CS4501 Cryptographic Protocols
Lecture 24: Commitments, Intro to Zero Knowledge

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

Commitment Schemes

Definition 1: 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.

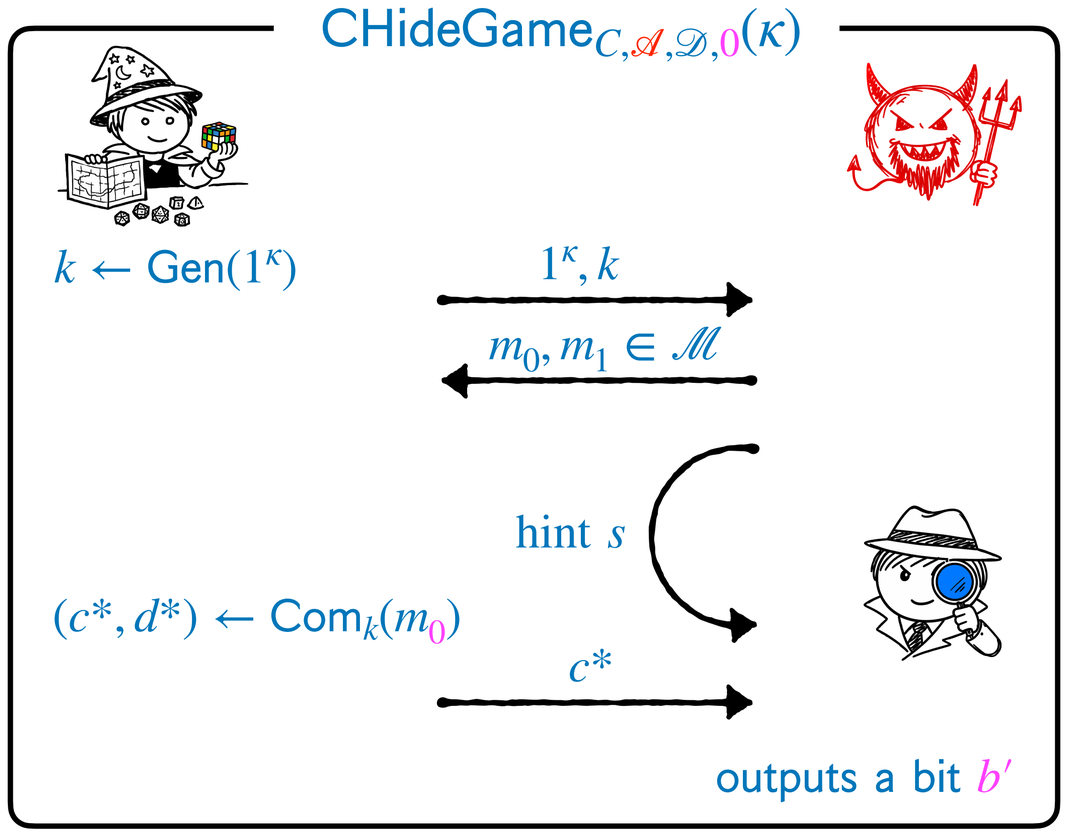
Definition 2: A commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *correct* and *complete* if $\forall m \in \mathcal{M}$, $\forall k \in \mathrm{image}(\mathsf{Gen})$, we have $\mathsf{Open}_k(\mathsf{Com}_k(m)) = m$.

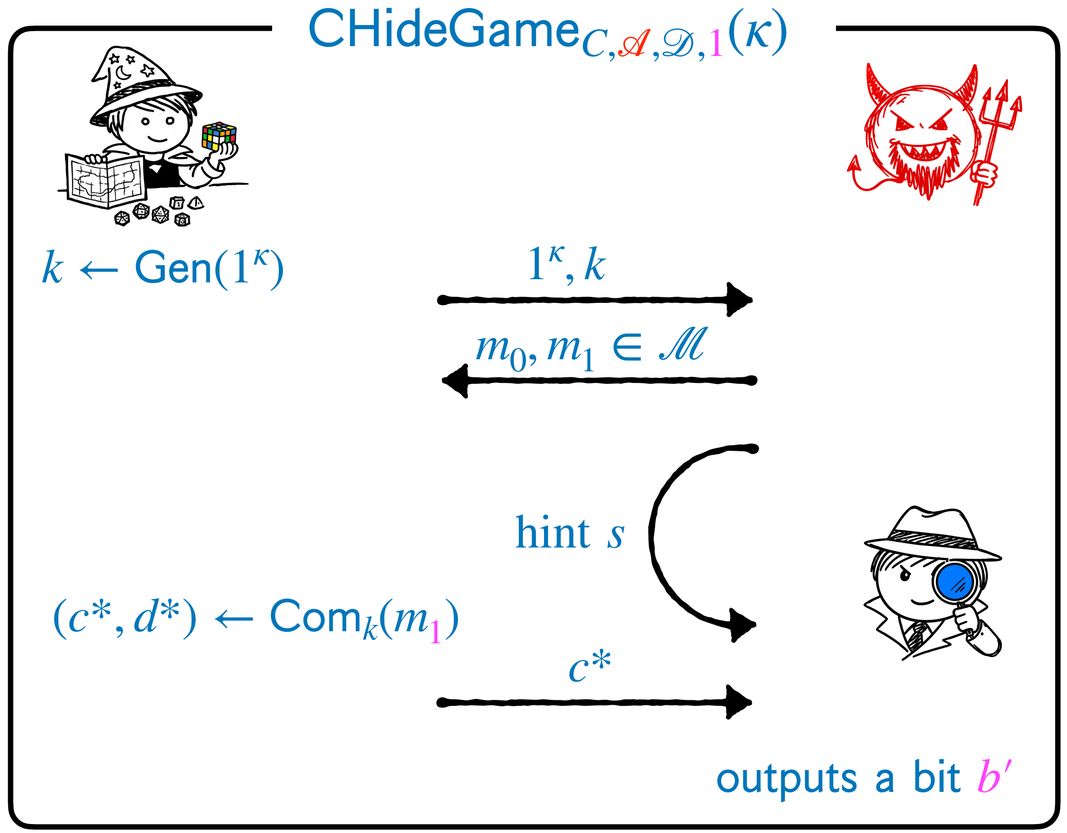
Intuitively, we need two different security properties: the *sender* (Alice in the previous protocol) wants to be sure that her value is *hidden* from Bob until she opens the commitment it.

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 3: 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 4: 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 5: 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 6: 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).

Pedersen Commitments

Definition 7 (Pedersen Commitment for $\mathcal{M} = [0,\ell]$):
Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q > \ell$, $|q| \geq \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). The *Pedersen Commitment Scheme* comprises the following algorithms:

- $\mathsf{Gen}(1^\kappa)$ outputs $k := (\mathbb{G}, g, q, h)$ where $(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$, $x \leftarrow \mathbb{Z}_q$, and $h := g^x$.
- $\mathsf{Com}_k(m)$ outputs (C, d) where $r \leftarrow \mathbb{Z}_q$, $C := g^m \cdot h^r$, and $d := (m, r)$.
- $\mathsf{Open}_k(C, (m, r))$ outputs m if and only if $g^m \cdot h^r = C$.

Theorem 1: If the Discrete Logarithm Problem is hard relative to $\mathcal{G}$, then the Pedersen commitment scheme is *perfectly* hiding and *computationally* binding.

Recall: The Discrete Logarithm Assumption

Definition 8 (Hardness for DLog): The Discrete Logarithm Problem is *hard* relative to $\mathcal{G}$ if for every PPT adversary $\mathcal{A}$, there exists some negligible function ε such that

$$\Pr\left[\mathsf{DLGame}_{\mathcal{G},\mathcal{A}}(\kappa) = 1\right] \leq \varepsilon(\kappa)$$

Definition 9 (DLog Assumption): *The Discrete Logarithm Assumption* asserts that there exists some PPT algorithm $\mathcal{G}$ relative to which the discrete logarithm problem is hard.

$\mathsf{DLGame}_{\mathcal{G},\mathcal{A}}(\kappa)$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$h \leftarrow \mathbb{G}$

$(\mathbb{G}, g, q, h)$ →

← $x \in \mathbb{Z}_q$

outputs 1 if $g^x = h$
outputs 0 otherwise

Pedersen Commitments

Proof: First, consider *hiding*. For any fixed m and h (even *adversarially chosen* ones), the commitment C is uniformly distributed in $\mathbb{G}$ because $r \leftarrow \mathbb{Z}_q$. Thus the distinguisher sees *exactly* the same distribution in the two hiding games.

Next, consider *binding*. Suppose toward contradiction that there exists some PPT adversary $\mathscr{A}$ that contradicts the binding property. That is, $\mathscr{A}$ has a *non-negligible* probability of producing $(C, (m, r), (m', r'))$ such that $\bot \neq \mathsf{Open}_k(C, (m, r)) \neq \mathsf{Open}_k(C, (m', r')) \neq \bot$ and $m \neq m'$.

Notice that when $\mathscr{A}$ succeeds, we have $g^m \cdot h^r = g^{m'} \cdot h^{r'}$ which implies that $g^m \cdot g^{x \cdot r} = g^{m'} \cdot g^{x \cdot r'}$, which implies that $m - m' = x \cdot (r' - r)$.

Since $m - m' \neq 0$, it must be that $r - r' \neq 0$, which finally yields $x = (m - m')/(r' - r)$.

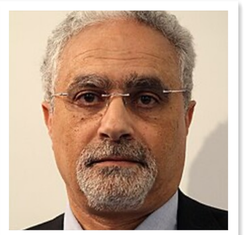
In other words, $\mathscr{A}$ has implicitly recovered the discrete logarithm x of h!

We can construct a second adversary $\mathscr{A}'$ for $\mathsf{DLGame}_{\mathscr{G},\mathscr{A}'}(\kappa)$, which emulates $\mathsf{CBindGame}_{C,\mathscr{A}}(\kappa)$ internally, but sets $h := X$, where X is the challenge $\mathscr{A}'$ receives in $\mathsf{DLGame}_{\mathscr{G},\mathscr{A}'}(\kappa)$.

If $\mathscr{A}$ succeeds, then uses the above method to recover x such that $g^x = X$, with no loss of advantage.

This contradicts our assumption that the Discrete Logarithm Problem is hard relative to $\mathscr{G}$. ∎

ElGamal Commitments



Definition 10 (ElGamal Commitment for $\mathcal{M} = [0,\ell]$):
Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q > \ell$, $|q| \geq \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). The *ElGamal Commitment Scheme* comprises the following algorithms:

- $\mathsf{Gen}(1^\kappa)$ outputs $k := (\mathbb{G}, g, q, h)$ where $(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$, $x \leftarrow \mathbb{Z}_q$, and $h := g^x$.
- $\mathsf{Com}_k(m)$ outputs (c, d) where $r \leftarrow \mathbb{Z}_q$, $c := (g^r, h^r \cdot g^m)$, and $d := (m, r)$.
- $\mathsf{Open}_k((C_1, C_2), (m, r))$ outputs m iff $C_1 = g^r \wedge C_2 = h^r \cdot g^m$.

Notice: this is effectively just ElGamal encryption! But we argued that encryption isn't binding *generally*. Specific encryption schemes often have extra properties that the definition neither forbids nor mandates. Sometimes we don't recognize those properties…

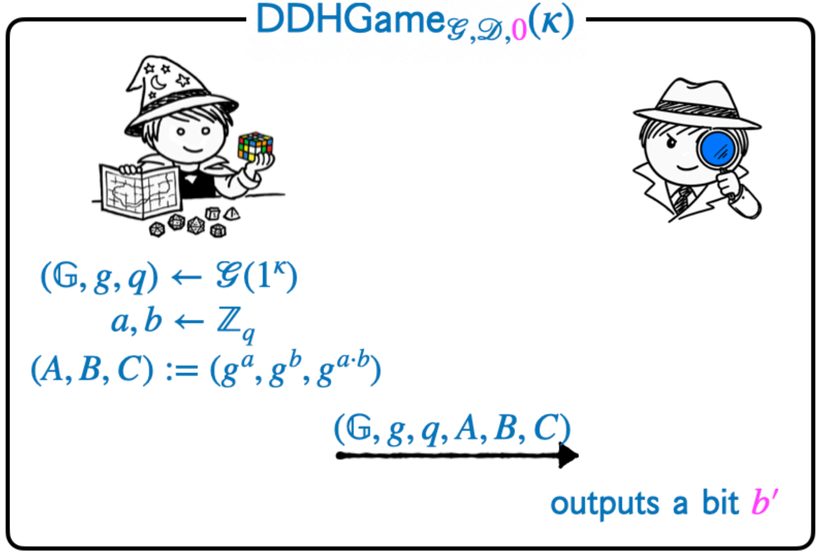
Theorem 2: If the Decisional Diffie-Hellman Problem is hard relative to $\mathcal{G}$, then the Elgamal commitment scheme is *computationally* hiding and *perfectly* binding.

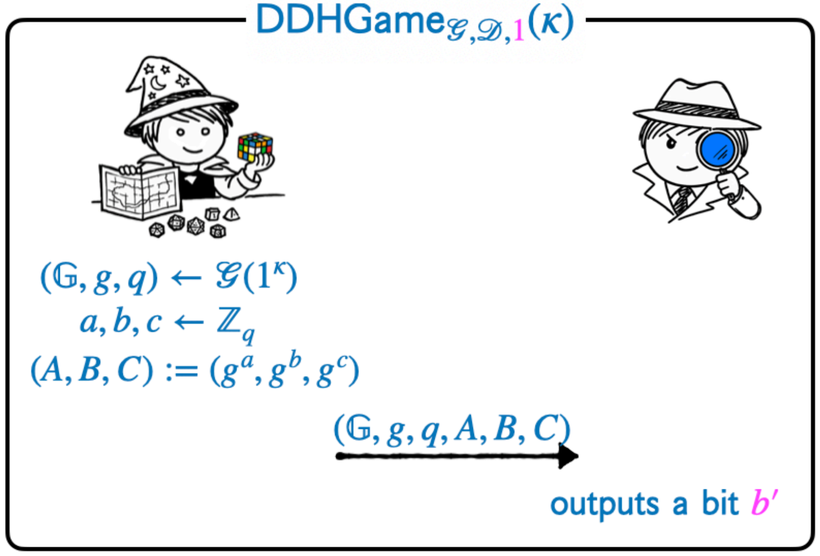
Recall: Decisional Diffie-Hellman

Definition 11 (DDH Hardness): The Decisional Diffie-Hellman Problem is *hard* relative to $\mathcal{G}$ if for every PPT distinguisher $\mathcal{D}$, there exists some negligible function ε such that

$$\left| \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},0}(\kappa) = 1\right] - \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},1}(\kappa) = 1\right] \right| \leq \varepsilon(\kappa)$$

Definition 12 (DDH Assumption): *The Decisional Diffie-Hellman Assumption* asserts that there exists a PPT algorithm $\mathcal{G}$ relative to which the DDH problem is hard.





ElGamal Commitments

Proof Sketch: Consider *binding*. $c_1 = g^r$ and h uniquely fix $m = c_3 \cdot h^{-r}$. Thus binding is perfect, even if h is adversarially chosen.

Next, consider *hiding*. The proof of this property is nearly identical to the proof of IND-CPA security for ElGamal encryption, so we will only sketch it informally…

Given any DDH tuple $(X = g^x, Y = g^y, Z)$ such that either $Z = g^z$ or $Z = g^{x \cdot y}$, one can convert such a tuple into an *either* an ElGamal commitment to $m \in \mathbb{Z}_q$ *or* a commitment-shaped trio of uniform group elements by setting the key to be $k = (\mathbb{G}, g, q, Y)$ and the commitment itself to be $c = (X, Z \cdot g^m)$.

We can construct a hybrid experiment by starting from one of the commitment hiding games for ElGamal, and replacing the ElGamal commitment with a commitment-shaped trio of uniform group elements. Using the idea from the last paragraph, we can prove by reduction to the DDH game that this hybrid must be indistinguishable from *both* commitment hiding games if the DDH problem is hard relative to $\mathcal{G}$.

Thus, by the transitivity of computational indistinguishability, ElGamal commitments are computationally hiding. ∎

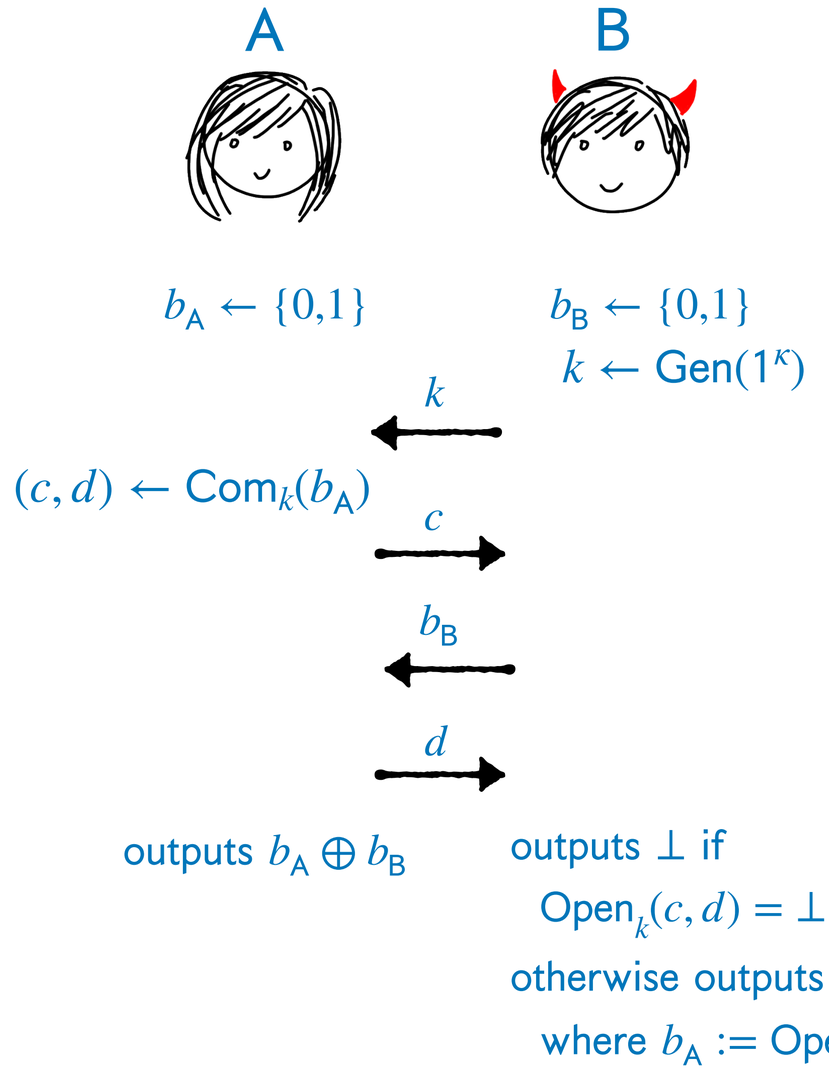
Toward Malicious 2-Party CT with Abort

Now let's look back at Blum's Protocol. Suppose that we use Pedersen's commitment scheme for our "secure box".

Pedersen commitments are hiding even if the receiver samples $(\mathbb{G}, g, q, h) = k \leftarrow \mathsf{Gen}(1^\kappa)$, assuming that the sender verifies that $(\mathbb{G}, g, q)$ really is a prime-order group that contains h. Thus Bob will sample and send k at the start.

Since Pedersen commitments are perfectly hiding, it's intuitive to see that we achieve (perfect) bias-freeness when Bob is corrupted: his view does not contain any information about b_A until after he sends b_B. Alice outputs $\perp$ only if he blindly fails to send a message.

What about when Alice is corrupted? *Any thoughts?*



Toward Malicious 2-Party CT with Abort

If we let y_B be Bob's output, we said informally that we wanted *either* $y_B = b$ where $b \leftarrow \{0,1\}$ is uniform, *or* $y_B = \perp$. $\mathcal{A}$ gets to choose between these two outcomes, but it cannot force $y_B = 1 - b$.

Can you define what b is? Where in the experiment does this magical uniform reference value live?

Maybe we can make sense of it by defining a functionality!

When $\mathcal{F}_{\text{CT-Abort}}$ is invoked by A and B, it samples $b \leftarrow \{0,1\}$ and sends b directly to $\mathcal{S}$. If $\mathcal{S}$ replies "OK", then $\mathcal{F}_{\text{CT-Abort}}$ sends b to A and B. If $\mathcal{S}$ replies "Abort", then $\mathcal{F}_{\text{CT-Abort}}$ sends $\perp$ to A and B.

So, can we write an appropriate simulator $\mathcal{S}$?

$b_A \leftarrow \{0,1\}$

$b_B \leftarrow \{0,1\}$
$k \leftarrow \text{Gen}(1^\kappa)$

k

$(c, d) \leftarrow \text{Com}_k(b_A)$

c

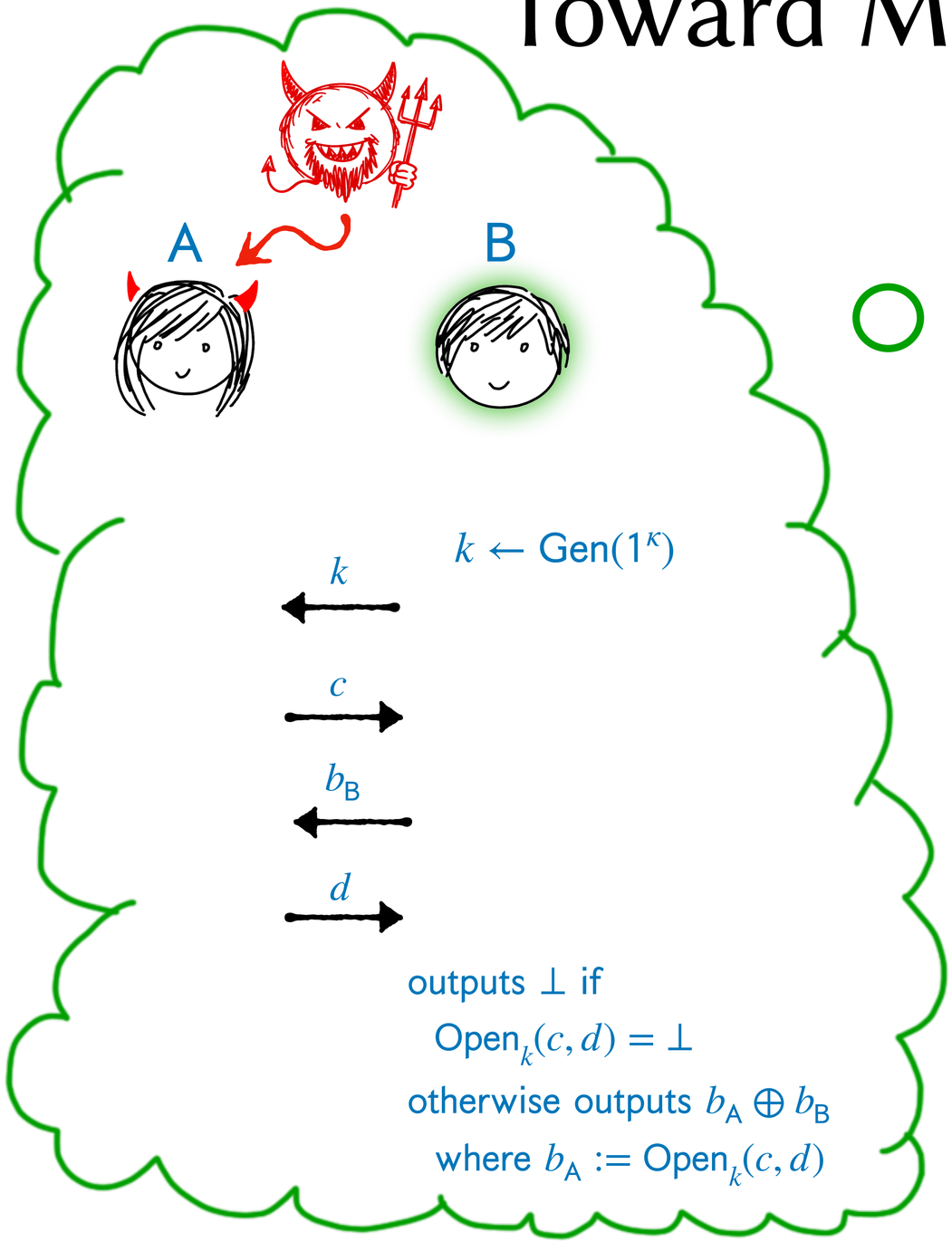
b_B

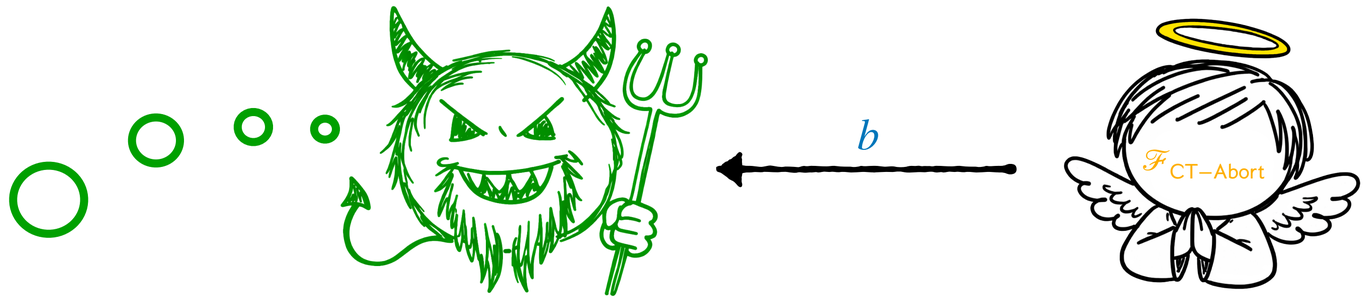
d

outputs $b_A \oplus b_B$

outputs $\perp$ if
$\text{Open}_k(c, d) = \perp$
otherwise outputs $b_A \oplus b_B$
where $b_A := \text{Open}_k(c, d)$

Toward Malicious 2-Party CT with Abort





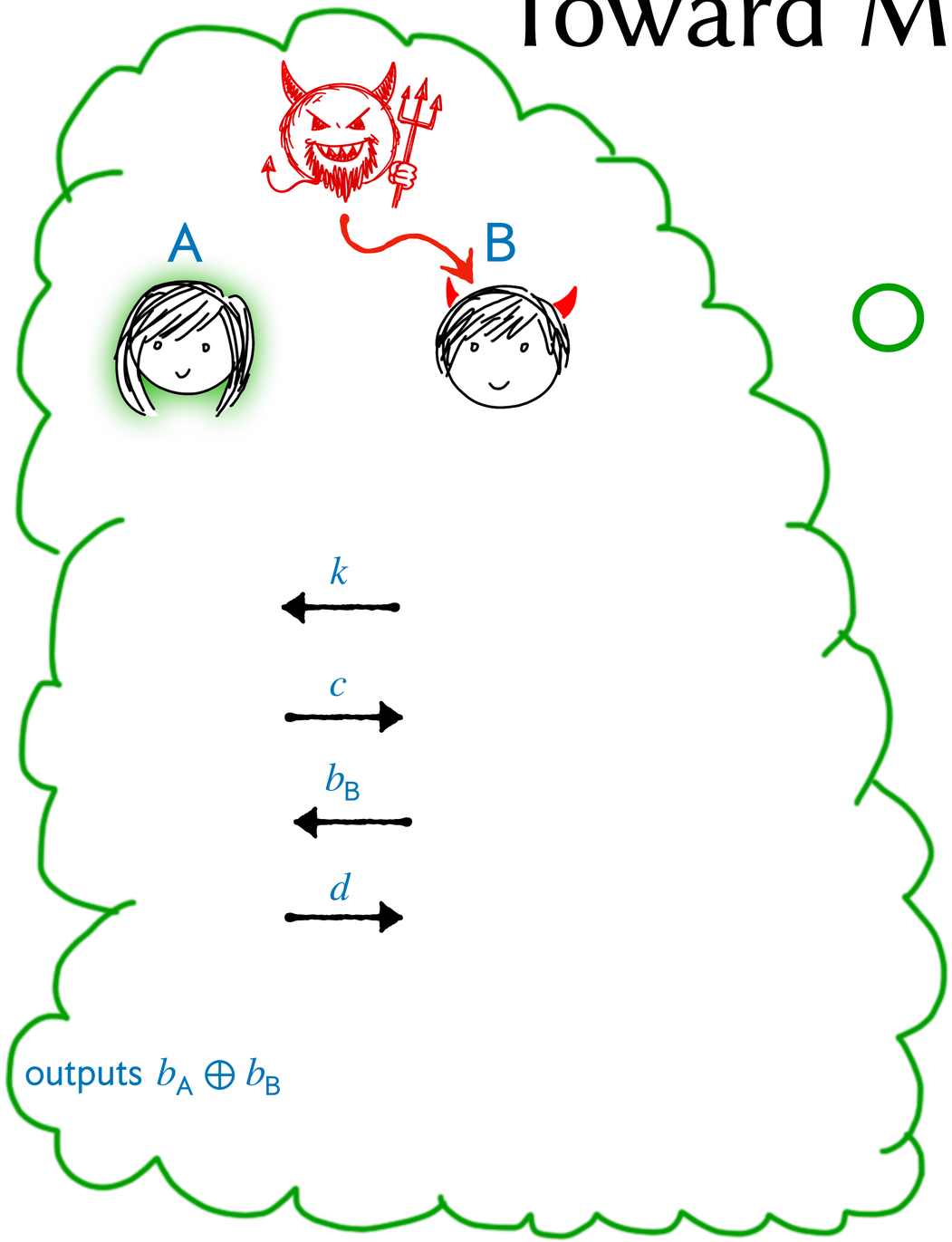
$\mathcal{S}$ needs to choose b_B such that $b_\mathsf{A} \oplus b_\mathsf{B} = b$ if $\mathsf{Open}_k(c,d) \neq \perp$.

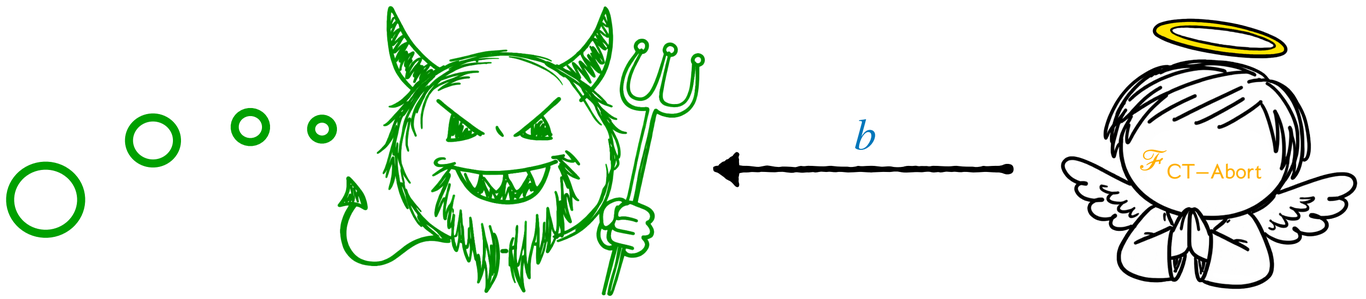
But how? The hiding property of c prevents $\mathcal{S}$ from learning b_A until after it sends b_B.

Or does it? Remember, $\mathcal{A}$ and A are *subroutines* of $\mathcal{S}$. So the information is in there *somewhere*...

How can $\mathcal{S}$ extract it without knowing how $\mathcal{A}$ works?

Toward Malicious 2-Party CT with Abort





What if Bob is corrupt? We had an intuition that the protocol is bias-free, but that *doesn't* imply simulatability.

Once again, $\mathcal{S}$ needs to choose b_A such that $b_A \oplus b_B = b$.

But how? b_A is committed using c, and the binding property prevents $\mathcal{S}$ from changing b_A after it learns b_B.

Or does it? Remember, $\mathcal{A}$ and B are *subroutines* of $\mathcal{S}$. $\mathcal{S}$ can alter their views if it wants to!

But $\mathcal{A}$ could do something tricky like make b_B depend on c. How can $\mathcal{S}$ change b_A without knowing how $\mathcal{A}$ works?

Hold onto these questions… we will come back to them :)

Commitments with Special Powers

We have identified two different ways in which we would like to give $\mathcal{S}$ extra power when we use a commitment scheme, while denying that power to $\mathcal{A}$ in the real world.

- We want to be able to learn a value committed by $\mathcal{A}$ at the moment it is committed, and before it is opened. We will call this power *extraction.*
- We want to be able to change a value committed by $\mathcal{S}$ at the moment the commitment is opened to $\mathcal{A}$. We will call this power *equivocation.*

Both of these special powers can be meaningfully formulated for other primitives as well, but they are clearest in the context of commitments - and most obviously in contradiction to the standard properties of the primitive.

In order to grant these powers to the simulator and not the adversary, we will design commitment schemes that are *dual-mode.* There will be two different ways to generate the keys for our commitments. In one mode, the commitment scheme behaves like we have described. In the other mode, there is a secret *trapdoor* that can be used to extract or equivocate.

Of course, it's important that the keys for the two modes must be indistinguishable!

Commitments with Special Powers

Definition 13: A Commitment Scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ for message space $\mathscr{M}$ *extractable* if it includes a pair of extra PPT algorithms, $(\mathsf{TGen}, \mathsf{Ext})$, such that:

- TGen generates *trapdoored* keys that are indistinguishable from ordinary ones. That is, $\{k : (k,t) \leftarrow \mathsf{TGen}(1^\kappa)\}_{\kappa\in\mathbb{N}} \approx_c \{\mathsf{Gen}(1^\kappa)\}_{\kappa\in\mathbb{N}}$
- If $(k,t) \leftarrow \mathsf{TGen}(1^\kappa)$ and $(c,d) \leftarrow \mathsf{Com}_k(m)$ for some $m \in \mathscr{M}$, then $\mathsf{Ext}_t(c) = m$.

Theorem 3: If $\mathscr{M} = [0,\ell]$ for $\ell \in O(\mathrm{poly}(\kappa))$, then ElGamal Commitments for $\mathscr{M}$ are *extractable*.

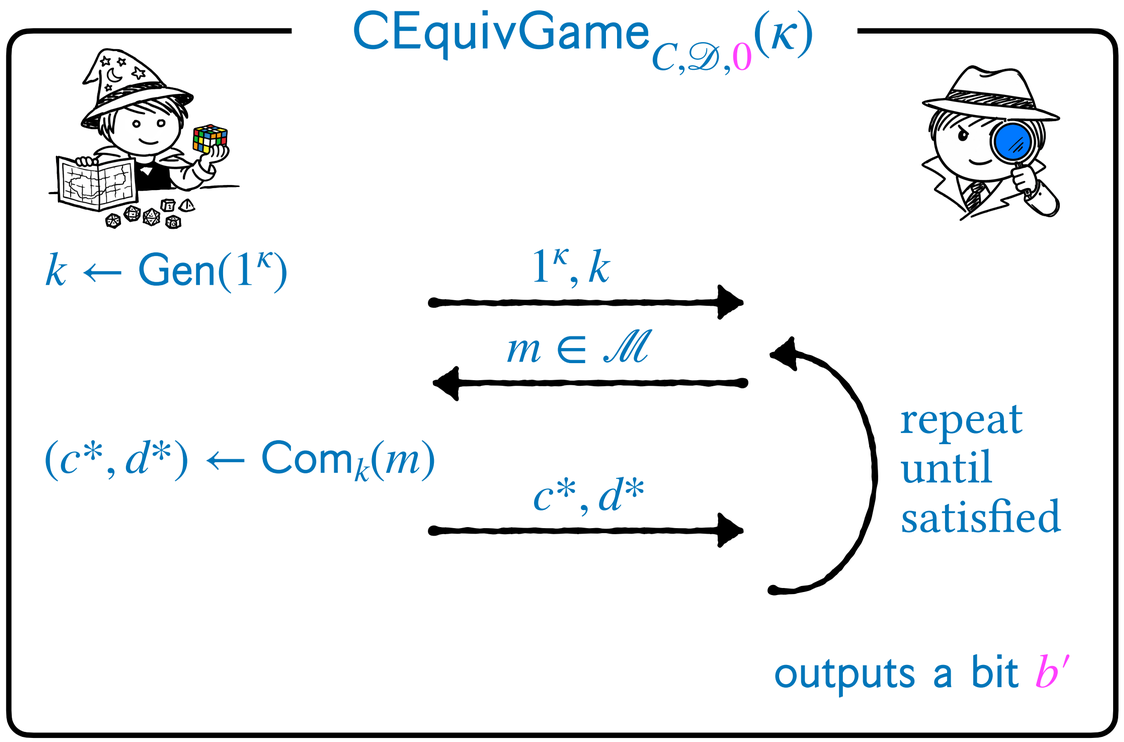
Proof Sketch:

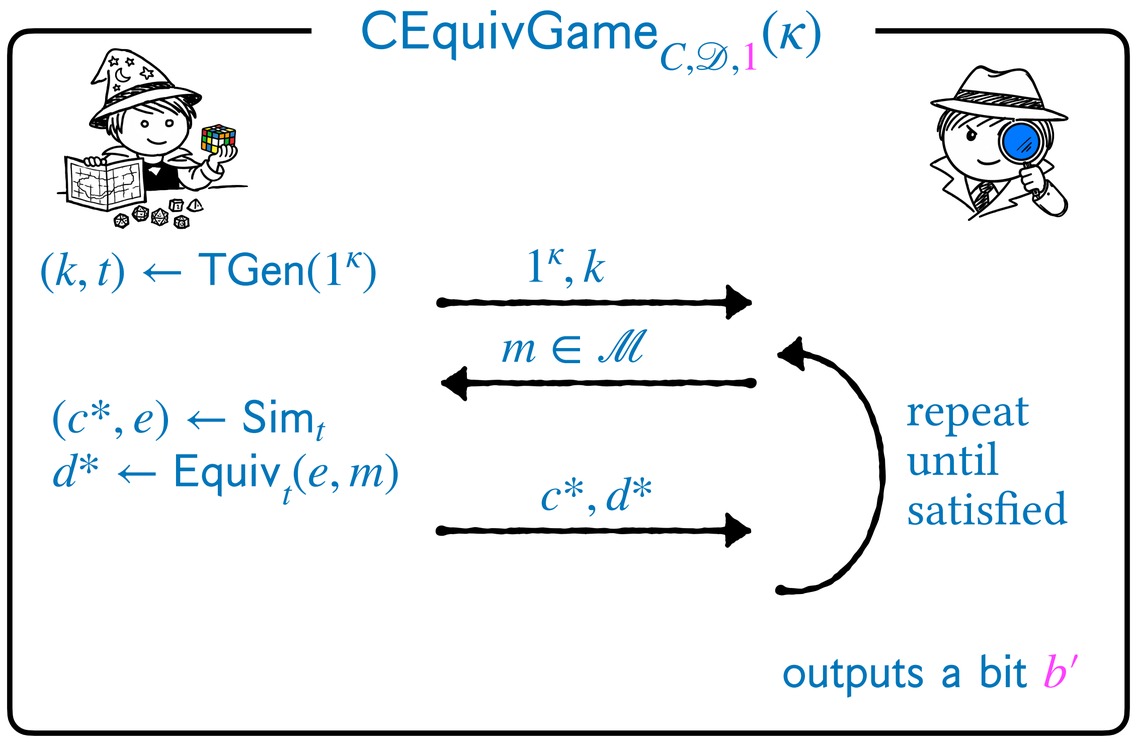
- $\mathsf{TGen}(1^\kappa)$ outputs (k,t) where $(\mathbb{G},g,q) \leftarrow \mathscr{G}(1^\kappa)$, $k := (\mathbb{G},g,q,h)$, $t \leftarrow \mathbb{Z}_q$, and $h := g^t$.
- Recall that an ElGamal Commitment is $(C_1,C_2) = (g^r, h^r \cdot g^m)$, for some $(m,r) \in \mathscr{M} \times \mathbb{Z}_q$.
- $\mathsf{Ext}_t((C_1,C_2))$ uses a brute force search to find and output $m' \in \mathscr{M}$ such that $g^{m'} \cdot C_1^t = C_2$. (this is just like decrypting an ElGamal encryption of m). ∎

Commitments with Special Powers

Definition 14: A Commitment Scheme $C = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ for message space $\mathscr{M}$ *equivocable* if it includes a trio of extra PPT algorithms, $(\mathsf{TGen}, \mathsf{Sim}, \mathsf{Equiv})$, such that:

- If $(k, t) \leftarrow \mathsf{TGen}(1^\kappa)$, $(c, e) \leftarrow \mathsf{Sim}_t$, and $d \leftarrow \mathsf{Equiv}_t(e, m)$ for $m \in \mathscr{M}$, then $\mathsf{Open}_k(c, d) = m$.
- $\forall$ PPT $\mathscr{D}$, $\exists$ negl. ε s.t. $\left|\Pr[\mathsf{CEquivGame}_{C,\mathscr{D},0}(\kappa) = 1] - \Pr[\mathsf{CEquivGame}_{C,\mathscr{D},1}(\kappa) = 1]\right| \leq \varepsilon(\kappa)$.





Commitments with Special Powers

Definition 14: A Commitment Scheme $C = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ for message space $\mathscr{M}$ *equivocable* if it includes a trio of extra PPT algorithms, $(\mathsf{TGen}, \mathsf{Sim}, \mathsf{Equiv})$, such that:

- If $(k, t) \leftarrow \mathsf{TGen}(1^\kappa)$, $(c, e) \leftarrow \mathsf{Sim}_t$, and $d \leftarrow \mathsf{Equiv}_t(e, m)$ for $m \in \mathscr{M}$, then $\mathsf{Open}_k(c, d) = m$.
- $\forall$ PPT $\mathscr{D}$, $\exists$ negl. ε s.t. $\left|\Pr[\mathsf{CEquivGame}_{C,\mathscr{D},0}(\kappa) = 1] - \Pr[\mathsf{CEquivGame}_{C,\mathscr{D},1}(\kappa) = 1]\right| \leq \varepsilon(\kappa)$.

Theorem 4: Pedersen Commitments for are *equivocable.*

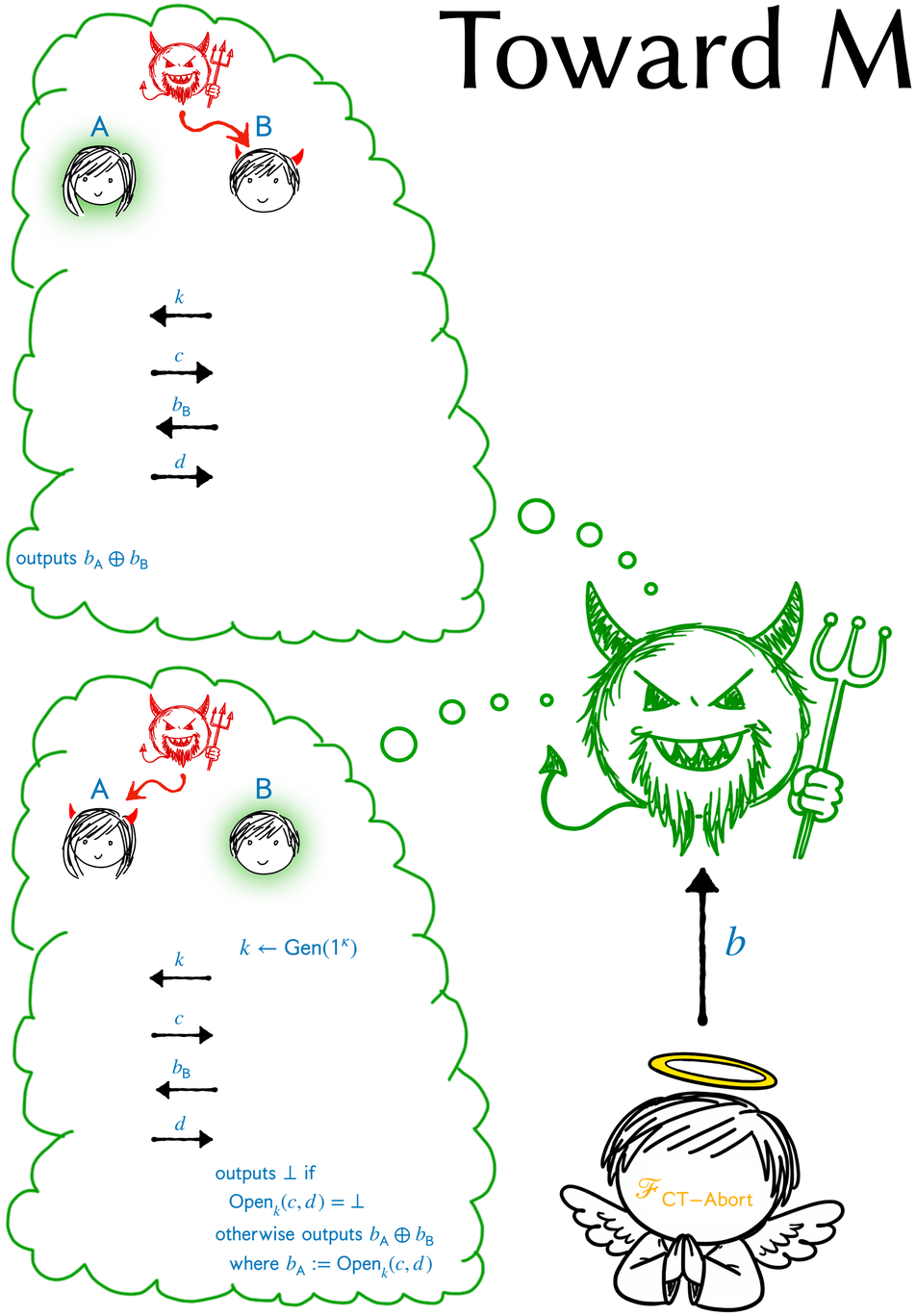
Proof Sketch:

- $\mathsf{TGen}(1^\kappa)$ outputs (k, t) where $(\mathbb{G}, g, q) \leftarrow \mathscr{G}(1^\kappa)$, $k := (\mathbb{G}, g, q, h)$, $t \leftarrow \mathbb{Z}_q$, and $h := g^t$.
- Recall that a Pedersen Commitment is $C = g^m \cdot h^r$, for some $(m, r) \in \mathscr{M} \times \mathbb{Z}_q$.
- Sim_t outputs $e \leftarrow \mathbb{Z}_q$ and $C := g^e$.
- $\mathsf{Equiv}_t(e, m)$ outputs $d := (m, r)$ where $r := (e - m)/t$. ∎

The Common Reference String (CRS) Model

- Now we have to revisit the question, *who samples the commitment key?*
- Nobody should be allowed to equivocate or extract in the real world, but $\mathcal{S}$ needs to have the power to do this when it's emulating an experiment toward $\mathcal{A}$. The trapdoor is like a secret key that nobody is allowed to know!
- We will introduce a new functionality, $\mathcal{F}_{\mathsf{CRS}}$ which is responsible for sampling *common reference strings* that need to be trapdoor-free in the real world.
- $\mathcal{F}_{\mathsf{CRS}}$ runs once at the beginning of the experiment. It runs the commitment scheme's $k \leftarrow \mathsf{Gen}(1^\kappa)$ function internally, and sends the resulting non-trapdoored k to everyone.
- In the ideal world, $\mathcal{S}$ must simulate $\mathcal{F}_{\mathsf{CRS}}$ toward $\mathcal{A}$, and it can use TGen instead of Gen!
- $\mathcal{F}_{\mathsf{CRS}}$ is sort of like fancy version of the coin-tossing functionality $\mathcal{F}_{\mathsf{CT}}$ (with a possibly non-uniform output distribution). Much like the PKI model assumes that we get one broadcast for free at the beginning of time, the CRS model assumes that we get one coin-toss for free.

Toward Malicious 2-Party CT with Abort



So now we have the tools to simulate the two parties. We still have a problem. *What is it?*

We need a different commitment scheme depending on which party we're simulating!

So we need a single commitment scheme that's equivocable *and* extractable.

We have seen enough pieces to construct such a commitment scheme! There are no more homeworks, but you can try constructing this scheme just for the sake of the challenge.

If you want a hint, look at the OT protocol of Peikert, Vaikuntanathan, and Waters [PVW 08].

Remember Blum's Coin Tossing protocol and the problem of how to simulate it. We will return in a few lectures to solve it, after introducing our final big idea...

The Three Main Characters of this Class

1. Shamir Secret Sharing 
2. Oblivious Transfer 
3. Zero-Knowledge Proofs 

In this class I have tried to identify where each of the big ideas that we use originally came from.

Zero-knowledge is important not just as a technique, but because it is the origin point of the idea of *simulation-based* security.

As we learn about ZK, try to remember how strange and counterintuitive the idea of simulation was when you first encountered it at the beginning of the semester.

What is A Proof?

The way you usually experience it in mathematics, a proof starts from agreed axioms and applies some sequence of agreed derivation rules to reach a convincing conclusion ("the statement").

So I can stand up here and prove something, but it's not meaningful unless somebody *verifies* the proof. That's your job! I might make a mistake or tell a lie, and your job is to *not be convinced* if I do.

My job as a *prover* is to convince you. Your job as a *verifier* is to doubt me and check everything I say.

What is Knowledge?

Cryptography takes a *behaviorist* view of this question. Knowledge enables you to *do* something, and in particular, we say that you *gain knowledge* if and only if *you can do something new*. If you can achieve something you couldn't before.

So if you go home after a class today and you can't achieve anything new, then you haven't gained any knowledge.

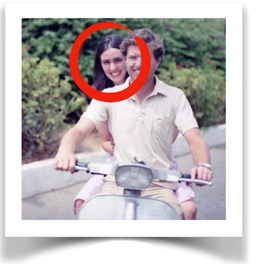
Claim: Zero-Knowledge Proofs

If you can prove a statement in such a way that a *computer* can verify the proof, then you can prove that statement without *conveying any knowledge*, beyond the fact that it is true.

In particular, the verifier of the proof cannot even prove the statement to somebody else afterwards!

Put more simply: you can prove that something is true without revealing *why* it is true.

Zero-Knowledge Proofs


Zero Knowledge Proofs were introduced by Goldwasser, Micali, and Rackoff in 1986. Their paper was rejected 3 times, allegedly because reviewers thought that the claim was impossible.

Goldwasser and Micali later received a Turing Award for this work (and for the modern security definitions of encryption and signatures, and so many other foundational ideas it would be hard to name them all).

Although ZKPs are just one kind of protocol, they're a very special and fundamental kind with a very rich body of literature and important connections to other areas of theoretical CS. We could teach an entire course on zero-knowledge, and barely scratch the surface.

Goldreich spends much of Part 1 of his foundations textbook on ZKPs. He doesn't get to encryption until Part 2.

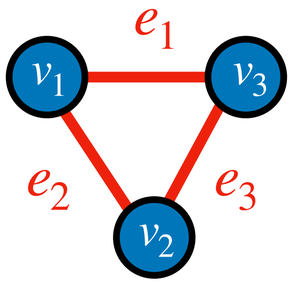
Claim: Zero-Knowledge Proofs

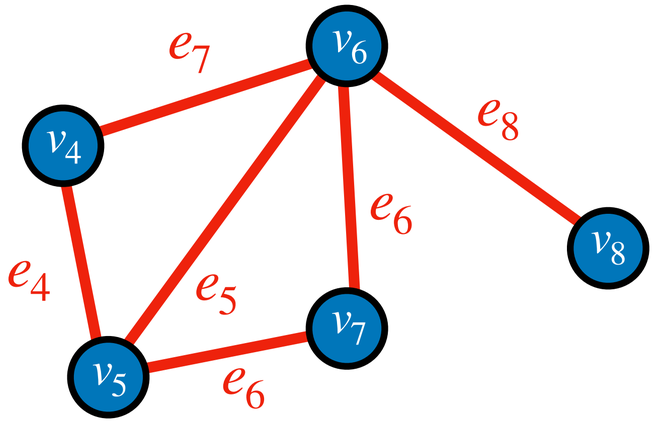
I hope to convince you by the end of the day that anything that can be proven to a computer can be proven in zero-knowledge.

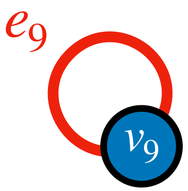
Before we talk about proving general statements to a computer, let's think about proofs for a single, simple class of problems.

What is a Graph?

- A Graph is a collection of *Vertices*, connected by *Edges*. Every edge connects exactly two vertices, and vertices may have many edges.
- Usually we will give the vertices and edges unique labels $v_1, v_2, \ldots$ and $e_1, e_2, \ldots$
- The set of all vertices will be V and the set of all edges is E, and the graph is $G = (V, E)$.
- An edge is just a pair of vertices. So e.g. $e_2 = (v_1, v_2)$, $e_6 = (v_5, v_7)$, and $e_9 = (v_9, v_9)$.
- A Graph is a *logical structure*. The values of the labels don't matter. All that matters is how the vertices are connected (or not connected).



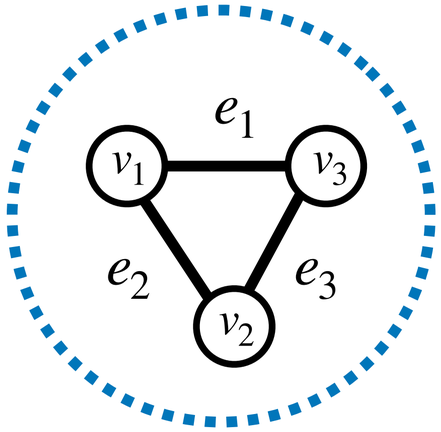


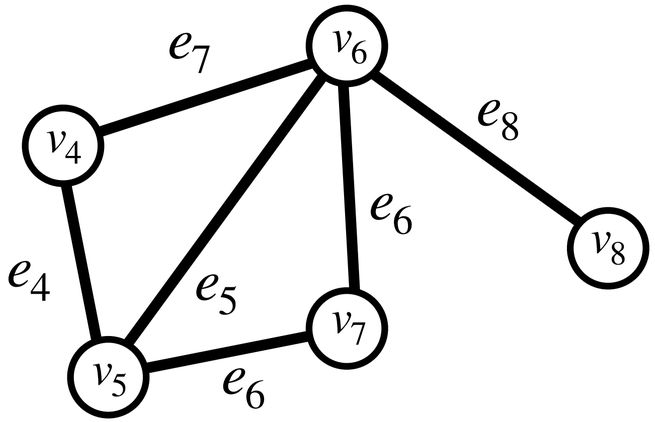


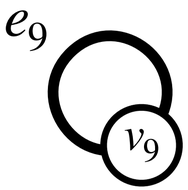
Coloring Graphs

- We say that a graph is *k-colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

Can you 2-color this one? **No.**
What about 3-coloring it?



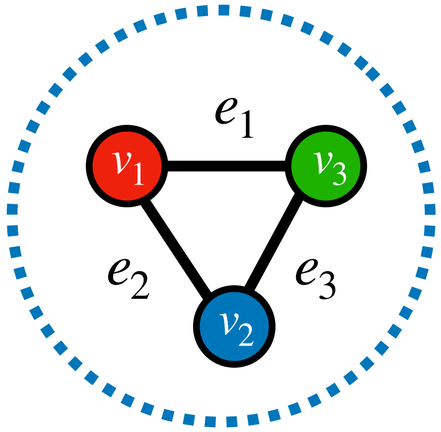


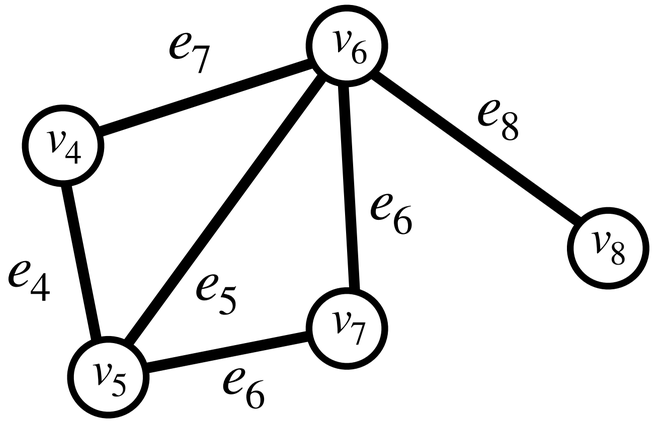


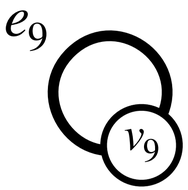
Coloring Graphs

- We say that a graph is *k-colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

Can you 2-color this one? **No.**
What about 3-coloring it? **Yes.**

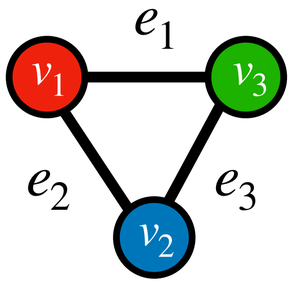


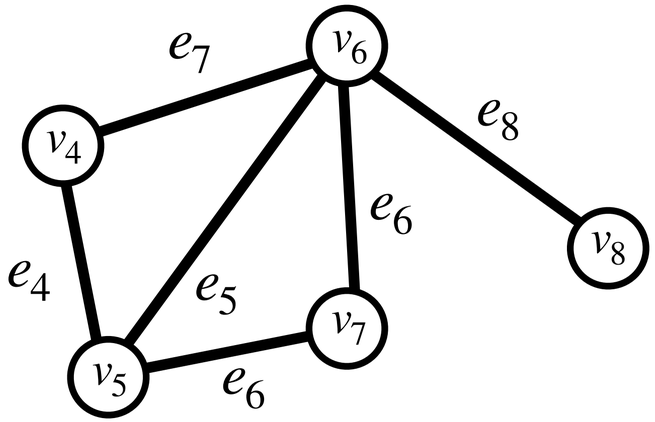




Coloring Graphs

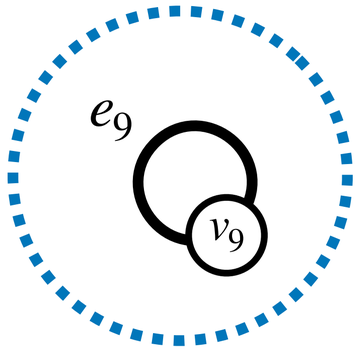
- We say that a graph is *k-colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.





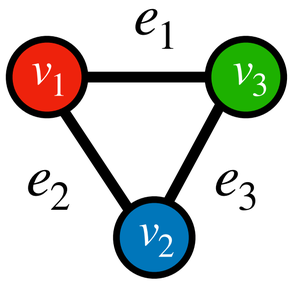
What about this one?

It can't even be 1-colored!

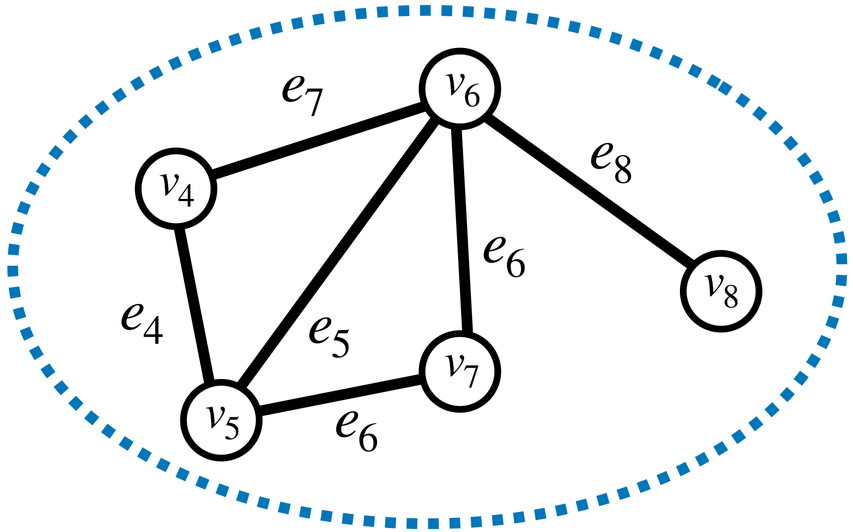


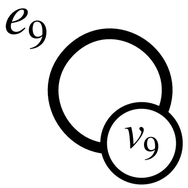
Coloring Graphs

- We say that a graph is *k-colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.



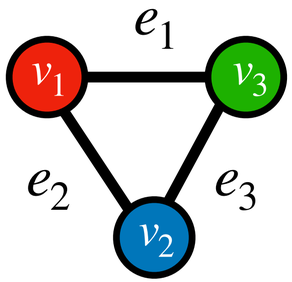
What's the smallest number of colors for this one?



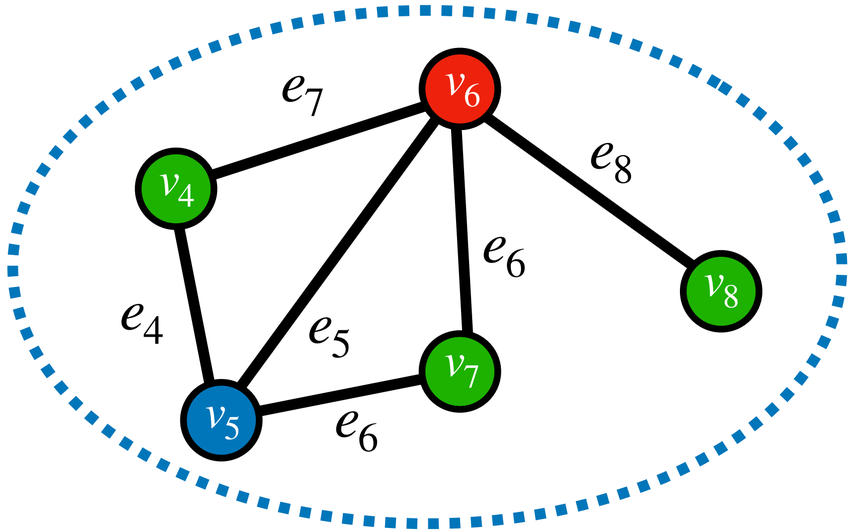


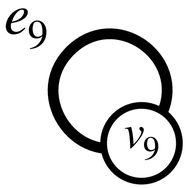
Coloring Graphs

- We say that a graph is *k-colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.



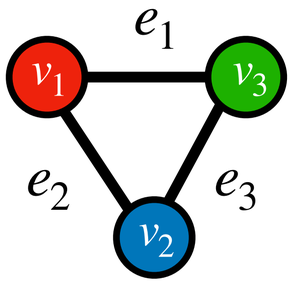
What's the smallest number of colors for this one? **3**



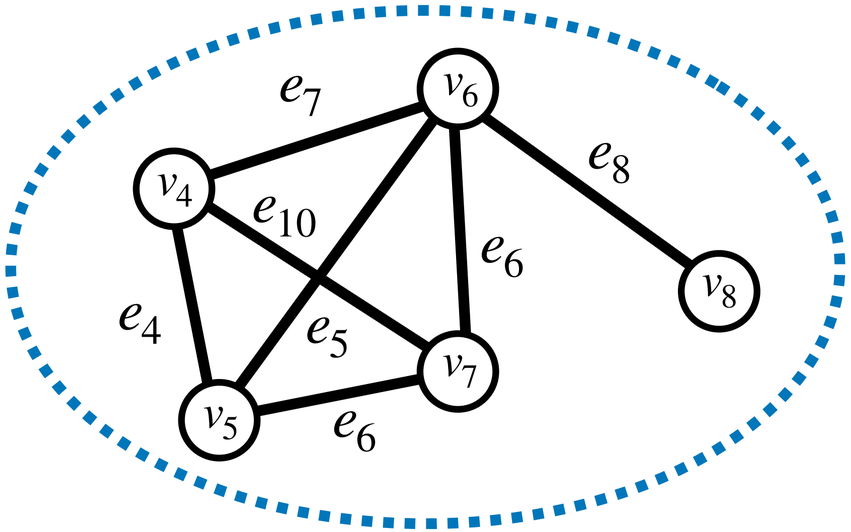


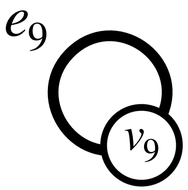
Coloring Graphs

- We say that a graph is *k-colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.



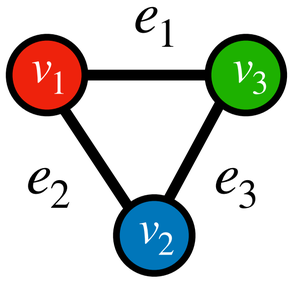
But if I add one more edge?



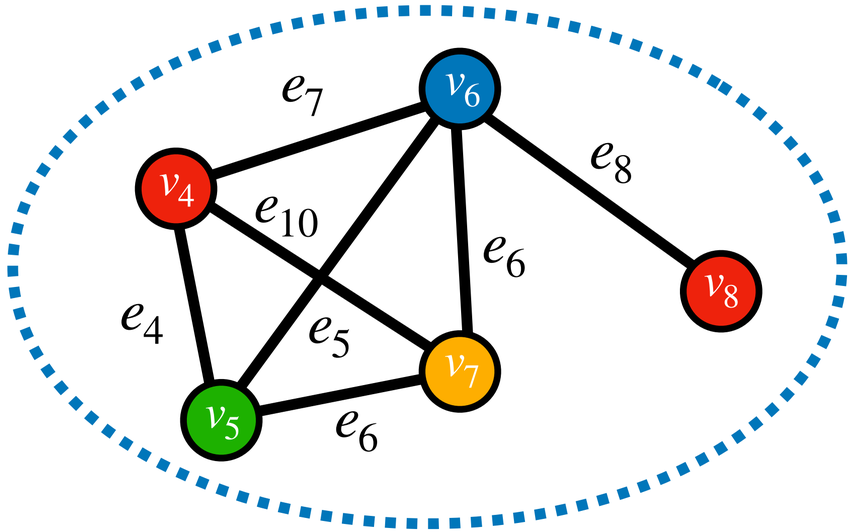


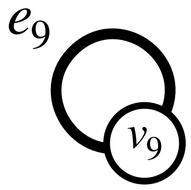
Coloring Graphs

- We say that a graph is *k-colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

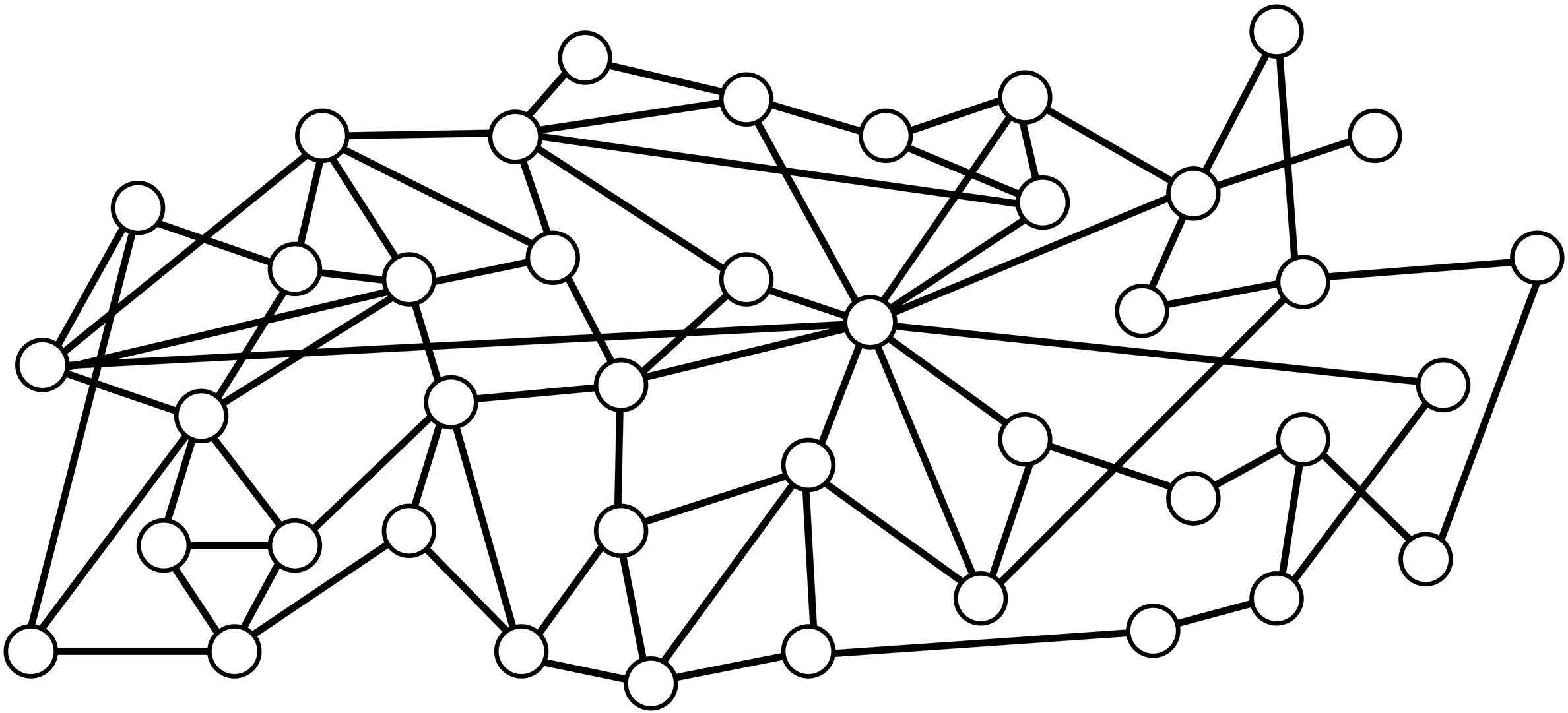


But if I add one more edge? 4





Can you 3-color this graph? It's hard, right?



Can you 3-color this graph?

1. It turns out that answering this question *in general* is NP-hard.
2. I want to prove to you that for *this* graph, the answer is **Yes, it can be 3-colored.**
3. If I just show you the coloring, then I won't be able to ask you to find it on the final exam!
4. Instead, I will *prove* to you that this graph can be 3-colored without increasing your knowledge.

What is a Proof (Reprise)

- Here is a different idea: **A proof is whatever convinces me!**
- In experimental sciences, people often speak of *proving* a statement by running *many* experiments and gathering a preponderance of evidence.
- Every experiment might individually have a large chance of a false positive, but if you do enough of them, then the chance that they *all* have a false positive can become *astronomically small*.
  Is this convincing to you?
- We're used to thinking about doing this with chemistry and physics, but we can also do it with math!

How Convincing?

- There are two properties we want from any kind of proof:
  1. *Completeness:* I can convince you of any true statement (within scope).
  2. *Soundness:* I cannot convince you of any false statement.
- When talking about proofs-by-experiment, we can weaken these requirements:
  1. *Completeness:* If the statement is **true**, the experiment always succeeds.
  2. *Soundness:* If the statement is **false**, the experiment might succeed sometimes, but it often fails.
- Doing the experiment *once* might not tell us if the statement is true.
- If we repeat it, and the statement is true, it will always succeed.
- If we repeat it enough times, and the statement is false, we will *almost certainly* witness a failure.

Interactive Experiments

- In order to convince you of a statement that I know to be true (the graph is 3-colorable), we will do an experiment *together*. I am the *prover* and you are (collectively) the *verifier*.
- You have to be careful, because I might try to *cheat* in the experiment in order to deceive you into believing a false statement.
- In this context, *soundness* means that if I try to prove a false statement to you, you have a high chance of catching me!
- *Completeness* means that if the graph is indeed 3-colorable, our experiment will never fail, and you'll never think that I'm cheating.

Zero Knowledge

- I said that I would prove that I have a 3-coloring without giving you any knowledge about what that 3-coloring is.
- Earlier we said that gaining knowledge means gaining the ability to *do something* new. How can I ensure that you won't be able to do anything new?
- Since this is our 24th lecture, you probably guessed that *you can simulate our experiment by yourself.*
- If you interact with me, I can only make the experiment succeed consistently if I truly know a 3-coloring.
- If the graph is truly 3-colorable, then you can exactly simulate what you *would* have seen in an interaction with me, even without knowing the 3-coloring.
- Since anything you can do using the experiment you could also do using the simulation (which is *identical* from your perspective), you didn't gain knowledge!

A Two Party Experiment with 3-coloring.

- First, I am going to *randomize* the coloring that I know by swapping all of the colors around (so e.g. red becomes green, green becomes blue, blue becomes red).
- Next, I'll write down the randomized 3-coloring on a sheet of paper so that I can't change it later. I am *committed* to a coloring. If I'm lying, it won't be a 3-coloring.
- After that, I'm going to cover each vertex, so that you can't see what colors I wrote down, and I'm going to show you the sheet of paper.
- You get to pick an *edge* using any strategy you choose (I suggest picking one randomly) and uncover the vertices at each end of that edge.
- If the vertices you uncover have different colors, the experiment succeeds.
- If the vertices have the same color, or one of them has a color that isn't red, green, or blue, the experiment has failed and I am a liar.

Let’s do it!

Can I Fool You?

- If I know a 3-coloring, I should pass every time. *Do you agree?*
- If I don't know a 3-coloring, then by definition there must be at least one edge with the same color at both ends, or a fourth color at one end. *Do you agree?*
- There are 77 edges in this graph. If I am cheating and you pick an edge completely randomly, then the probability that you catch me is at least 1/77, and the probability that I get away with it is at most 76/77.
- In order to fool you, I have to get away with cheating in *all* repetitions of the experiment. If you choose your random edge independently each time we repeat (you did that, right?) then the probability that I fool you is

$$\underbrace{\frac{76}{77} \cdot \frac{76}{77} \cdot \ldots \cdot \frac{76}{77}}_{r \text{ times}} = \left(\frac{76}{77}\right)^r$$

Can I Fool You?

- In order to fool you, I have to get away with cheating in *all* repetitions of the experiment. If you choose your random edge independently each time we repeat (you did that, right?) then the probability that I fool you is

$$\underbrace{\frac{76}{77} \cdot \frac{76}{77} \cdot \ldots \cdot \frac{76}{77}}_{r \text{ times}} = \left(\frac{76}{77}\right)^r$$

- If set $r = 881$, then the probability that I fool you is less than $1/1{,}000{,}000$.
- If we repeat $r = 1591$ times, the probability that I fool you is less than 2^{-30}. You are more likely to be struck by lightning this year than to be fooled.
- The probability that I fool you is the *soundness error*. If we have a soundness error of $1 - 1/\delta$ for some δ, and we repeat the proof $\delta \cdot \kappa$ times, then the probability that I fool you in all instances is $(1 - 1/\delta)^{\delta \cdot \kappa} \leq e^{-\kappa}$, which is negligible in κ.

How Can You Simulate This?

- Suppose you are convinced of the truth of the statement. The graph really is 3-colorable. *Do we agree that in this case, I will never fail the experiment?*
- In that case, you can simulate the experiment all on your own!
- What did you see? Each time we ran the experiment, I re-randomized the colors, so you saw two random colors at each end of the edge you chose. All of the other vertices were hidden by my perfect masking-tape-based commitment scheme.
- You can easily take a blank picture of the graph, choose a random edge, color the vertices at the ends of that edge two random colors, and then cover them all and uncover just the edge you chose.
- If you do this, you will see *exactly* the same distribution of color pairs as when you interact with me, even though you don't know the 3-coloring and I do!
- Therefore, you didn't gain any knowledge apart from the truth of my claim.

CS4501 Cryptographic Protocols
Lecture 24: Commitments, Intro to Zero Knowledge

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