Zero-knowledge proofs for all of NP

Noah Stephens-Davidowitz

June 9, 2023

1 Recap

In the previous lecture, we defined zero-knowledge proofs, which are interactive protocols that can be used by a prover $\mathcal{P}$ to convince a skeptical verifier $\mathcal{V}$ of a certain fact without revealing any other information. Specifically, for some language $L \subset \{0,1\}^*$, the (possibly unbounded) prover $\mathcal{P}$ wants to convince the computationally bounded verifier $\mathcal{V}$ that some string $\boldsymbol{x} \in \{0,1\}^*$ is in the language L.

However, our definition from the previous lecture was slightly too strong in that we required *perfect* zero knowledge. That is, there had to exist a simulator whose output distribution was *identical* to the distribution of the view of $\mathcal{V}^*$ in a run of the protocol with $\mathcal{P}$. In order to show examples of problems with such protocols, we had to reach for some slightly esoteric problems because, well, not many problems are known to have such protocols (besides problems that are in P, which have trivial perfect zero-knowledge protocols).

There is a much more general type of zero-knowledge protocol called *statistical zero knowledge* in which the two distributions are not required to be exactly the same, but may instead have some negligible statistical distance between them. The class of problems with such protocols is known as SZK and is very well studied.

However, since this is a cryptography class, we skip right to *computational zero-knowledge* proofs, which are typically just called *zero-knowledge proofs* (because they're the kind that people are most interested in). Here is the formal definition, first for honest verifiers, and then for malicious verifiers. I will work with both definitions, but I will only really expect you to be comfortable with the first definition.

Definition 1.1 (Honest-verifier zero knowledge)**.** *A protocol* $(\mathcal{P},\mathcal{V})$ *is an* honest-verifier zero-knowledge *protocol for a language L if the verifier $\mathcal{V}$ is efficient and the protocol satisfies the following properties.*

- **Completeness:** *For every $x \in L$, $\langle \mathcal{P},\mathcal{V}\rangle(x) = 1$ with probability* 1.
- **Soundness:** *For every $x \notin L$ and every (unbounded) prover $\mathcal{P}^*$,*

$$\Pr[\langle \mathcal{P}^*,\mathcal{V}\rangle(x) = 1]] \le 1/2 \; .$$

- **Zero Knowledge:** *There exists a PPT simulator S such that for any PPT adversary $\mathcal{A}$, there exists negligible $\varepsilon(n)$, such that*

$$\Pr[\mathcal{A}(x, S(x)) = 1] - \Pr[\mathcal{A}(x, \mathsf{view}^{\mathcal{V}}_{(\mathcal{P},\mathcal{V})}(x)) = 1] \le \varepsilon(|x|) \; ,$$

for every $x \in L$, where we use the notation $\mathsf{view}^{\mathcal{V}}_{(\mathcal{P},\mathcal{V})}(x)$ to represent the view of $\mathcal{V}$ in a run of the protocol with P on input x. ($\mathcal{A}$ does not actually need to take x as input, since x is part of the view of $\mathcal{V}$. But, it seems wise to emphasize that $\mathcal{A}$ knows x.)

Definition 1.2 (Zero knowledge against malicious verifiers)**.** *A protocol $(\mathcal{P},\mathcal{V})$ is a* zero-knowledge *protocol for a language L if the verifier $\mathcal{V}$ is efficient and the protocol satisfies the following properties.*

- **Completeness:** *For every $x \in L$, $\langle \mathcal{P},\mathcal{V}\rangle(x) = 1$ with probability* 1.
- **Soundness:** *For every $x \notin L$ and every (unbounded) prover $\mathcal{P}^*$,*

$$\Pr[\langle \mathcal{P}^*,\mathcal{V}\rangle(x) = 1]] \le 1/2 \; .$$

- **Zero Knowledge:** *There exists a PPT simulator S such that for every PPT verifier $\mathcal{V}^*$ and PPT adversary $\mathcal{A}$, there exists negligible $\varepsilon(n)$, such that*

$$\Pr[\mathcal{A}(x, S^{\mathcal{V}^*}(x)) = 1] - \Pr[\mathcal{A}(x, \mathsf{view}^{\mathcal{V}^*}_{(\mathcal{P},\mathcal{V}^*)}(x)) = 1] \le \varepsilon(|x|) \; ,$$

  for every $x \in L$, where we use the notation $\mathsf{view}^{\mathcal{V}^*}_{(\mathcal{P},\mathcal{V}^*)}(x)$ to represent the view of $\mathcal{V}^*$ in a run of the protocol with P on input x.

Intuitively, the zero-knowledge property says that interaction with $\mathcal{P}$ is "useless." For example, suppose z is the output of some one-way function, and $\mathcal{V}^*$ would like to use $\mathcal{P}$ to find a preimage of z. The zero-knowledge property rules this out. More generally, the zero-knowledge property says that "if a PPT algorithm can solve some computational problem by interacting with $\mathcal{P}$, then a PPT algorithm can solve the same problem without interacting with $\mathcal{P}$."

This is certainly a weaker notion than perfect zero knowledge, but weakening our definition is necessary for us to prove the main result of this lecture, which is truly magical. We will show how to construct zero-knowledge proofs for all of NP! (Actually, every language in IP has a zero-knowledge proof! But, we will not show this.)

1.1 Some new notation

The definitions of zero knowledge above are rather cumbersome because there are in some sense *two* adversaries: $\mathcal{V}^*$ and $\mathcal{A}$. So, we're quantifying over a lot of stuff: a simulator S, PPT malicious verifiers $\mathcal{V}^*$, adversaries $\mathcal{A}$, ε, and $x \in L$. We therefore introduce some new notation that is typically used in this setting (and in fact in many other settings).

Intuitively, for two random variables X and Y, we write

$$X \approx_c Y$$

if "X is computationally indistinguishable from Y"—that is, if no polynomial-time adversary can distinguish X from Y with non-negligible advantage.

However, if you think about it a bit, you might realize that the above intuitive definition doesn't actually make sense, since it is not clear what the security parameter is! Formally, we should actually define two families of distributions $X_1, X_2, X_3, \ldots$, and $Y_1, Y_2, Y_3, \ldots,$, one distribution for every security parameter. And then we should write

$$X_n \approx_c Y_n \; .$$

to mean that for every PPT $\mathcal{A}$ there exists negligible $\varepsilon(n)$ such that

$$\Pr[\mathcal{A}(X_n) = 1] - \Pr[\mathcal{A}(Y_n) = 1] \leq \varepsilon(n) \ .$$

This is the true formal definition of computational indistinguishability $\approx_c$.

However, we are at the point now where we're pretty happy to use slightly imprecise notation if there is relatively little risk of confusion. So, we will often write things $X \approx_c Y$, when we really mean something like the above—i.e., we don't specifically mention the security parameter and we don't specifically define the *families* of distributions $X_1, X_2, X_3, \ldots$ and $Y_1, Y_2, Y_3, \ldots$, as long as we know that we *could* achieve this level of formality if we needed to.

For example, we might write the pseudorandomness property of a PRG as simply

$$G(x) \approx_c U \ ,$$

where U is the uniform distribution on $|G(x)|$ bits. Notice that this is quite dangerous because we have not specified the security parameter *or* the distribution of x. Really, what we should do is define U_i to be the uniform distribution on i bits and to write something like

$$G(U_n) \approx_c U_{m(n)} \ ,$$

where m is the stretch of our PRG. But, again, this is cumbersome.

Anyway, given this notation, we can write the zero-knowledge property above quite succinctly. Honest-verifier zero knowledge simply states that there exists a PPT simulator S such that

$$S(x) \approx_c \mathsf{view}_{(\mathcal{P},\mathcal{V})}^{\mathcal{V}}(x)$$

for all $x \in L$. Similarly, zero knowledge against malicious verifiers says that there exists a PPT simulator S such that for every PPT $\mathcal{V}^*$,

$$S^{\mathcal{V}^*}(x) \approx_c \mathsf{view}_{(\mathcal{P},\mathcal{V}^*)}^{\mathcal{V}^*}(x)$$

for all $x \in L$.

Again, this notation is dangerous, since it is no longer clear what the security parameter is. However, we often strongly prefer it.

2 Commitment

In lecture 4, we briefly discussed commitment. A commitment scheme is sort of analogous to a box with a lock and key. I "commit" to something (say, a bit) by placing it in the box, locking the box, and handing the box to you. Later, I can open the box for you and reveal what was committed. If our locked box is secure, it should guarantee two things. First, this scheme should be *binding.* I.e., I should not be able to put one thing in the box and later reveal something else. (Magicians really like to violate the binding property of locked boxes. . .) Second, the scheme should be hiding. I.e., you should not be able to guess the contents of the box unless I reveal them to you.

Here is the formal definition.

Definition 2.1. *A commitment scheme is a PPT algorithm* $\mathsf{Com} : \{0,1\}^{\ell} \times \{0,1\}^* \to \{0,1\}^*$ *satisfying the following properties.*

- **Binding:** *For any* $r_0, r_1 \in \{0,1\}^*$ *and distinct* $\boldsymbol{m}_0, \boldsymbol{m}_1 \in \{0,1\}^\ell$, $\mathsf{Com}(\boldsymbol{m}_0, r_0) \neq \mathsf{Com}(\boldsymbol{m}_1, r_1)$.
- **Hiding:** *For any PPT adversary* $\mathcal{A}$*, there exists negligible* $\varepsilon(n)$ *such that for all* $\boldsymbol{m}_0, \boldsymbol{m}_1 \in \{0,1\}^\ell$

$$\Pr_{r \sim \{0,1\}^n}[\mathcal{A}(1^n, \mathsf{Com}(\boldsymbol{m}_1, r)) = 1] - \Pr_{r \sim \{0,1\}^n}[\mathcal{A}(1^n, \mathsf{Com}(\boldsymbol{m}_0, r)) = 1] \leq \varepsilon(n)$$

for all $n \in \mathbb{N}$.

As we saw in lecture five, we can build a commitment scheme for a single bit $\ell = 1$ from any one-way permutation (or just an injective one-way function) f with a hardcore predicate P by taking $\mathsf{Com}(b, r) := (f(r), b \oplus P(r))$. We can then extend this to a commitment for larger ℓ by concatenation. (In fact, it is possible to build a more complicated version of a commitment scheme from any one-way function, though we will not see this in this course.)

When we talk about commitment schemes, we use some useful shorthand. We say that "$\mathcal{P}$ commits to x" if $\mathcal{P}$ samples a uniformly random string $r \in \{0,1\}^n$ and sends the message $\mathsf{Com}(x, r)$ to the verifier $\mathcal{V}$. We say that "$\mathcal{P}$ opens this commitment" or "$\mathcal{P}$ reveals x" if it sends (x, r) to $\mathcal{V}$ after having committed to x.

3 Zero knowledge for all of NP

In this section, we prove the following surprising theorem, due to Goldreich, Micali, and Wigderson [GMW91].

Theorem 3.1 ([GMW91])**.** *Every language in NP has a zero-knowledge proof.*

To show that there exists a zero-knowledge proof for every language in NP, it suffices to show a zero-knowledge proof for a single NP-complete language. Then, to obtain a proof for any language L in NP, we can first reduce the statement $x \in L$ to a statement that $x' \in L'$ for our NP-complete language L'. We can then prove the latter statement.

In fact, we will show two such protocols: one for 3-Colorability and one for Hamiltonian Cycle. (We present two because the 3-Colorability protocol is too beautiful to omit, but the Hamiltonian Cycle protocol is a bit more "typical," in the sense that many zero-knowledge proofs look rather similar to the Hamiltonian Cycle protocol.)

We also note that both protocols presented below have *efficient provers*. That is, if L is an NP language and w is a witness that $x \in L$, then there is a *probabilistic polynomial-time* prover $\mathcal{P}$ that takes as input (x, w) and interacts with a PPT verifier $\mathcal{V}$ to prove that $x \in L$ in zero knowledge. So, for example, if I knew a proof to the Riemann Hypothesis (I don't), then I could convince you of its truth without revealing the proof (or any other information) to you! More practically, I might use a zero-knowledge proof to convince you that I have behaved honestly in some more complicated protocol (say, that a certain value y is actually the output $f(x)$ of some one-way function, without revealing x).

3.1 3-Colorability

Recall that a graph G is *3-colorable* if there exists some map $\phi : V(G) \to \{0,1,2\}$ (i.e., a coloring) such that for every edge $\{u, v\} \in E(G)$, $\phi(u) \neq \phi(v)$. In words, a graph is 3-colorable if you can

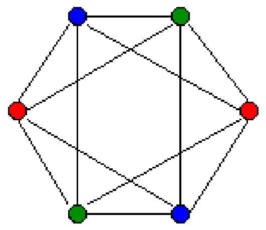

Figure 1: A 3-coloring. CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=614041.

label each vertex with one of three colors (e.g., 1 = red, 2 = green, and 3 = blue) such that no neighboring vertices have the same color. Figure 1 shows a simple example a graph with a 3-coloring from Wikipedia.

The problem of determining whether a graph is 3-colorable is NP-complete. So, if we can show a zero-knowledge protocol for three-colorability, then we get a zero-knowledge protocol for every problem in NP, which works by first reducing to three-colorability and then running the proof of 3-colorability.

First, some notation. We will need the concept of a "random permutation of a coloring ϕ." In particular, for a coloring ϕ, we write $R_\phi := \{\phi' = \pi \circ \phi \ : \ \pi \in S_3\}$ for the set of all colorings ϕ' obtained by taking a permutation π of $\{0, 1, 2\}$ and setting $\phi'(v) = \pi(\phi(v))$. In other words, ϕ' is the same as ϕ with the colors "renamed." In particular, ϕ' is a valid three-coloring if and only if ϕ is a valid three-coloring. (This is perhaps overly fancy notation, since this is just a set with six elements in it.)

The figure below shows the protocol. (When you study these protocols, it is often best to first ignore the many "checks" that the verifier must perform, since these are rather tedious and typically pretty obvious once you understand what the protocol is doing.) In words, $\mathcal{P}$ commits to a random permutation ϕ' of ϕ, $\mathcal{V}$ selects a random edge $\{u, v\} \in E(G)$, and asks $\mathcal{P}$ to reveal the colors $\phi'(u), \phi'(v)$ of the vertices of that edge. $\mathcal{V}$ then checks that the commitments are valid and that $\phi'(u) \neq \phi'(v)$ and accepts if these two things hold.

Prover		**Verifier**
INPUT: G, ϕ		INPUT: G
$\phi' \sim R_\phi$		
FOR $v \in V(G)$,		
$\quad r_v \sim \{0,1\}^{\lvert V(G)\rvert}$		
$\quad c_v := \mathsf{Com}(\phi'(v), r_v)$		
	$\xrightarrow{(c_v)_{v \in V(G)}}$	
		$\{u,v\} \sim E(G)$
	$\xleftarrow{u,v}$	
	$\xrightarrow{r_u, r_v, \phi'(u), \phi'(v)}$	
		check that $\phi'(u), \phi'(v) \in \{0,1,2\}$
		check that $c_u = \mathsf{Com}(\phi'(u), r_u)$
		check that $c_v = \mathsf{Com}(\phi'(v), r_v)$
		check that $\phi'(u) \neq \phi'(v)$
		output 1 if and only if all checks pass

Let's first check that this protocol is complete and sound. For completeness, we simply notice that if ϕ is a valid 3-coloring and $\mathcal{P}$ behaves honestly, then ϕ' is a valid coloring as well, and we must have $\phi'(u) \neq \phi'(v)$.

For soundness, we rely on the binding property of the commitment scheme. Specifically, for each v, there is at most one color $\phi'(v) \in \{0,1,2\}$ such that $\mathsf{Com}(\phi'(v), r'_v) = c_v$ for some $r'_v \in \{0,1\}^n$. If G is not three-colorable, then there is at least one edge $\{u,v\}$ with $\phi'(v) = \phi'(u)$ (or one of the commitments c_u, c_v might not open to a valid color). Therefore, with probability at least $1/|E(G)|$, the prover will fail to convince the verifier in this case.

(Of course, a soundness error of $1 - 1/|E(G)|$ is not very reassuring—and it does not even formally satisfy our definition, which required soundness error of $1/2$—but we can amplify this by repeating the protocol many times. E.g., after $|E(G)|$ independent runs of the protocol, the soundness error is less than $1/2$. As we discussed earlier, we can do this without losing the completeness or zero-knowledge properties. But, see Section 4 for some discussion of subtleties there.)

The hard part is proving that the protocol is zero knowledge. Intuitively, the protocol is zero knowledge because, no matter what a malicious verifier $\mathcal{V}^*$ does, all it sees is an opening to a commitment of two random different colors. Of course, the verifier can generate a commitment to two random different colors itself, so intuitively, "it does not learn anything."

Let's first prove honest-verifier zero-knowledge, which is not too difficult. Then we will prove zero knowledge against possibly malicious verifiers.

Theorem 3.2. *The above protocol for three-colorability is honest-verifier zero-knowledge.*

Proof. Our simulator will formalize our intuition above that "the verifier only sees two randomly colored edges." Specifically, on input a graph G, our simulator S will sample a uniformly random edge $\{u,v\} \sim E(G)$. It sets $\phi'(u)$ and $\phi'(v)$ to be two uniformly random colors. For all $w \in V(G)$ with $w \notin \{u,v\}$, it sets $\phi'(w) = 0$. (This choice is arbitrary. Any value for $\phi'(w)$ is fine.) Then, it outputs (as the verifier's simulated view) the commitments c_w to $\phi'(w)$ for all $w \in V(G)$, the edge u, v, and the openings $r_u, r_v, \phi'(u), \phi'(v)$.

It is clear that the simulator runs in polynomial time. Intuitively, the output of the simulator is indistinguishable from the view of $\mathcal{V}$ in a true run of the protocol because the only difference is the commitments. And, since the commitment is hiding, the two views should be indistinguishable.

To prove this, it is convenient to define a "hybrid simulator" S'. S' takes as input both G and a valid coloring ϕ—the same coloring ϕ used by $\mathcal{P}$. (Crucially, we are only using S' for our analysis of S. It is of course very important that our simulator S *only* takes G as input. S' is just a thought experiment to help with the proof that S works.) It then samples a uniformly random edge $\{u, v\} \in E(G)$, and samples two uniformly random different colors $\phi'(u)$ and $\phi'(v)$. It then takes ϕ' to be the unique coloring in R_ϕ with this property—i.e., the unique renaming of the coloring ϕ with these values of $\phi'(u)$ and $\phi'(v)$. Then, it outputs (as the verifier's simulated view) the commitments c_w to $\phi'(w)$ for all $w \in V(G)$, the edge u, v, and the openings openings $r_u, r_v, \phi'(u), \phi'(v)$.

A quick check shows that $S'(x, \phi)$ is distributed identically to the view of $\mathcal{V}(G)$ in a true interaction with the prover $\mathcal{P}(G, \phi)$.

So, it remains to show that

$$S(x) \approx_c S'(x, \phi) \; .$$

To do this, we rely on the hiding property of the commitment scheme. Specifically, by construction, $S(x)$ and $S'(x, \phi)$ only differ in the unopened commitments c_w for $w \notin \{u, v\}$. Call these commitments $(c_w)_{w \notin \{u,v\}}$ and $(c'_w)_{w \notin \{u,v\}}$. By a hybrid argument, which works by replacing the commitments (c_w) to (c'_w) one at a time), we see that $(c'_w)_{w \notin \{u,v\}} \approx_c (c'_w)_{w \notin \{u,v\}}$, and the result follows. □

We now prove that the protocol is actually zero knowledge even against possibly malicious verifiers. (Again, for the purposes of this course, I'll be satisfied if you understand the above proof.)

Theorem 3.3. *The above protocol for three-colorability is zero-knowledge (even against possibly malicious verifiers.*

Proof. Given some (possibly malicious) PPT verifier $\mathcal{V}^*$, we construct a PPT simulator $S^{\mathcal{V}^*}$ as follows. On input G, the simulator first samples a uniformly random edge $\{u, v\} \in E(G)$. Let $\phi'(w) := 0$ for all vertices $w \notin \{u, v\}$, and let $\phi'(u) \in \{0, 1, 2\}$ and $\phi'(v) \in \{0, 1, 2\}$ be uniformly random subject to the constraint that $\phi'(u) \neq \phi'(v)$. S then samples $r_w \sim \{0, 1\}^{|V(G)|}$ for each $w \in V(G)$, sets $c_w := \mathsf{Com}(\phi'(w), r_w)$, and passes all of the c_w to $\mathcal{V}^*$ (together with G).

$\mathcal{V}^*$ responds with some edge $\{u', v'\} \in E(G)$. If $\{u, v\} \neq \{u', v'\}$, then S simply starts the process over again. Otherwise, S reveals r_u, r_v, $\phi'(u)$, and $\phi'(v)$ to $\mathcal{V}^*$.[1]

So, suppose that G has a valid three-coloring ϕ. We need to show that (1) the view of $\mathcal{V}^*$ in a completed interaction with S on input G is indistinguishable from its view in its interaction with $\mathcal{P}$ on input (G, ϕ); and (2) that S runs in expected polynomial time. Neither fact is obvious, and both rely on the hiding property of the commitment scheme.

Imagine replacing the simulator S with a new simulator $\widetilde{S}$ that has access to ϕ and behaves as follows. First, $\widetilde{S}$ samples $\{u, v\} \in E(G)$ and $\phi'(u), \phi'(v) \in \{0, 1, 2\}$ as above. Then, it sets

[1]Formally, S is meant to output a *view* of $\mathcal{V}^*$. I.e., S should output the coins used by $\mathcal{V}^*$ together with $(c_w)_{w \in V(G)}$, $\{u, v\}$, r_u, r_v, $\phi'(u)$, and $\phi'(v)$. And formally we must argue that this view is indistinguishable from the view seen by $\mathcal{V}^*$ in its interaction with $\mathcal{P}$. In practice, we almost always leave out the actual step in which S outputs this view, and simply argue that the view seen by $\mathcal{V}^*$ in a completed interaction with S is indistinguishable from its view in an interaction with $\mathcal{P}$. We also typically ignore corner cases. E.g., we don't bother to explain how the simulator behaves if, say, $\mathcal{V}^*$ sends an invalid message.

$\widetilde{\phi} \in R_\phi$ to be the unique coloring in R_ϕ with $\widetilde{\phi}(u) = \phi'(u)$ and $\widetilde{\phi}(v) = \phi'(v)$. (Notice that there is in fact one coloring in R_ϕ that satisfies this.) It then sends commitments corresponding to $\widetilde{\phi}$ to $\mathcal{V}^*$ and continues the protocol just like S does—receiving a response $\{u', v'\} \in E(G)$, restarting if $\{u', v'\} \neq \{u, v\}$, and otherwise opening the commitments to $\phi'(u)$ and $\phi'(v)$.

Let's first see that $\widetilde{S}$ produces the appropriate view and runs in expected polynomial time. (Of course, $\widetilde{S}$ is not itself a valid simulator, because it receives a coloring ϕ as input. So, we will have to compare S and $\widetilde{S}$ to finish the proof.) To see this, first notice that the random variable $\{u, v\}$ is actually independent of the coloring $\widetilde{\phi}$ and is therefore independent of the first message sent by $\widetilde{S}$. Therefore, regardless of how $\mathcal{V}^*$ computes $\{u', v'\}$, we have $\{u', v'\} = \{u, v\}$ with probability exactly $1/|E(G)|$, independent of $\{u', v'\}$. (This step was quite delicate. E.g., notice that these random variables are not independent for S, since $\mathcal{V}^*$ could choose $\{u, v\}$ in a way that depends on the commitments sent by S, and the commitments sent by S do depend on $\{u, v\}$.) It follows that $\widetilde{S}$ will complete the protocol with probability exactly $1/|E(G)|$ each time that it calls $\mathcal{V}^*$, which in particular means that it runs in expected polynomial time. Furthermore, since $\{u, v\}$ is independent of $\{u', v'\}$, it follows that the view of $\mathcal{V}^*$ in a completed interaction with $\widetilde{S}$ is exactly the same as its view in an honest interaction with $\mathcal{P}$.

It remains to show that the behavior of S is indistinguishable from the behavior of $\widetilde{S}$. By a hybrid argument, the commitments sent by S are indistinguishable from the commitments sent by $\widetilde{S}$. I.e., we can replace the first message sent by S by the first message sent by $\widetilde{S}$. Finally, we notice that, after replacing the first message sent by S with the first message sent by $\widetilde{S}$, the two simulators behave identically. Therefore, the probability that S finishes the protocol on each try must be at most negligibly far from $1/|E(G)|$, so that its running time is also expected polynomial. And, the view of $\mathcal{V}^*$ in a completed interaction with S is indistinguishable from its view in a completed interaction with $\widetilde{S}$. Since the latter view was exactly the same as the view of $\mathcal{V}^*$ in an honest interaction with $\mathcal{P}$, the protocol is in fact zero knowledge.[2] $\square$

3.2 Hamiltonian Cycle

Recall that a Hamiltonian cycle in a graph is a cycle that includes each vertex exactly once. Figure 2 shows a nice example from Wikipedia. The Hamiltonian Cycle problem is the computational problem that asks us to decide whether a given graph has a Hamiltonian cycle $C = (e_1, \ldots, e_{|V(G)|})$, and it is known to be NP-complete.

We present a zero-knowledge proof for Hamiltonian Cycle as follows. In words, the protocol works as follows. The prover $\mathcal{P}$ samples a uniformly random permutation $\pi^* \sim \Pi_G$ from the set of permutations $\Pi_G := \{\pi : V(G) \to V(G)\}$ of the vertices of the graph G. It then commits to the edges of the permuted graph in a random order. I.e., for each edge $e := \{u, v\} \in E(G)$, it commits to $\pi^*(e) := \{\pi^*(u), \pi^*(v)\}$. The verifier $\mathcal{V}$ then samples a uniformly random bit $b \sim \{0, 1\}$ and sends it to $\mathcal{P}$. If $b = 0$, then $\mathcal{P}$ responds by opening the commitments to $\{\pi^*(u), \pi^*(v)\}$ for each $\{u, v\} \in C$. I.e., $\mathcal{P}$ shows that the committed graph has a Hamiltonian cycle. If $b = 1$, $\mathcal{P}$ opens all commitments and reveals π^* as well. I.e., $\mathcal{P}$ shows that the committed graph is a valid permutation

[2] I have cheated quite a bit here, since in order for the hybrid argument to work, the distinguisher needs access to the graph coloring—which is of course hard to compute. One can handle this via non-uniformity. I.e., one says that if the proof is false, then for some verifier $\mathcal{V}^*$ and infinitely many n, there must be some graph G with n vertices with coloring ϕ for which one can distinguish the simulated view from a real view. One can then hard code that graph G and ϕ into a distinguisher for the commitment scheme to derive a contradiction. To get a uniform reduction takes significantly more work.

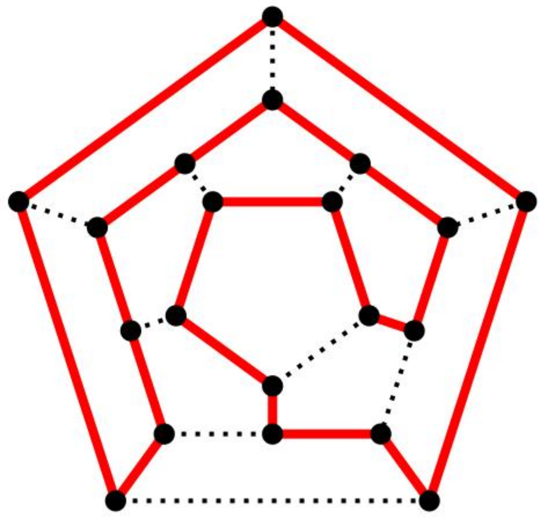

Figure 2: A Hamiltonian cycle. (By Wikipedia user Patrick24 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5461910.)

of the original graph.

```
Prover                                                              Verifier
INPUT: $G, C$                                                       INPUT: $G$

$\pi^* \sim \Pi_G$
FOR $e \in E(G)$,
    $r_e \sim \{0,1\}^{|V(G)|}$
    $c_e := \mathsf{Com}(\pi^*(e), r_e)$
                $\xrightarrow{(c_e)_{e \in E(G)} \text{ (in random order)}}$
                                                                    $b \sim \{0,1\}$
                $\xleftarrow{b}$
IF $b = 0$
                $\xrightarrow{(\pi^*(e), r_e)_{e \in C} \text{ (in random order)}}$
                                                                    check that openings are valid
                                                                    check that $\pi^*(e)$ form a cycle
IF $b = 1$
                $\xrightarrow{(\pi^*(e), r_e)_{e \in E(G)}, \pi^*}$
                                                                    check that openings are valid
                                                                    check that openings match $\pi^*(E)$
```

The proof that this is a zero-knowledge protocol is essentially the same as the proof for 3-Colorability. In particular, completeness and soundness are relatively straightforward.

To prove zero knowledge, we construct a simulator S for each malicious verifier $\mathcal{V}^*$ as follows. S samples a uniformly random bit $b \sim \{0,1\}$. If $b = 0$, S commits to the edges of a uniformly random Hamiltonian cycle C^* on $V(G)$ together with $|E(G)| - |V(G)|$ additional random edges. If $b = 1$, S commits to a uniformly random permutation of the edges of G. In response $\mathcal{V}^*$ sends a bit b'. If $b' \neq b$, then S simply restarts the process. Otherwise, if $b = b' = 0$, S opens the commitment to C^* (but not to the remaining edges). If $b = b' = 1$, S opens all commitments and sends the permutation to $\mathcal{V}^*$.

We leave it as an exercise to prove that S is in fact a valid simulator.

4 Bonus content: On auxiliary input and zero knowledge under repetition

The two protocols that we have shown have soundness error that is quite large—$1 - 1/|E|$ and $1/2$ respectively. In practice, we would like to repeat the protocol polynomially many times (in sequence, not in parallel) in order to lower the soundness error. This *does* work, but to make the proof go through, one must be careful.

The standard way to handle this is to introduce a notion of *zero knowledge with auxiliary input.* This modifies the definition of zero knowledge so that the malicious verifier and the simulator each receive as input an additional string $z \in \{0,1\}^*$, with the only restriction on z being that $|z| \leq \text{poly}(|x|)$. This is called the *auxiliary input.* The protocol is zero knowledge with auxiliary input if for any such string z, $S^{\mathcal{V}^*}(x, z)$ is indistinguishable from $\mathsf{view}\langle \mathcal{V}^*(x, z), \mathcal{P}(x, w)\rangle$.

Intuitively, this definition of zero knowledge with auxiliary input captures the notion that "no matter what $\mathcal{V}^*$ knows at the start of the protocol, it learns nothing after interacting with $\mathcal{P}$." Perhaps a better name for it would therefore be "prior information." E.g., the auxiliary input might be a partial coloring of the input graph G. A good zero knowledge protocol should not reveal anything even to an adversary who happens to know such a partial coloring. So, this is really the *right* definition of zero knowledge. In this course, we are simply ignoring the auxiliary input to save ourselves some ink.

The auxiliary input definition is also far more robust. In particular, it is stable under sequential repetition. So, we can take a protocol that is zero knowledge with auxiliary input, repeat it many times in sequence, and still get a protocol that is zero knowledge with auxiliary input. To prove this, in the ith run of the protocol, we consider the view from the previous $i-1$ runs of the protocol to be auxiliary input.

It is straightforward to see that the above protocols are still zero knowledge with auxiliary input. But, one can create contrived examples in which this is not the case. For example, we could add a first message to our 3-coloring protocol, sent by the verifier, which the prover $\mathcal{P}$ ignores *unless* the first message happens to be a commitment to a valid 3-coloring of the graph G. Repeating the resulting protocol many times would not be wise :).

References

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3), 1991. 4