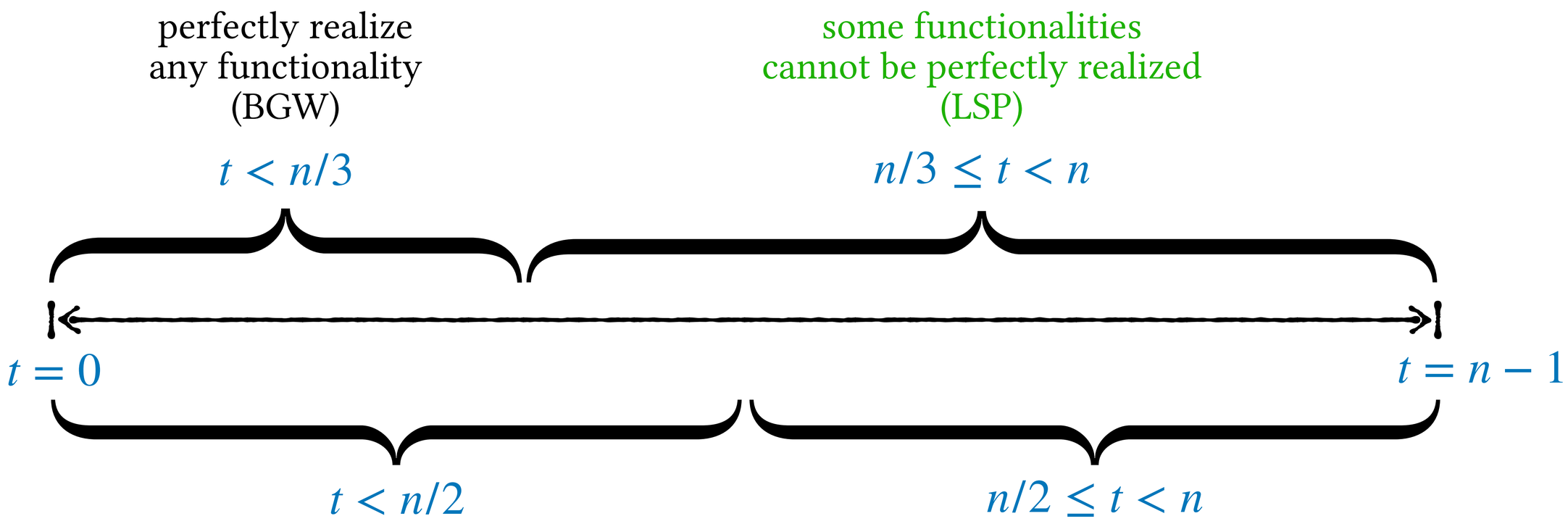
What does a Hybrid Experiment Look Like?

- Usually, a sequence of hybrid experiments will construct $\mathcal{S}$ piece by piece around $\mathcal{A}$, in such a way that $\mathcal{A}$ never notices the difference.
- As in the semi-honest case, the code that the challenger runs in the hybrid experiment probably won't "make sense" in the real or ideal worlds. In particular, it might require knowledge of all of the honest state.
- Do the relationships between the parts seem intricate? Usually, the functionality, simulator, and hybrids must all be co-designed by the cryptographer. This is part of the art!



Where are We? The Landscape of Malicious Security

(Let n be the number of parties and t be the number of corruptions)



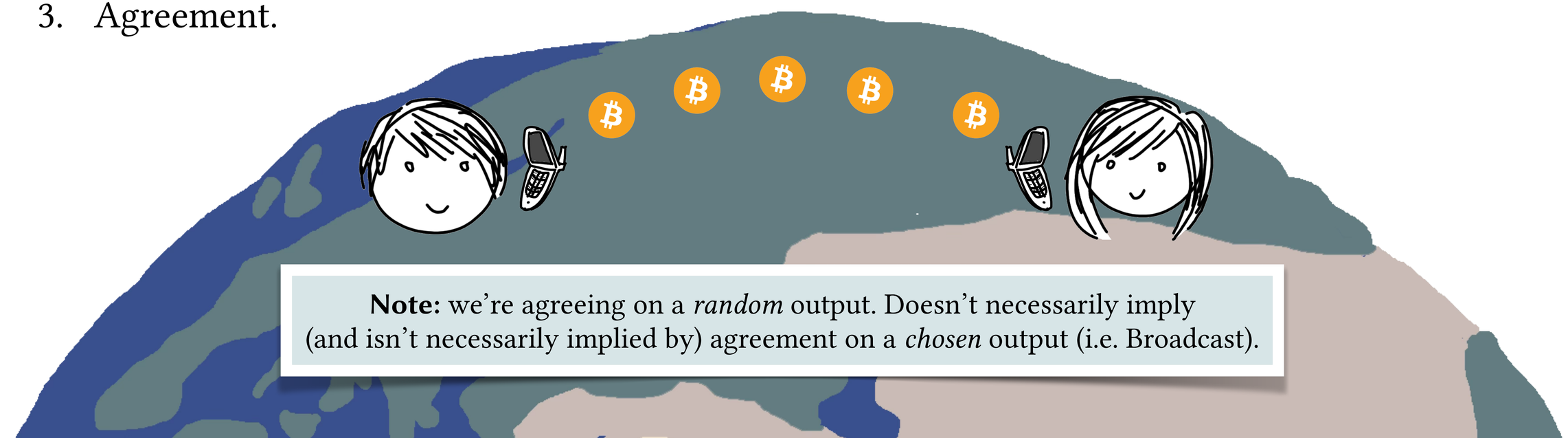
One of my goals in this class is to convince you that Cryptographic Protocols solve practical problems.

Imagine you went on a date and bonded over your favorite one-way function candidates.

Next relationship milestone: your first argument.

Alice and Bob Can't Agree on Dinner.

- Alice always wants Stinky Tofu. Bob prefers Surströmming.
- The only *fair* way to resolve this dispute is to flip a coin. Unfortunately, Alice and Bob live very far apart and they're talking on the phone.
- Can they use a protocol to toss a fair coin over the telephone? They need three properties:
  1. Bias-freeness (so everyone gets what they want about half the time).
  2. Guaranteed Output (so they don't go hungry).
  3. Agreement.



Note: we're agreeing on a *random* output. Doesn't necessarily imply (and isn't necessarily implied by) agreement on a *chosen* output (i.e. Broadcast).

Property-Based Coin Tossing

Definition 3. Let $\kappa, n, t \in \mathbb{N}$ such that $t < n$, and let $P_1, \ldots, P_n$ be a group of parties that are connected by synchronous, authenticated channels. Let π be a protocol in which every P_i produces an output $y_i \in \{0,1\}$. We say that π is a (t, δ)-secure *Coin Tossing* protocol if the following conditions hold when $\mathcal{A}$ maliciously corrupts up to t parties:

1. **Consistency:** All honest parties output the same bit y with probability all-but-negligible in κ.
2. **Bias-Freeness:** $\left|\Pr[y = 0] - 1/2\right| \leq \delta(\kappa)$. We call δ the *bias*.

Implicitly, protocols meeting the above definition also have **Guaranteed Ouput** because the parties always produce an output $y_i \in \{0,1\}$.

We can *informally* define a weaker notion of *Coin Tossing with Abort*, in which $y_i \in \{0,1,\perp\}$ and uniformity implies that *either* $y = b$ where $b \leftarrow \{0,1\}$ is uniform, *or* $y = \perp$. $\mathcal{A}$ gets to choose between these two outcomes, but it cannot force $y = 1 - b$. We will formalize this idea later.

How do the above definitions relate to Coin Tossing Functionalities?

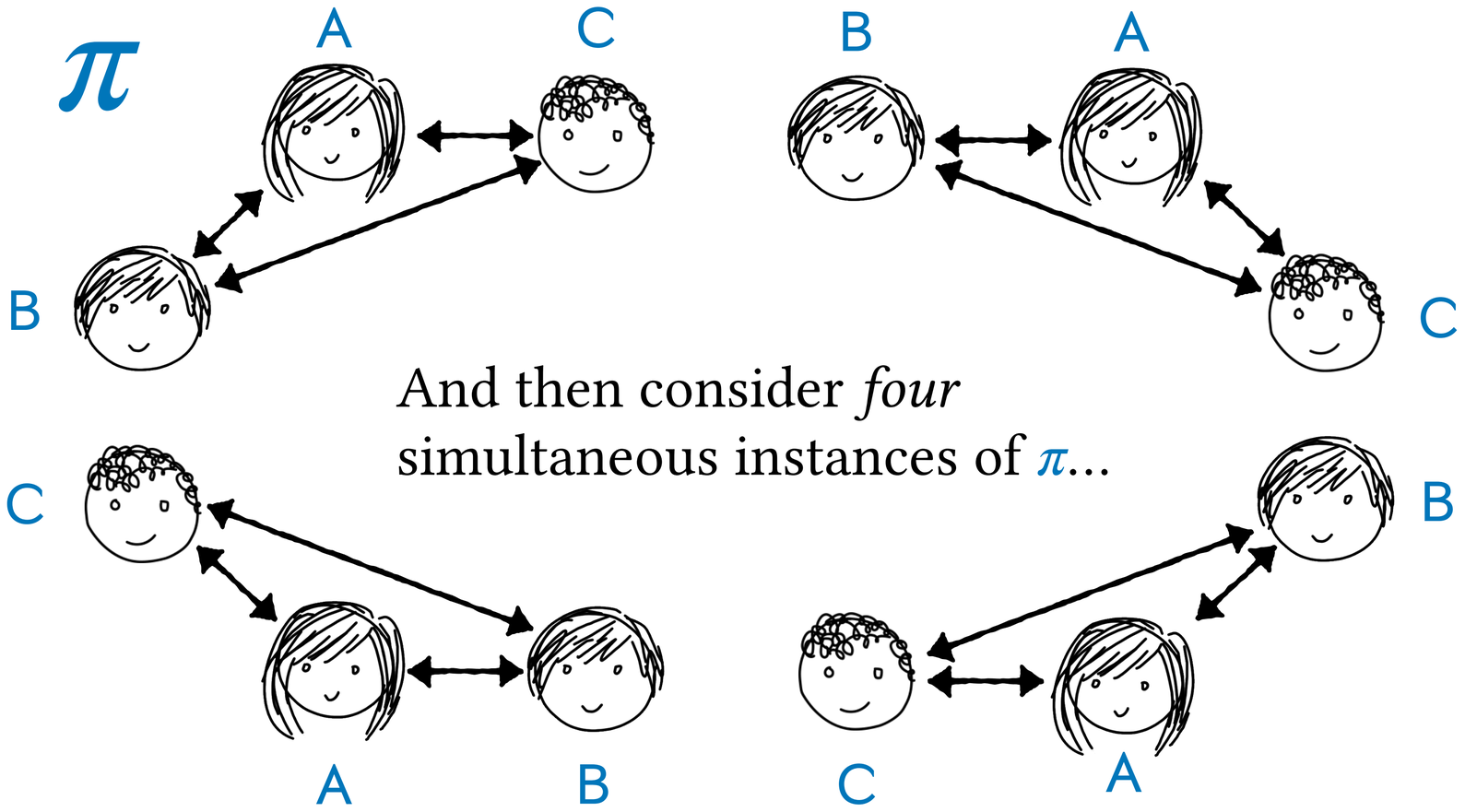
Surveying the Landscape

- In the semi-honest setting, you already solved this problem for homework! Think about your solution. *Would it work very well if someone was malicious?*
- We didn't talk about it, but it's possible to perfectly realize any functionality using the malicious BGW protocol if $t < n/3$. This includes coin tossing functionalities!
- What if $t = n/3$?

Theorem 1: There does not exist a 3-party $(1,\delta)$-secure coin-tossing protocol in the plain model, for *any* nontrivial δ.

Proof: No plain model 3-party 1-secure CT.

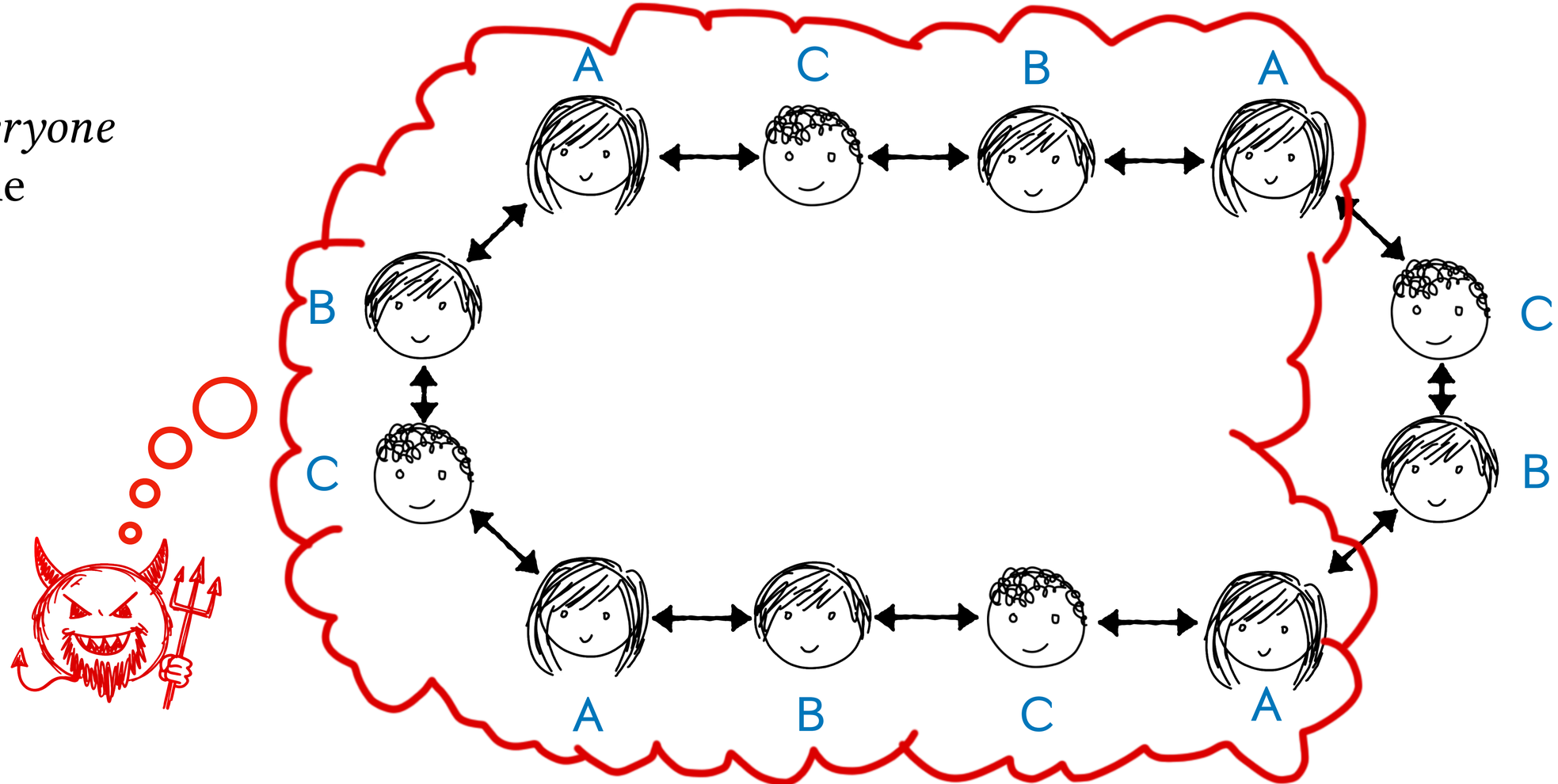
Assume towards contradiction that a 3-party $(1,\delta)$-secure Coin Tossing protocol π exists. Suppose that π has ρ rounds (for concreteness, we will say $\rho = 3$, but this is easy to generalize).



Proof: No plain model 3-party 1-secure CT.

If we reconnect them in a ring, the *consistency* property guarantees each pair of neighbors must still output the same bit (because corrupt A could imagine the rest).

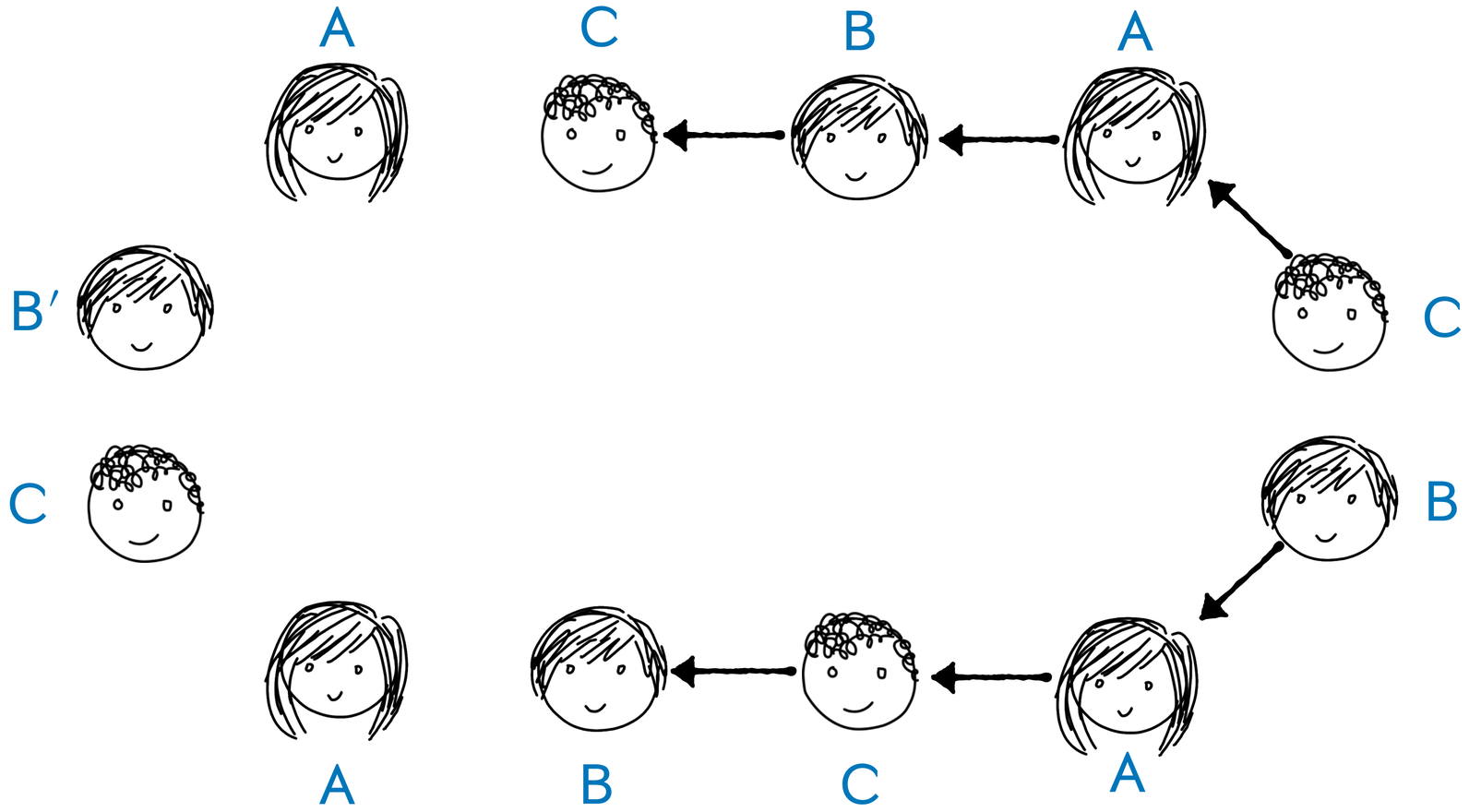
This implies *everyone* must outputs the same bit!



Proof: No plain model 3-party 1-secure CT.

Notice: since we have only *three synchronous rounds*, information from parties B and C on the right-hand side can't possibly propagate to party B′ on the left-hand side.

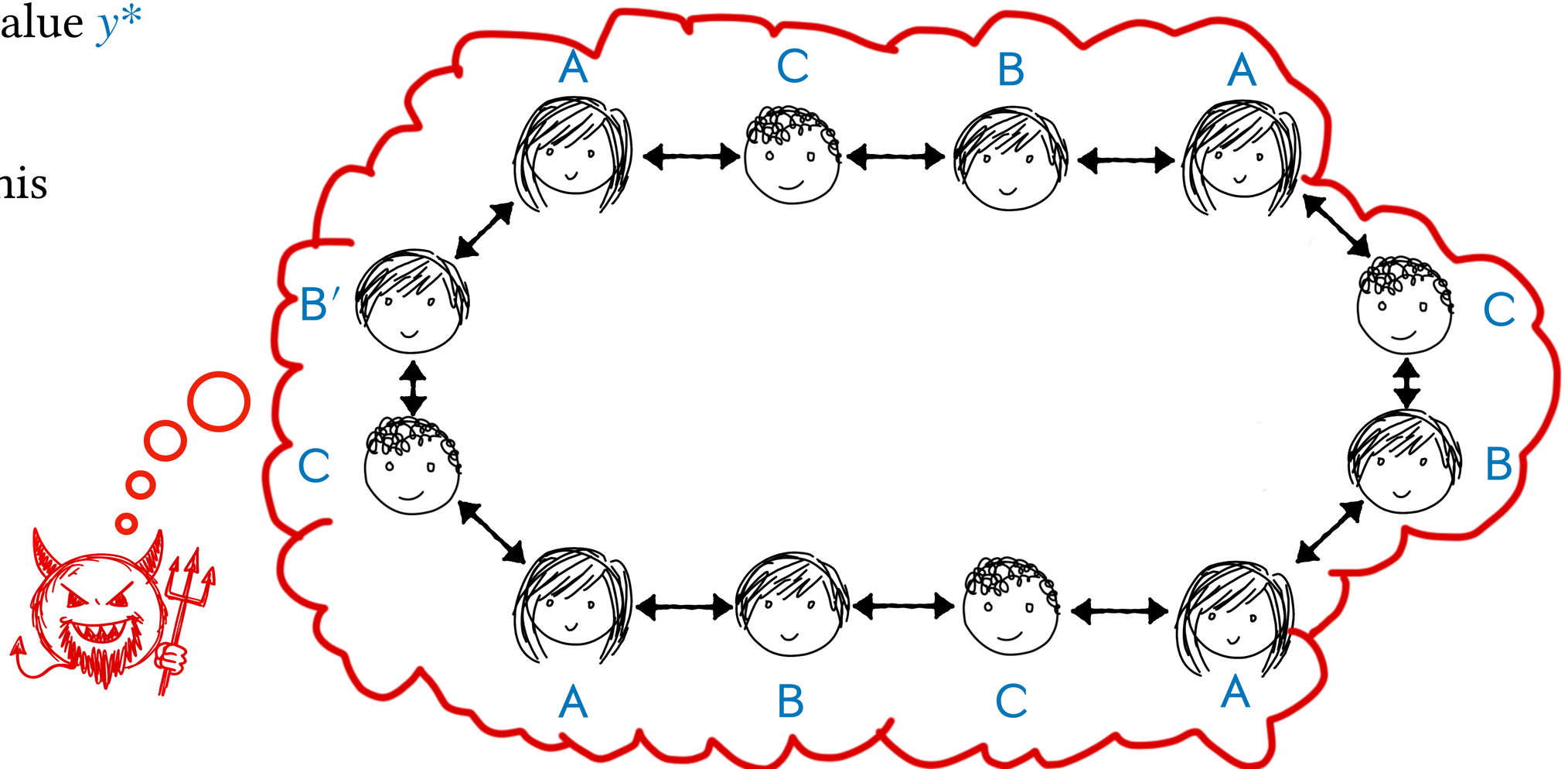
Nevertheless they must all output the same bit (spooky!)



Proof: No plain model 3-party 1-secure CT.

Now consider the following attack: First, $\mathscr{A}$ imagines running the big ring protocol in its head. It uses the honest code of all the parties and tries different randomness until B' outputs the value y^* that $\mathscr{A}$ desires.

$\mathscr{A}$ remembers this randomness….

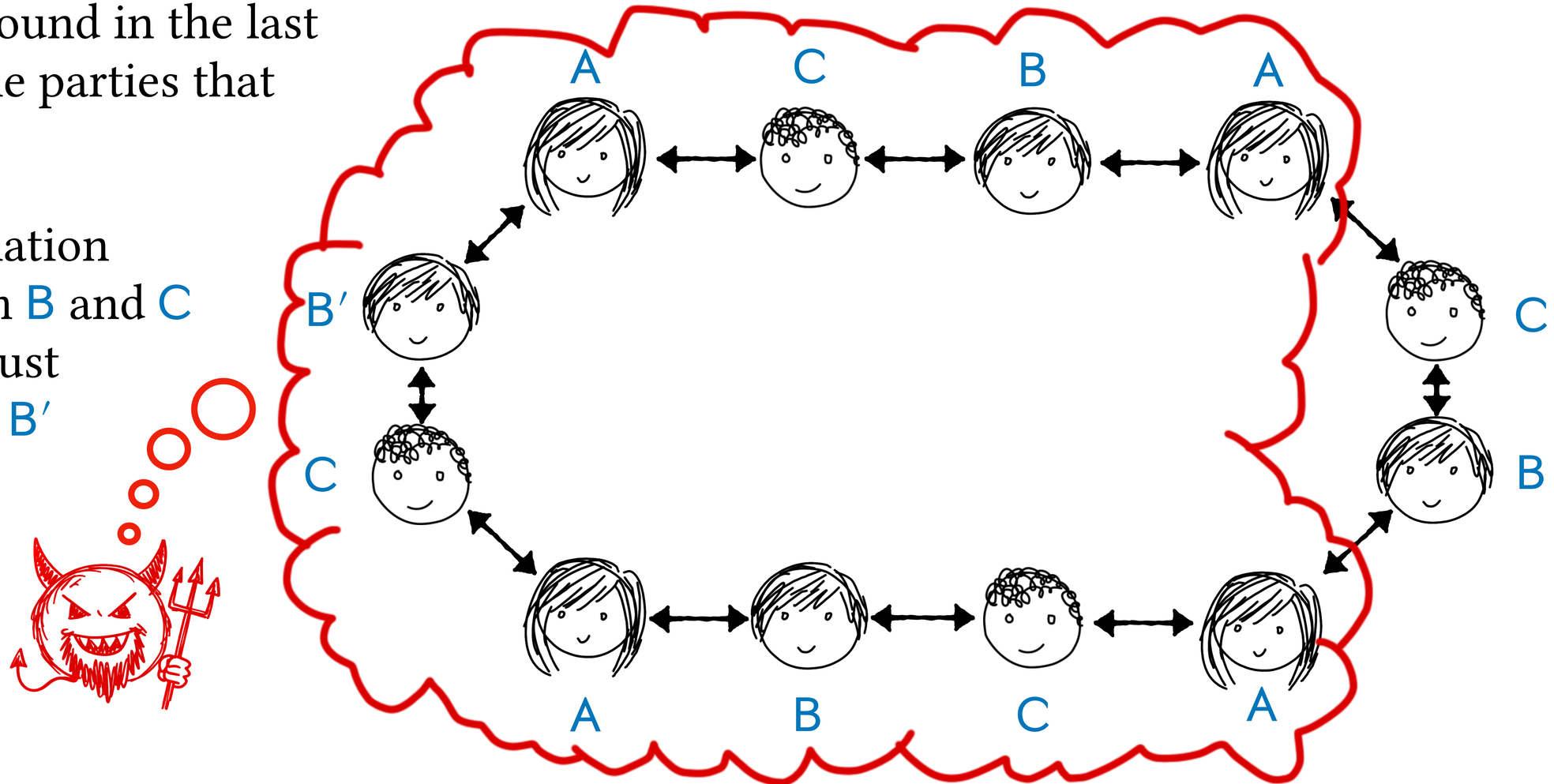


Proof: No plain model 3-party 1-secure CT.

Next, in the 3-party protocol, $\mathscr{A}$ corrupts party A *only*, and emulates the remainder of the big ring (sending the messages to uncorrupted B and C via A). It uses the same randomness it found in the last step for all of the parties that it controls.

Since no information propagates from B and C to party B′, it must be the case that B′ still outputs y^*.

But everyone has to output the same thing!

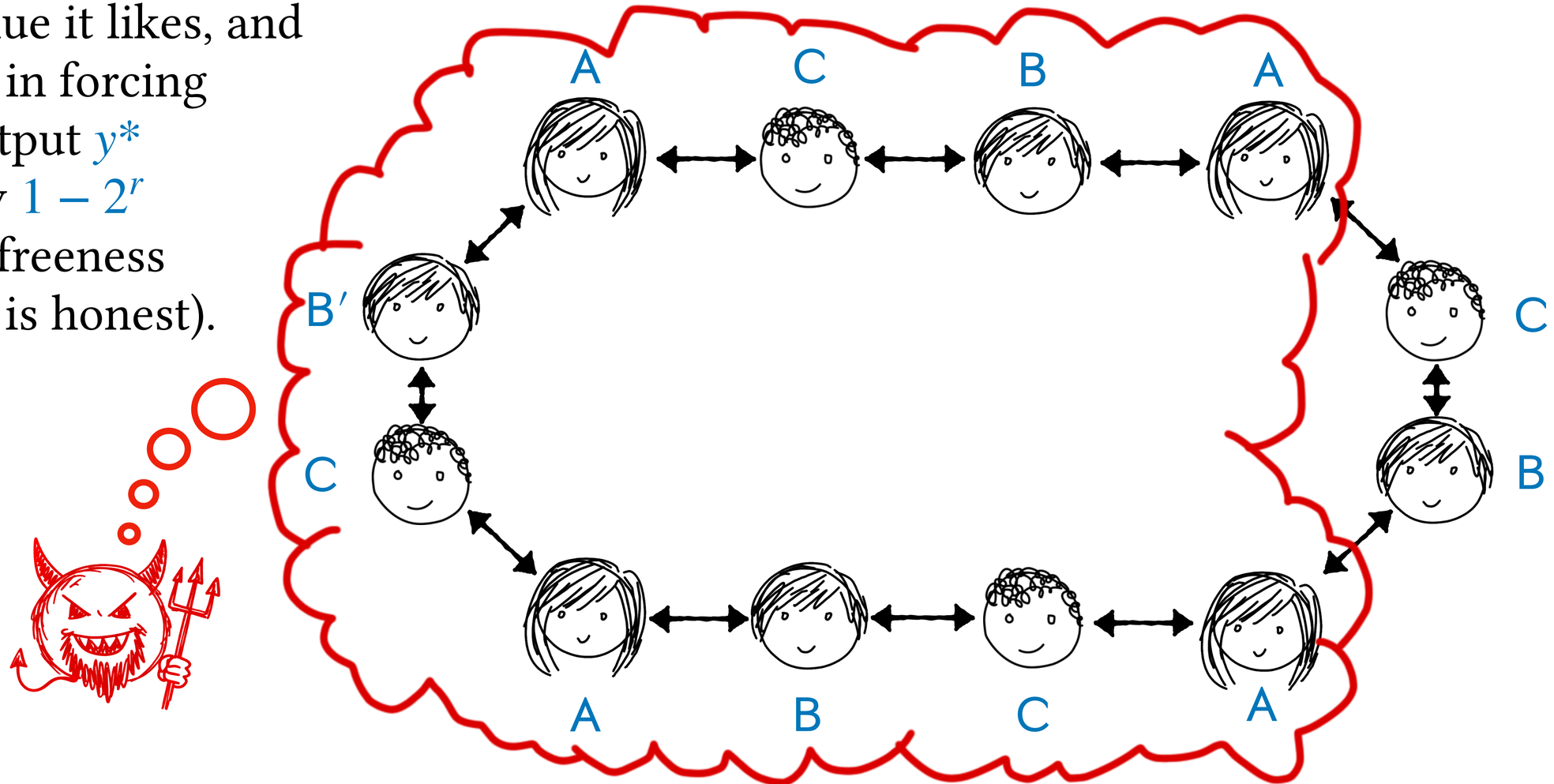


Proof: No plain model 3-party 1-secure CT.

Notice that this adversary is PPT if the protocol is (it simply runs several copies of the honest parties, and retries the experiment in its head r times), that it can choose y^* to be whatever value it likes, and that it succeeds in forcing all parties to output y^* with probability $1 - 2^r$ (assuming bias-freeness when everyone is honest).

Therefore, the protocol π cannot be (even close to) bias-free.

∎



Surveying the Landscape

- In the semi-honest setting, you already solved this problem for homework! Think about your solution. *Would it work very well if someone was malicious?*
- We didn't talk about it, but it's possible to perfectly realize any functionality using the malicious BGW protocol if $t < n/3$. This includes coin tossing functionalities!
- What if $t = n/3$?

Theorem 1: There does not exist a 3-party $(1,\delta)$-secure coin-tossing protocol in the plain model, for *any* nontrivial δ. (This theorem is due to Cohen, Haitner, Omri, Rotem, 2016)

- It seems obvious that, even if broadcast doesn't imply coin tossing, at least it prevents the attack we just saw. So let's assume that the parties have access to $\mathcal{F}_{\mathsf{PKI}}$ and cryptography.
- We might *hope* to get a dishonest majority protocol using powerful tools. As a test case, consider $n = 2$. In this case, using $\mathcal{F}_{\mathsf{PKI}}$ or $\mathcal{F}_{\mathsf{BC}}$ is equivalent to sending ordinary messages.

Theorem 2 (Cleve's Bound, 1984): For every $\rho \in \mathbb{N}$, every 2-party, ρ-round, $(1,\delta)$-secure coin-tossing protocol in the plain model must $\delta \in \Omega(1/\rho)$, even against fail-stop adversaries. (This can be generalized to any $n \in \mathbb{N}$ and $t \geq n/2$, even if we use $\mathcal{F}_{\mathsf{PKI}}$ or $\mathcal{F}_{\mathsf{BC}}$).

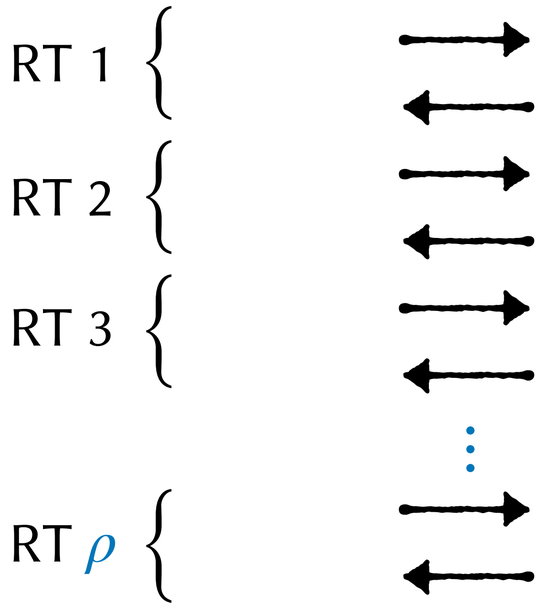
Notice: if we want *negligible* bias, Cleve says that we need *exponentially many* rounds.

Proof: No CT for Dishonest Majority

In the *synchronous* model, we usually permit adversaries to execute *rushing attacks*. This means that in any given round, an adversary can wait to see all the messages of the other participants in that round, before sending its own.

Any time we prove a protocol secure against a rushing adversary in the synchronous model, it remains secure when we take all of the messages transmitted in a single round (which are necessarily independent of one another when computed honestly) and instead transmit them one at a time in contiguous rounds.

Therefore, if we suppose towards contradiction that there exists a 2-party $(1,\delta)$-secure coin-tossing protocol with ρ rounds, we can always transform it into a protocol made up of ρ "round trips" like the one at right.



Proof: No CT for Dishonest Majority

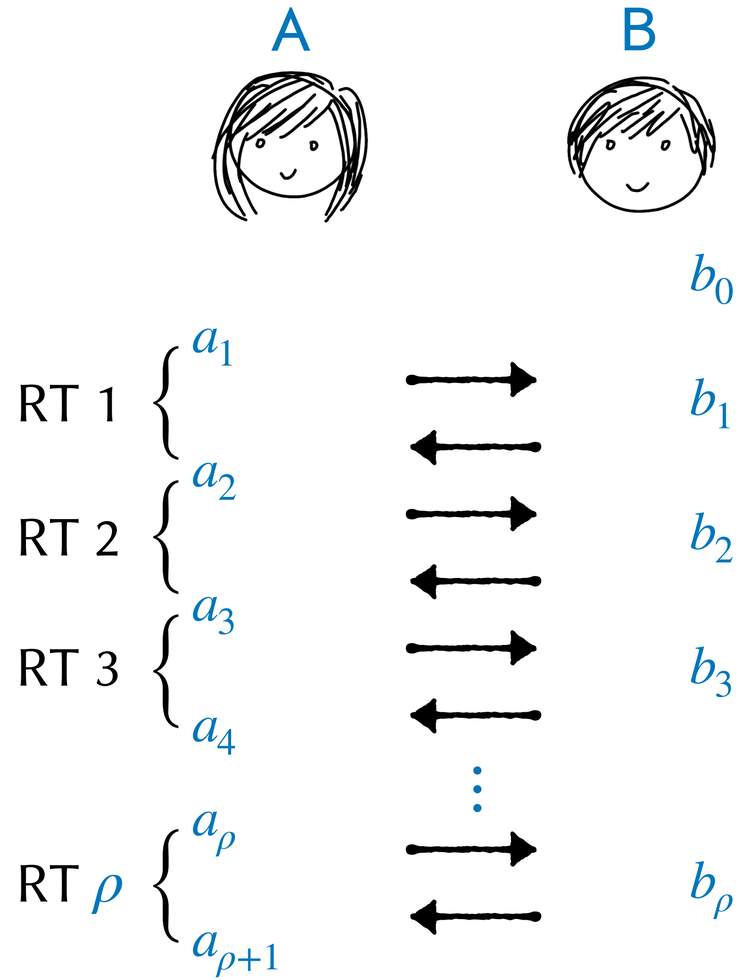
Since $\mathscr{A}$ can corrupt parties maliciously, it is legal for it to instruct the corrupt party to *go silent* from some point onward. We insist the output of an honest party is always in $\{0,1\}$, even if this occurs.

Let a_r for $r \in [\rho]$ denote the output of A if B goes silent starting in round trip r. $a_{\rho+1}$ is the final output if B is honest.

Notice that a_r can be computed *before* A sends its message in round trip r, and that r is fixed by the transcript plus the randomness of A.

At left we labeled when these outputs become available.

Similarly, let b_r for $r \in [0,\rho-1]$ denote the output of B if A goes silent in round trip $r+1$. B can compute b_r before it sends its message in round trip r.



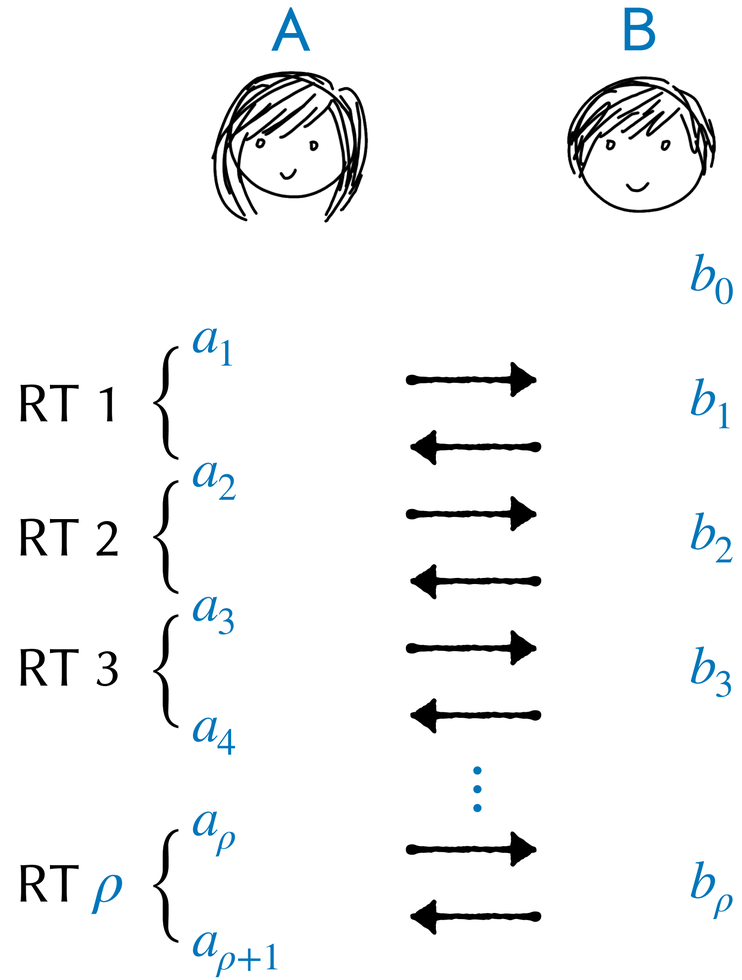
Proof: No CT for Dishonest Majority

Intuition: If $\mathscr{A}$ prevents any messages from ever being sent, then the outputs of the honest party is (nearly) uniform and uncorrelated with anything in the corrupt party's view.

On the other hand, the consistency property says that if the protocol runs to completion, the final outputs of the two parties are almost perfectly correlated (and $\mathscr{A}$ knows the output of the honest party with near certainty).

So there must be some *first* round where $\mathscr{A}$ learns about the correlated final output. If it learns this information before the honest party does, and it doesn't like correlated final output, then it can go silent and force the honest party to use its previous, *uncorrelated* output instead.

We will prove formally that one of the two can induce a bias in the other party's output using this strategy.



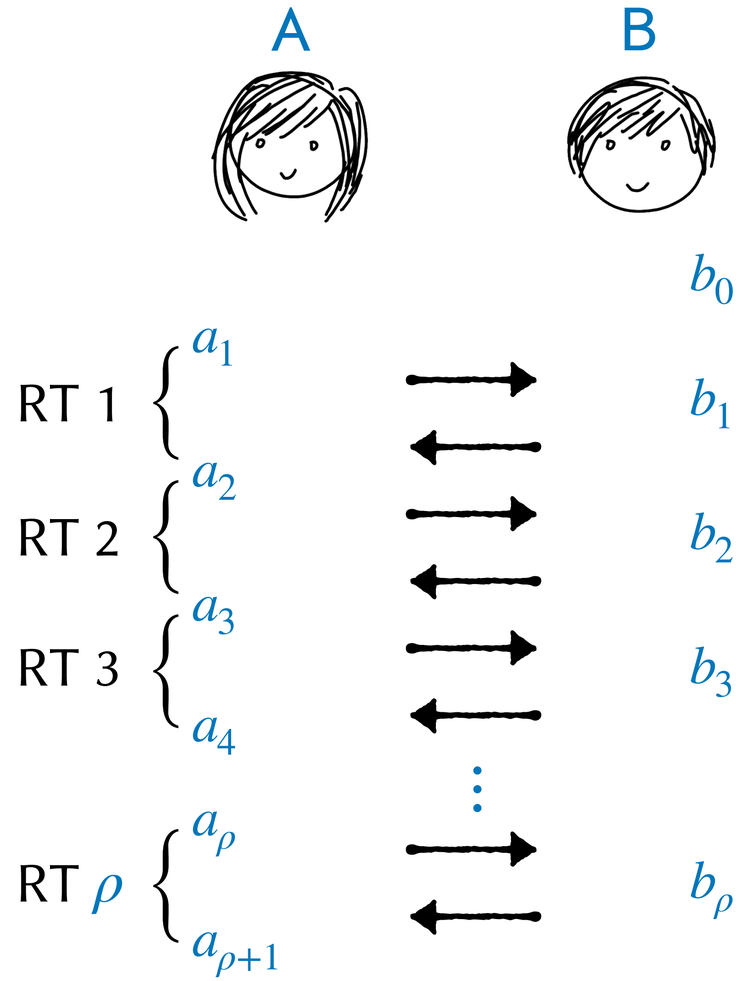
Proof: No CT for Dishonest Majority

More Intuition: We think there must be *some* round where a correlation forms between the parties' outputs, and that the adversary can use this to inject a bias, but we don't know which round.

For each round, we can define an adversarial strategy that performs the attack using that round.

If we're right, then at least one of strategies must be effective at inducing bias, but we still don't know which.

What if we average them? If there are polynomially many rounds then there are polynomially many strategies. So if one strategy injects non-negligible bias, then the average bias also should be non-negligible.



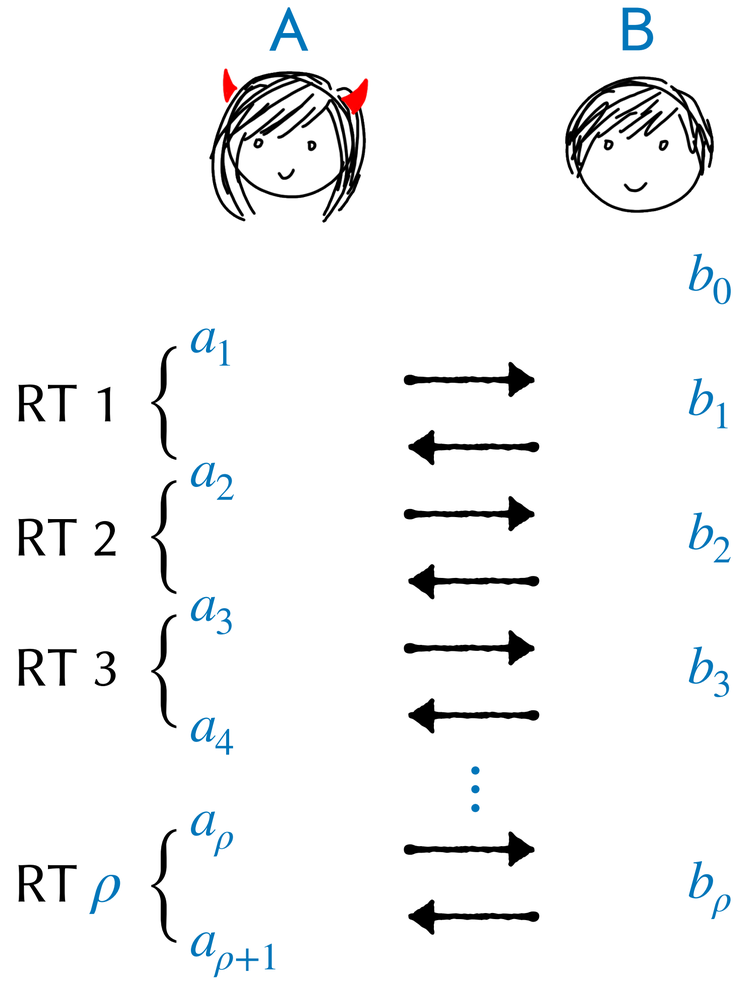
Proof: No CT for Dishonest Majority

Formalizing our intuition: We will define the bias *towards zero* of x as $\Pr[x = 0] - 1/2$. Bias *towards one* is defined similarly. Both are no greater than the overall bias.

Let us consider a simple strategy for a corrupt A, which we will refer to as $\hat{\mathsf{A}}$. Under strategy $\hat{\mathsf{A}}$, corrupted A simply never speaks, and B outputs b_0.

Claim: The bias induced by $\hat{\mathsf{A}}$ upon the output of B is $\max(\Pr[b_0 = 0], \Pr[b_0 = 1]) - 1/2$, where the probability is over the random coins of B.

Next, we will consider 2ρ additional, slightly more sophisticated strategies for each party when it is corrupt, and argue that they each produce a certain bias.



Proof: No CT for Dishonest Majority

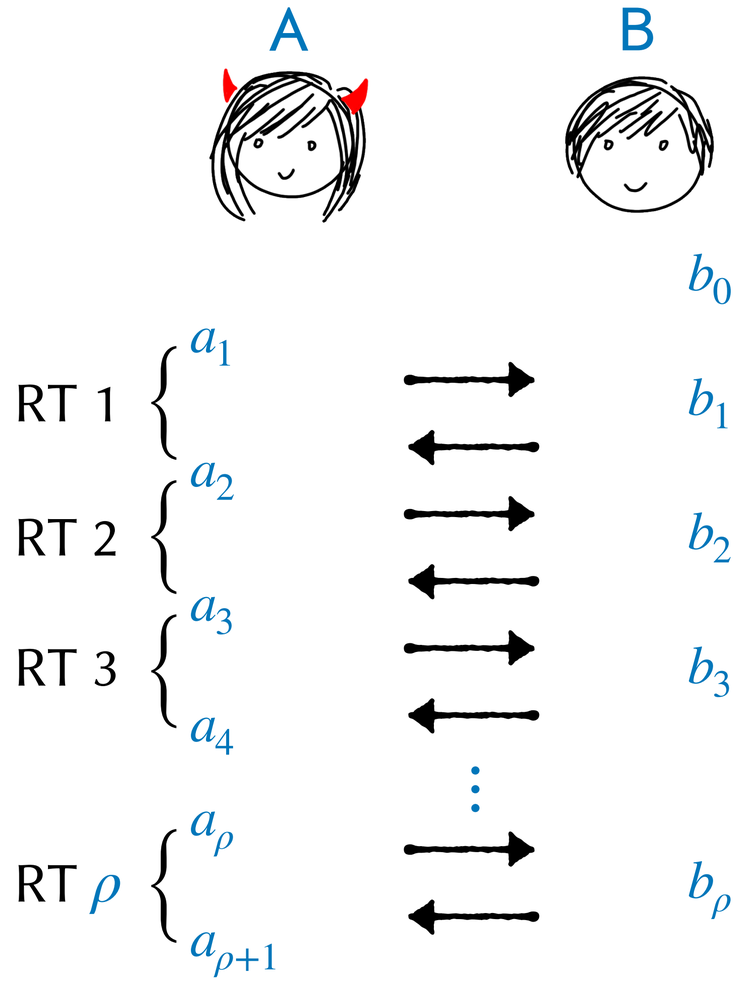
Strategy $\tilde{\mathsf{A}}_r^\beta$ for $r \in [\rho]$ and $\beta \in \{0,1\}$:

1. Run honestly for $r-1$ round trips, then compute a_r.
2. If $a_r = \beta$, run honestly during round trip r, then go silent.
3. If $a_r = 1-\beta$, go silent at the beginning of round trip r.

Intuition: if a_r and b_r are correlated, but a_r and b_{r-1} are not, then this strategy injects a bias towards β.

Claim: when B interacts with $\tilde{\mathsf{A}}_r^0$, the bias towards 0 of its output is $\Pr[a_r = 0 \wedge b_r = 0] + \Pr[a_r = 1 \wedge b_{r-1} = 0] - 1/2$.

Claim: when B interacts with $\tilde{\mathsf{A}}_r^1$, the bias towards 1 of its output is $\Pr[a_r = 1 \wedge b_r = 1] + \Pr[a_r = 0 \wedge b_{r-1} = 1] - 1/2$.



Proof: No CT for Dishonest Majority

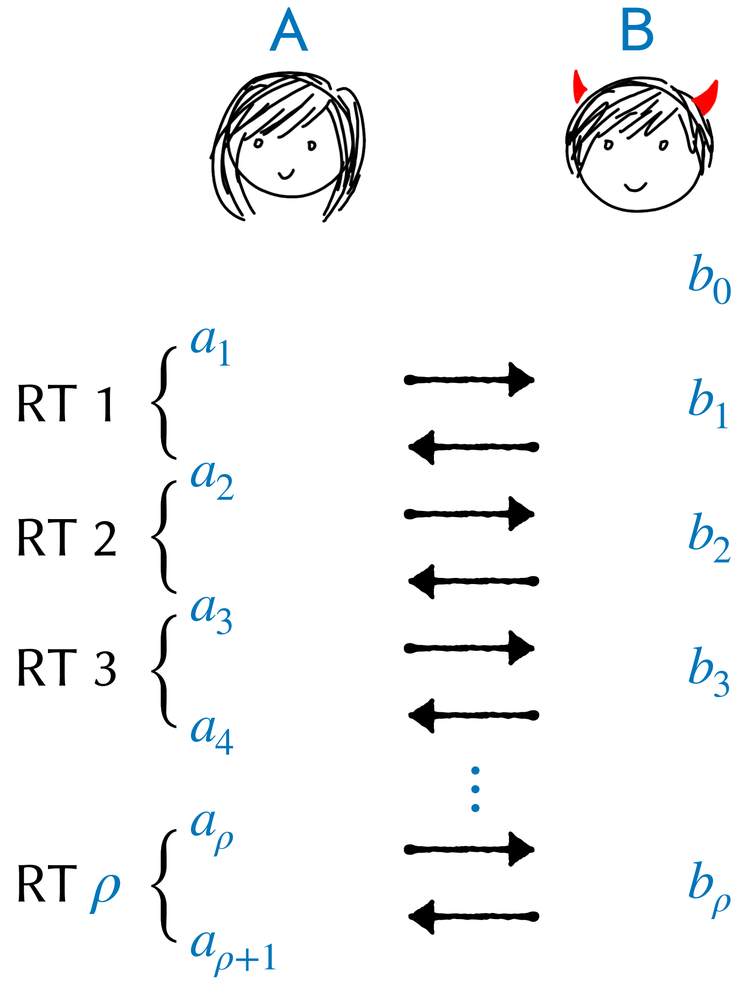
Strategy $\tilde{\mathsf{B}}_r^\beta$ for $r \in [\rho]$ and $\beta \in \{0,1\}$:

1. Run honestly for $r-1$ round trips, then compute b_r.
2. If $b_r = \beta$, run honestly during round trip r, then go silent.
3. If $b_r = 1-\beta$, go silent at the beginning of round trip r.

Intuition: if b_r and a_{r+1} are correlated, but b_r and a_r are not, then this strategy injects a bias towards β.

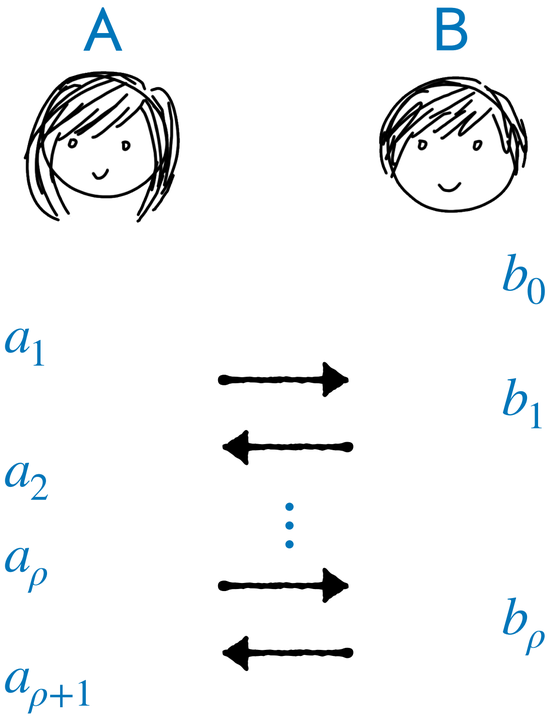
Claim: when A interacts with $\tilde{\mathsf{B}}_r^0$, the bias towards 0 of its output is $\Pr[b_r = 0 \wedge a_{r+1} = 0] + \Pr[b_r = 1 \wedge a_r = 0] - 1/2$.

Claim: when A interacts with $\tilde{\mathsf{B}}_r^1$, the bias towards 1 of its output is $\Pr[b_r = 1 \wedge a_{r+1} = 1] + \Pr[b_r = 0 \wedge a_r = 1] - 1/2$.



Proof: No CT for Dishonest Majority

Now let Δ be the *average* of the biases induced by the $4\rho + 1$ strategies (i.e. $\hat{\mathsf{A}}$, $\tilde{\mathsf{A}}_r^\beta$, and $\tilde{\mathsf{B}}_r^\beta$ for $r \in [\rho]$ and $\beta \in \{0,1\}$) we have just described. We have:

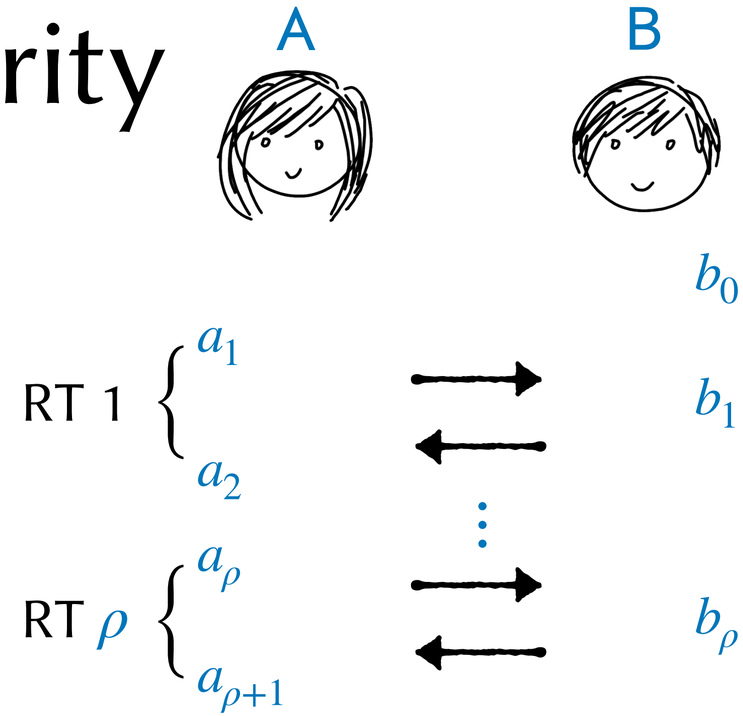


$$\Delta \geq \frac{\left(\begin{array}{l}\sum_{r\in[\rho]}\left(\begin{array}{l}\Pr[a_r = 0 \wedge b_r = 0] + \Pr[a_r = 1 \wedge b_{r-1} = 0] - 1/2 \\ +\Pr[a_r = 1 \wedge b_r = 1] + \Pr[a_r = 0 \wedge b_{r-1} = 1] - 1/2 \\ +\Pr[b_r = 0 \wedge a_{r+1} = 0] + \Pr[b_r = 1 \wedge a_r = 0] - 1/2 \\ +\Pr[b_r = 1 \wedge a_{r+1} = 1] + \Pr[b_r = 0 \wedge a_r = 1] - 1/2\end{array}\right) \\ +\max(\Pr[b_0 = 0], \Pr[b_0 = 1]) - 1/2\end{array}\right)}{4\rho + 1}$$

By the law of total probability, we have for every $r \in [\rho]$:
$\Pr[a_r = 0 \wedge b_r = 0] + \Pr[a_r = 0 \wedge b_r = 1] + \Pr[a_r = 1 \wedge b_r = 0] + \Pr[a_r = 1 \wedge b_r = 1] = 1$.
Notice that all of these terms appear in the above sum! Therefore we can simplify to get...

Proof: No CT for Dishonest Majority

$$\Delta \geq \frac{\left(\sum_{r\in[\rho]}\left(\begin{array}{l}\Pr[a_r=0\wedge b_r=0]+\Pr[a_r=1\wedge b_{r-1}=0]-1/2\\ +\Pr[a_r=1\wedge b_r=1]+\Pr[a_r=0\wedge b_{r-1}=1]-1/2\\ +\Pr[b_r=0\wedge a_{r+1}=0]+\Pr[b_r=1\wedge a_r=0]-1/2\\ +\Pr[b_r=1\wedge a_{r+1}=1]+\Pr[b_r=0\wedge a_r=1]-1/2\end{array}\right)\right.}{4\rho+1}$$
$$\left.\begin{array}{l}\\ \\ \\ \\ +\max(\Pr[b_0=0],\Pr[b_0=1])-1/2\end{array}\right)$$



By the law of total probability, we have for every $r \in [\rho]$:
$\Pr[a_r = 0 \wedge b_r = 0] + \Pr[a_r = 0 \wedge b_r = 1] + \Pr[a_r = 1 \wedge b_r = 0] + \Pr[a_r = 1 \wedge b_r = 1] = 1$.
Notice that all of these terms appear in the above sum! Therefore we can simplify to get...

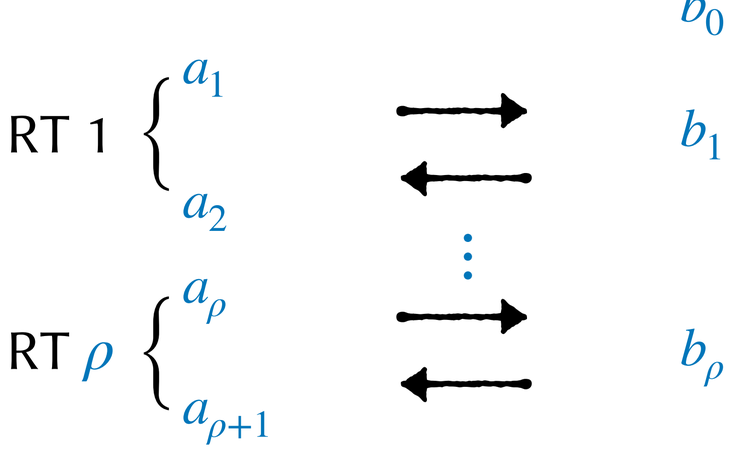
$$\Delta \geq \frac{1}{4\rho+1}\left(\begin{array}{l}\sum_{r\in[\rho]}\left(\begin{array}{l}\Pr[a_r=1\wedge b_{r-1}=0]+\Pr[a_r=0\wedge b_{r-1}=1]-1/2\\ +\Pr[b_r=0\wedge a_{r+1}=0]+\Pr[b_r=1\wedge a_{r+1}=1]-1/2\end{array}\right)\\ +\max(\Pr[b_0=0],\Pr[b_0=1])-1/2\end{array}\right)$$

Proof: No CT for Dishonest Majority

$$\Delta \geq \frac{1}{4\rho+1}\left(\sum_{r\in[\rho]}\begin{pmatrix}\Pr[a_r = 1 \wedge b_{r-1} = 0] + \Pr[a_r = 0 \wedge b_{r-1} = 1] - 1/2\\ +\Pr[b_r = 0 \wedge a_{r+1} = 0] + \Pr[b_r = 1 \wedge a_{r+1} = 1] - 1/2\end{pmatrix} + \max(\Pr[b_0 = 0], \Pr[b_0 = 1]) - 1/2\right)$$

First, notice that because both a_1 and b_0 can be computed *before* any information is exchanged, they must be *independent*, and therefore we have

$$\begin{aligned}\Pr[a_1 = b_0] &= \Pr[a_1 = 0 \wedge b_0 = 0] + \Pr[a_1 = 1 \wedge b_0 = 1]\\ &= \Pr[a_1 = 0] \cdot \Pr[b_0 = 0] + \Pr[a_1 = 1] \cdot \Pr[b_0 = 1]\\ &\leq (\Pr[a_1 = 0] + \Pr[a_1 = 1]) \cdot \max(\Pr[b_0 = 0], \Pr[b_0 = 1])\\ &= \max(\Pr[b_0 = 0], \Pr[b_0 = 1])\end{aligned}$$

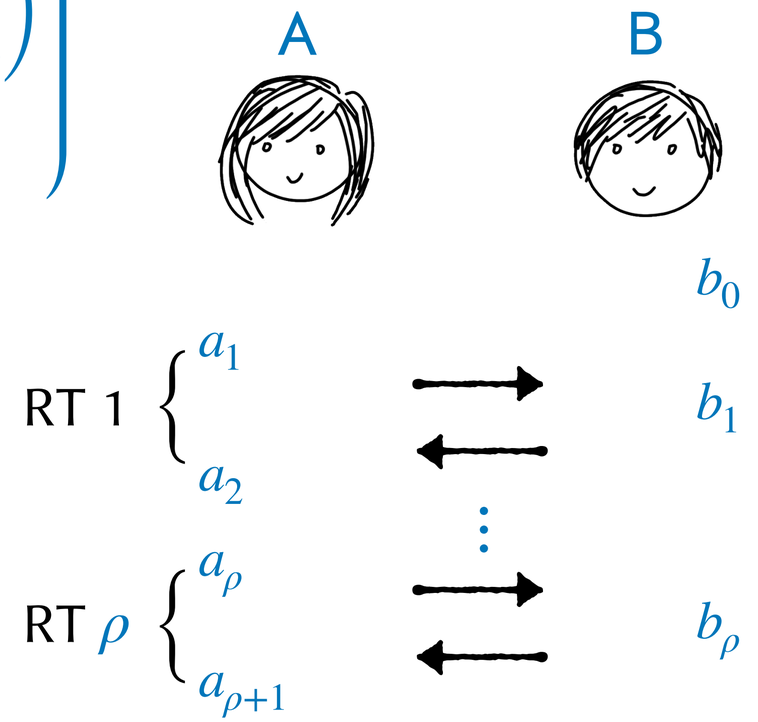


Second, notice also that in the equation for Δ, each element of the sum contains two terms involving a_r, b_{r-1} and two involving a_{r+1}, b_r. If we re-indexing and re-associating the terms in the sum, and also plug in $\max(\Pr[b_0 = 0], \Pr[b_0 = 1]) \geq \Pr[a_1 = 0 \wedge b_0 = 0] + \Pr[a_1 = 1 \wedge b_0 = 1]$, we get...

Proof: No CT for Dishonest Majority

$$\Delta \geq \frac{1}{4\rho+1}\left(\sum_{r\in[\rho]}\begin{pmatrix}\Pr[a_r=1\wedge b_{r-1}=0]+\Pr[a_r=0\wedge b_{r-1}=1]-1/2\\ +\Pr[b_r=0\wedge a_{r+1}=0]+\Pr[b_r=1\wedge a_{r+1}=1]-1/2\end{pmatrix}\right.$$
$$\left.+\max(\Pr[b_0=0],\Pr[b_0=1])-1/2\right)$$

$$\geq \frac{\left(\sum_{r\in[\rho]}\begin{pmatrix}\Pr[a_r=1\wedge b_{r-1}=0]+\Pr[a_r=0\wedge b_{r-1}=1]-1/2\\ +\Pr[a_r=0\wedge b_{r-1}=0]+\Pr[a_r=1\wedge b_{r-1}=1]-1/2\end{pmatrix}\right.}{4\rho+1}$$
$$\left.+\Pr[a_{\rho+1}=0\wedge b_\rho=0]+\Pr[a_{\rho+1}=1\wedge b_\rho=1]-1/2\right)$$

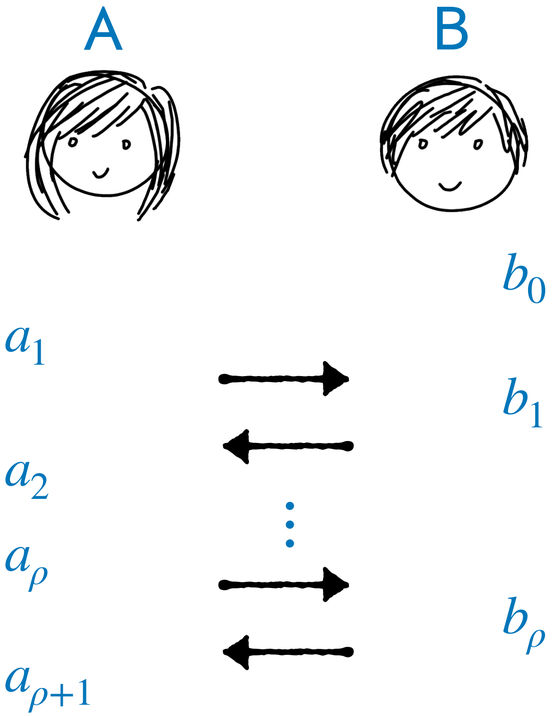


And now we can apply the law of total probability inside the sum *again* to get:

$$\Delta \geq \frac{\left(\Pr[a_{\rho+1}=0\wedge b_\rho=0]+\Pr[a_{\rho+1}=1\wedge b_\rho=1]-1/2\right)}{4\rho+1}$$

Proof: No CT for Dishonest Majority

$$\Delta \geq \frac{\left(\Pr[a_{\rho+1} = 0 \wedge b_\rho = 0] + \Pr[a_{\rho+1} = 1 \wedge b_\rho = 1] - 1/2\right)}{4\rho + 1}$$



Next, observe that *consistency* with all-but-negligible probability implies that there exists some negligible function ε such that

$$\Pr[a_{\rho+1} = 0 \wedge b_\rho = 0] + \Pr[a_{\rho+1} = 1 \wedge b_\rho = 1] \geq 1 - \varepsilon(\kappa)$$

And thus we have

$$\Delta \geq \frac{1/2 - \varepsilon(\kappa)}{4\rho + 1} \in \Omega(1/\rho)$$

Finally, since Δ is the *average* bias induced by our $4\rho + 1$ strategies (i.e. $\hat{\mathsf{A}}$, $\tilde{\mathsf{A}}^{\beta}_{r}$, and $\tilde{\mathsf{B}}^{\beta}_{r}$ for $r \in [\rho]$ and $\beta \in \{0,1\}$), at least one of them must *individually* induce this much bias or more. ∎

Where we Stand

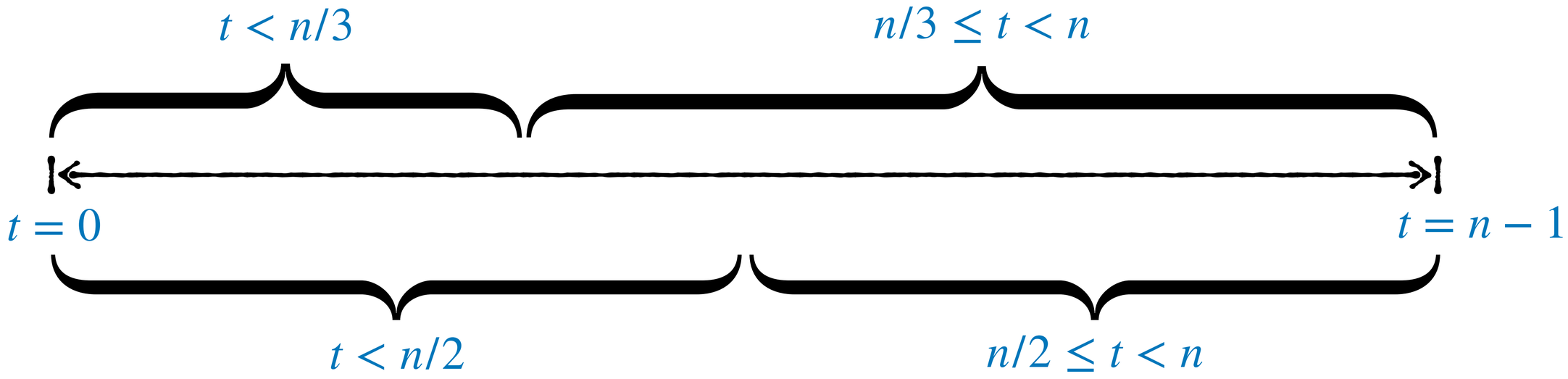
- We have shown that no protocol can achieve the property-based notion of coin-tossing when $t \geq n/2$ parties are corrupt, even using PKI or broadcast.
- This is a strong bound: it rules out any protocol that realizes any functionality that *implies* this coin tossing property.
- The coin tossing functionality obviously implies property-based coin tossing. *Can you think of other functionalities that do?*
- Almost any functionality that samples something uniform and outputs it to everyone. Some important examples:
  - Any functionality that samples a public key and outputs it. E.g. *threshold signing*.
  - Many functionalities that do trusted setup for other protocols.

Where are We? The Landscape of Malicious Coin Tossing

(Let n be the number of parties and t be the number of corruptions)

Perfect Coin Tossing from malicious BGW in the plain model

No non-trivial bias-freeness in the plain model [CHOR 16]



Can we hope for computational coin tossing in the $\mathcal{F}_{\mathsf{BC}}$-hybrid model?

Bias inversely proportional to rounds even assuming PKI (Cleve)

The best we can hope for here is coin tossing with *abort.*

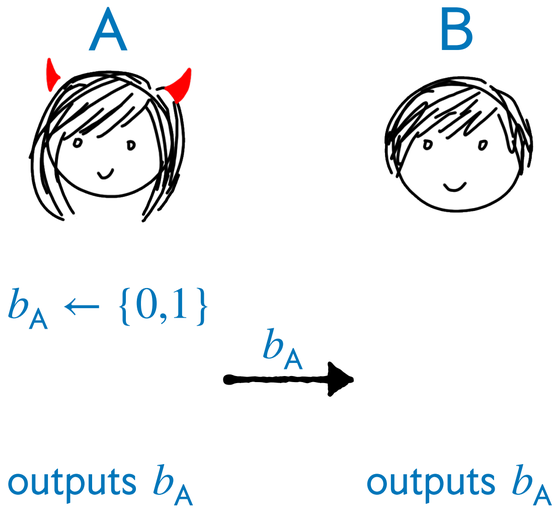
Toward Malicious 2-Party CT with Abort

We'll start by investigating two-party coin tossing with abort, and develop some intuition that will later allow us to handle the cases of arbitrary dishonest majorities with abort, and honest majorities with guaranteed output.

In homework 1, you constructed a simple coin tossing protocol for semi-honest adversaries. Maybe you only needed one message! If Alice is *maliciously* corrupted, there are at least two attacks she can perform. *What are they?*

1. If Alice sends *nothing*, then clearly she is cheating and Bob should *abort* (i.e. output $\perp$).

2. Alice might also sample b_A from the wrong distribution. Bob cannot detect this!

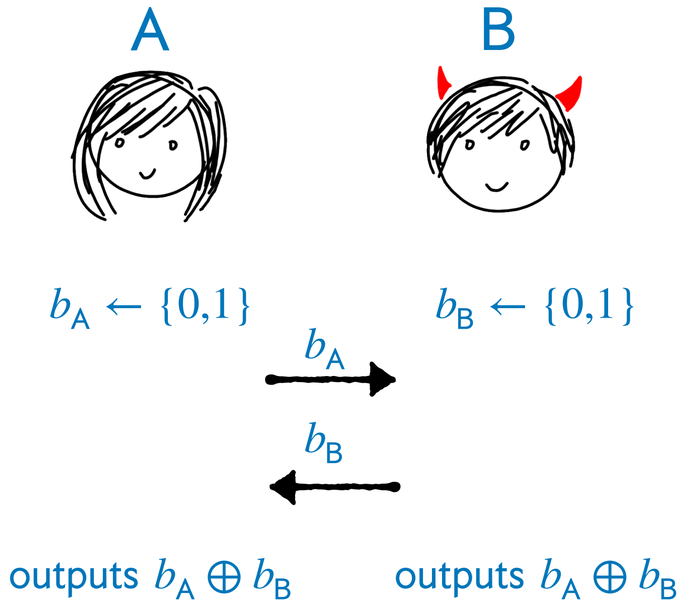


Toward Malicious 2-Party CT with Abort

In homework 1, you constructed a simple coin tossing protocol for semi-honest adversaries. Maybe you only needed one message! If Alice is *maliciously* corrupted, there are at least two attacks she can perform. *What are they?*

1. If Alice sends *nothing*, then clearly she is cheating and Bob should *abort* (i.e. output $\perp$).

2. Alice might also sample b_A from the wrong distribution. Bob cannot detect this!



Perhaps your protocol for homework 1 looked more like *this*. Now the problem is reversed… Bob's output is uniform (or $\perp$) so long as he is honest, but if he is corrupt, then he can choose b_B with knowledge of b_A and thereby fully control the output.

Toward Malicious 2-Party CT with Abort

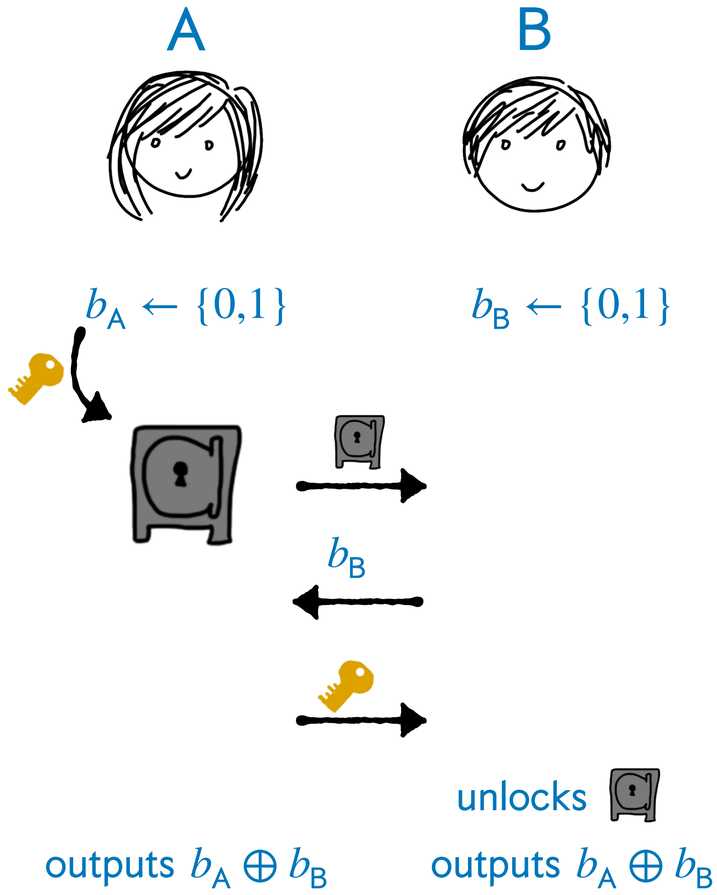
Using a visual metaphor, what is Alice could lock her bit in a secure box, and send Bob the box? While her bit is in the box, he can't see what it is, but she also cannot change it.

After Bob sends his bit, Alice can send him the key, and he can unlock the box in order to compute the output.

This is the essence of Manuel Blum's 1981 protocol for coin tossing over the phone

(pictured with his wife Lenore; later they wrote a paper that argues using complexity theory that AI consciousness is *inevitable*)

Of course, the main problem he resolved is how to construct the box *with cryptography!*

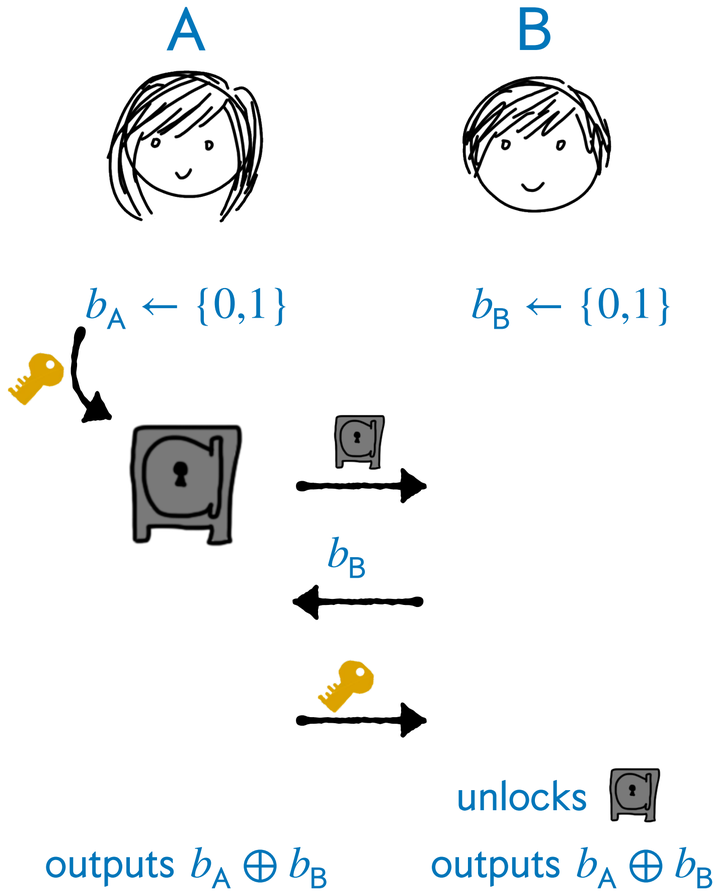


Toward Malicious 2-Party CT with Abort

Before we start formalizing though, let's think ahead. When we generalize this to *many* parties, we need all of them to contribute independent samples. *Should they all put their samples into boxes and exchange them?*

As you discovered in Homework 3 Problem 3, we need to be very careful to ensure that the adversary can't take a box from an honest party and *modify* it. This is a highly non-trivial property! It isn't guaranteed by encryption. *Can you think of another problem with encryption in this context?*

Encryption also doesn't guarantee that there is exactly one key that decrypts a ciphertext. It's perfectly legal for a ciphertext to decrypt to two different messages under two different secret keys! Clearly we need a new primitive…



Commitment Schemes

Definition 4: A Commitment Scheme for message space $\mathcal{M}$ scheme is a trio of algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ such that $k \leftarrow \mathsf{Gen}(1^\kappa)$ samples a key in PPT given a security parameter, $(c, d) \leftarrow \mathsf{Com}_k(m)$ samples a *commitment* c and an *opening* d in PPT given $m \in \mathcal{M}$, and $m' := \mathsf{Open}_k(c, d)$ deterministically opens c using d.

Open_k may also output $\perp$ to indicate that d is invalid for (k, c).

Notice: very much unlike an encryption scheme, there is only one key k, and it's *public*. Whatever privacy properties we specify will have to rely upon keeping something else secret.

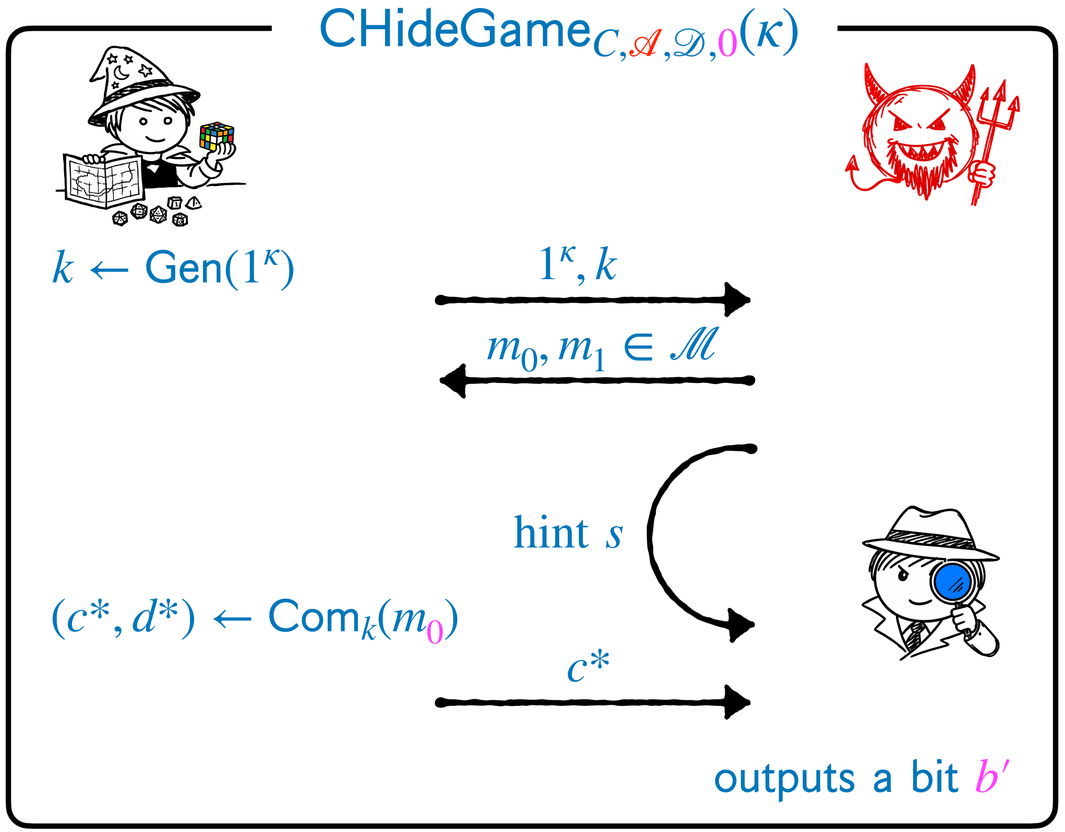
Definition 5: A commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *correct* and *complete* if $\forall m \in \mathcal{M}$, $\forall k \in \mathrm{image}(\mathsf{Gen})$, we have $\mathsf{Open}_k(\mathsf{Com}_k(m)) = m$.

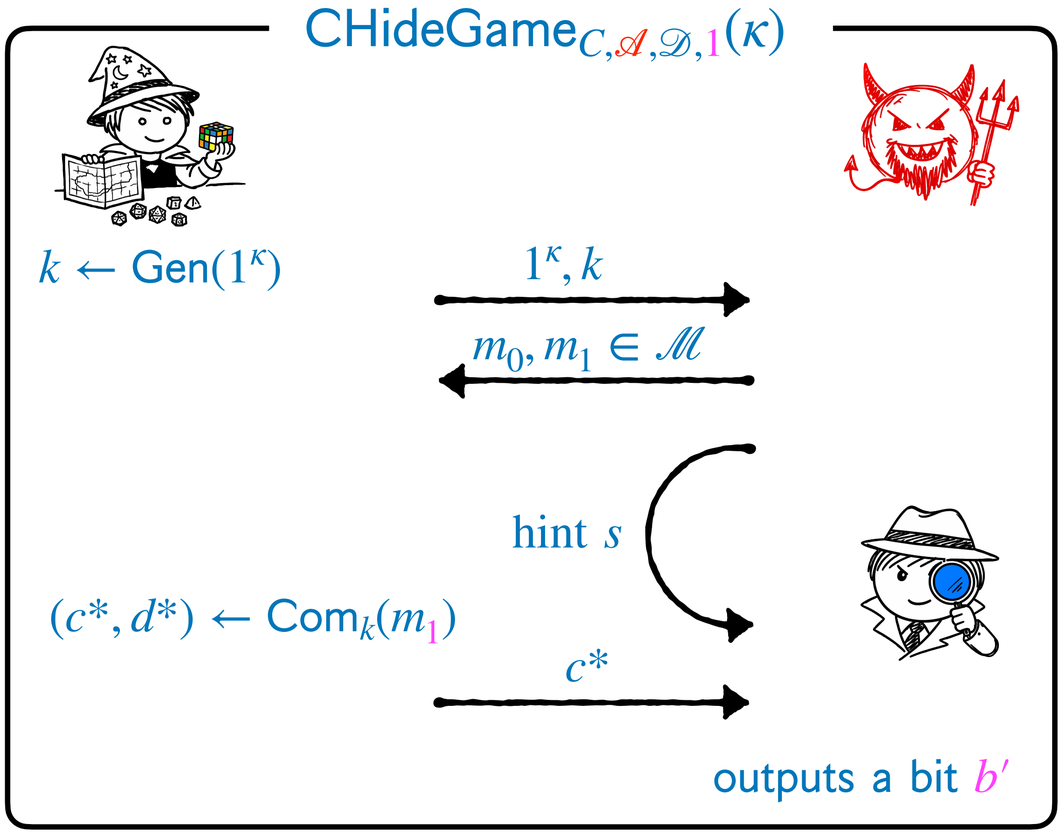
Intuitively, we need two different security properties: the *sender* (Alice in the previous protocol) wants to be sure that her value is *hidden* from Bob until she opens the commitment it.

The *receiver* (Bob) wants to be sure that Alice is *bound* to any value she commits. That is, he wants to be sure that she cannot open a commitment to more than one value.

Def 6: The Commitment Hiding Game

- Let $C = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ be a commitment scheme for message space $\mathscr{M}$.
- The *Commitment Hiding Game* $\mathsf{CHideGame}_{C,\mathscr{A},\mathscr{D},b}(\kappa)$ for $b \in \{0,1\}$ is played between a challenger and an adversary/distinguisher team. The game's output is the distinguisher's.





Def 7: The Commitment Binding Game

- Let $C = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ be a commitment scheme for message space $\mathcal{M}$.
- The *Commitment Binding Game* $\mathsf{CBindGame}_{C,\mathcal{A}}(\kappa)$ for is played between a challenger and an adversary. The game's output is the output of the challenger.
- Notice that the adversary wins by finding *any collision* rather than a second preimage. This means that the binding property for commitments is substantially stronger than security property we defined for UOWHFs that we defined in Lecture 21!

$\mathsf{CBindGame}_{C,\mathcal{A}}(\kappa)$

$k \leftarrow \mathsf{Gen}(1^\kappa)$

$1^\kappa, k$ →

← c, d, d'

$m := \mathsf{Open}_k(c, d)$

$m' := \mathsf{Open}_k(c, d')$

outputs 1 if :

$\bot \neq m \neq m' \neq \bot$

outputs 0 otherwise

Commitment Security Definition

Definition 8: A Commitment Scheme is *Perfectly Hiding* if, $\forall\ \mathscr{A}, \mathscr{D}$,

$$\left|\Pr[\mathsf{CHideGame}_{C,\mathscr{A},\mathscr{D},0}(\kappa) = 1] - \Pr[\mathsf{CHideGame}_{C,\mathscr{A},\mathscr{D},1}(\kappa) = 1]\right| = 0.$$

It is instead *Computationally Hiding* if $\forall$ PPT $\mathscr{A},\mathscr{D}$ there exists some negligible ε such that

$$\left|\Pr[\mathsf{CHideGame}_{C,\mathscr{A},\mathscr{D},0}(\kappa) = 1] - \Pr[\mathsf{CHideGame}_{C,\mathscr{A},\mathscr{D},1}(\kappa) = 1]\right| \leq \varepsilon(\kappa).$$

Definition 9: A Commitment Scheme is *Perfectly Binding* if, $\forall\mathscr{A}$, $\Pr[\mathsf{CBindGame}_{C,\mathscr{A}}(\kappa) = 1] = 0$.

It is instead *Computationally Binding* if $\forall$ PPT $\mathscr{A}$ there exists some negligible ε such that $\Pr[\mathsf{CBindGame}_{C,\mathscr{A}}(\kappa) = 1] \leq \varepsilon(\kappa)$.

Q: *Intuitively, can a commitment scheme be both perfectly hiding and perfectly binding?*

Q: *But who samples the key? You have one security property for each party, and both games specify that the challenger should sample!*

A: The way we've defined it, neither party can! They need a trusted helper. However, many schemes actually tolerate one of the two sampling the key (including the one we'll see next).

CS4501 Cryptographic Protocols
Lecture 23: Coin Tossing, Cleve's Bound, Commitments, Blum's Protocol

https://jackdoerner.net/teaching/#2026/Spring/CS4501