Lecture 2: Introduction and definitions of zero-knowledge (continued)

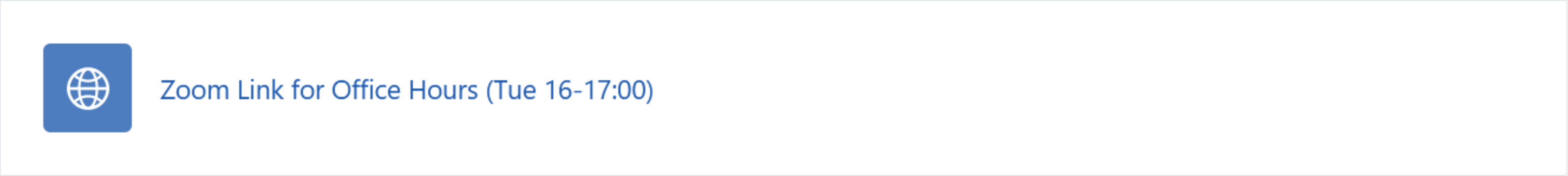
Zero-knowledge proofs

263-4665-00L

Lecturer: Jonathan Bootle

Announcements

- Office hours: Tuesdays, 16-17:00



Zoom Link for Office Hours (Tue 16-17:00)

- Mistake in simulator for GI protocol (notes not presented last time)
- Corrected today

Last time

- Definitions of complexity classes and IPs

- IP for graph non-isomorphism – increased proving power

- Definition of zero-knowledge proofs

Agenda

- **ZKP for graph isomorphism**

- Variations of zero-knowledge

- Variations of soundness

- Commitment schemes

- ZK for an **NP**-complete problem

Zero-knowledge

No knowledge gained by verifiers who could already produce the proof by themselves.

V may not follow the protocol

Definition:

Let (P, V) be an IP for $\mathcal{L}$.

everything V sees

- The *verifier's view* is $\text{View}_V^P = (x, s, a_1, \ldots, a_{k(x)})$.
- (P, V) is *perfect zero-knowledge* if ∀ efficient V^*, ∃ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^P\right\} = \{S(V^*, x)\}$.
- If so, (P, V) is a *perfect zero-knowledge proof (perfect ZKP)*.

expected probabilistic polynomial time

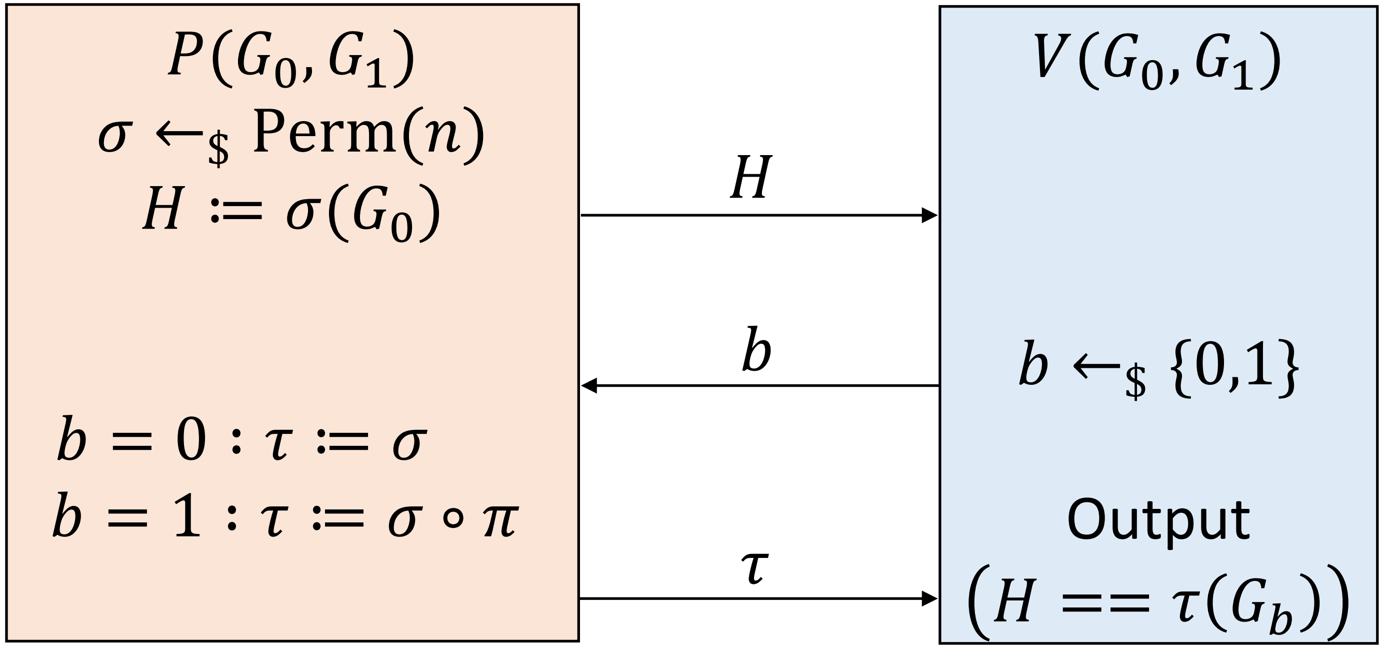
equal as probability distributions

Note: $\text{View}_{V^*}^P$ of *malicious* verifier V^*

Perfect ZKP for Graph Isomorphism

- $\mathcal{L}_{GI} \coloneqq \{(G_0, G_1) : G_0, G_1 \text{ graphs}, G_0 \cong G_1\}.$
- $\mathcal{R}_{GI} \coloneqq \{((G_0, G_1), \pi) : (G_0, G_1) \in \mathcal{L}_{GI}, G_0 = \pi(G_1)\}.$

Efficiently checkable so $\mathcal{L}_{GI} \in$ **NP**



$P(G_0, G_1)$

$\sigma \leftarrow_\$ \mathrm{Perm}(n)$

$H \coloneqq \sigma(G_0)$

H

$V(G_0, G_1)$

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

b

$b = 0 : \tau \coloneqq \sigma$

$b = 1 : \tau \coloneqq \sigma \circ \pi$

τ

Output $(H == \tau(G_b))$

Not known to be **NP**-complete

Almost in **P** (!)

Prove completeness, soundness and perfect ZK

Domain checks e.g. $\tau \in \mathrm{Perm}(n)$ sometimes extremely important but usually omitted

Completeness and soundness analysis

Completeness: $x \in \mathcal{L}_{GI}, G_0 \cong G_1$

- If $b = 0$ then $\tau = \sigma$.
- If $b = 1$ then $\tau = \sigma \circ \pi$.
- So V always accepts.

Soundness: $x \notin \mathcal{L}_{GI}, G_0 \ncong G_1$

- No graph H has both $G_0 \cong H$ and $G_1 \cong H$.
- Otherwise $G_0 \cong G_1$ #
- After P sends H, we have $G_b \cong H$ for at most one of $b = 0{,}1$.
- With probability $^1/_2$, V chooses b with no isomorphism and outputs 0

Protocol

$$\begin{tikzcd} G_0 \arrow[dr, "\sigma"'] & G_1 \arrow[l, "\pi"'] \\ & H \end{tikzcd}$$

$b = 0$ check

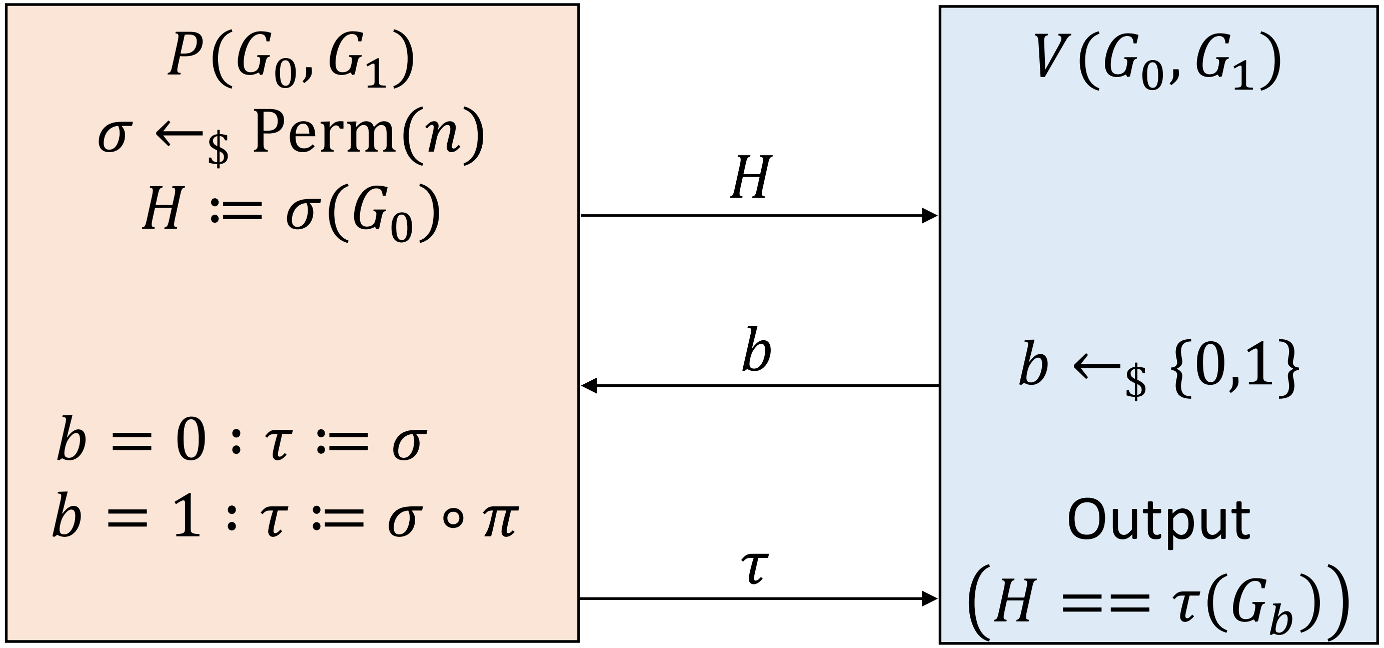
$$\begin{tikzcd} G_0 \arrow[dr, "\tau"'] & G_1 \\ & H \end{tikzcd}$$

$b = 1$ check

$$\begin{tikzcd} G_0 & & G_1 \arrow[dl, "\tau"] \\ & H & \end{tikzcd}$$

Simulation of the protocol

Corrected from L1 slides



$P(G_0, G_1)$

$\sigma \leftarrow_\$ \text{Perm}(n)$

$H \coloneqq \sigma(G_0)$

$b = 0 : \tau \coloneqq \sigma$

$b = 1 : \tau \coloneqq \sigma \circ \pi$

H

b

τ

$V(G_0, G_1)$

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

Output

$(H == \tau(G_b))$

```
$S(V^*, G_0, G_1)$
1. $\tau \leftarrow_\$ \text{Perm}(n)$.
2. $B \leftarrow_\$ \{0,1\}$.
3. $H \coloneqq \tau(G_B)$.
4. $b \leftarrow_\$ V^*(G_0, G_1, H)$.
5. If $b \neq B$ go back to 1.
6. Output $(G_0, G_1, \tau, H, b)$.
```

$V^*(G_0, G_1)$

???

S works backwards and guesses b

Perfect zero-knowledge analysis

```
$S(V^*, G_0, G_1)$
1. $\tau \leftarrow_\$ \mathrm{Perm}(n)$.
2. $B \leftarrow_\$ \{0,1\}$.
3. $H := \tau(G_B)$.
4. $b \leftarrow_\$ V^*(G_0, G_1, H)$.
5. If $b \neq B$ go back to 1.
6. Output $(G_0, G_1, \tau, H, b)$.
```

$V^*(G_0, G_1)$

???

What does the verifier see?

- Real protocol: $View_{V^*}^{P} = (G_0, G_1, \tau, H, b)$.
- $\tau = \sigma$ or $\tau = \sigma \circ \pi$ is uniform from $\mathrm{Perm}(n)$.
- τ, H, b random satisfying $H = \tau(G_b)$.

But V^* may sample b from a strange distribution

Why is the simulator valid? (efficient, indistinguishable)

- S is efficient because V^* is, and we expect to clear Step 5 in 2 tries.
- $\tau(G_0)$ and $\tau(G_1)$ identically distributed as $G_0 \cong G_1$.
- Simulated τ, b are distributed as in a real protocol.
- $H = \tau(G_b)$ so H is uniquely determined and correctly distributed too.

Agenda

- ZKP for graph isomorphism ✓
- **Variations of zero-knowledge**
- Variations of soundness
- Commitment schemes
- ZK for an **NP**-complete problem

Black-box zero-knowledge

Definition:

Let (P, V) be an IP for $\mathcal{L}$. We say (P, V) is

expected probabilistic polynomial time

- *perfect zero-knowledge* if $\forall$ efficient V^*, $\exists$ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^{P}\right\} = \{S(V^*, x)\}$.
- *black-box zero-knowledge* if $\exists$ efficient simulator S such that $\forall$ efficient V^*, $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^{P}\right\} = \left\{S^{V^*}(x)\right\}$.

Oracle access to next message function of V^*

Example: GI protocol

Zero-knowledge for honest verifiers

ZK: a property of provers against different classes of verifiers

Definition:

Let (P, V) be an IP for $\mathcal{L}$. We say (P, V) is

- *perfect zero-knowledge* if ∀ efficient V^*, ∃ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^{P}\right\} = \{S(V^*, x)\}$.
- *semi-honest-verifier zero-knowledge (SHVZK)* if ∃ efficient simulator S such that $\forall x \in \mathcal{L}$, $s \in \{0,1\}^*$, we have $\left\{\text{View}_{V(s)}^{P}\right\} = \{S(x, s)\}$.
- *honest-verifier zero-knowledge (HVZK)* if ∃ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\{\text{View}_{V}^{P}\} = \{S(x)\}$.

~~expected~~ probabilistic polynomial time

Easier to design simulators for the honest verifier only

Transformations to ZK against malicious verifiers

Indistinguishability

D tries to guess whether it is sampling from X or Y

Definition:

Let $\lambda \in \mathbb{N}$ be a security parameter. Let $X = (X_\lambda)_{\lambda \in \mathbb{N}}, Y = (Y_\lambda)_{\lambda \in \mathbb{N}}$ be two ensemble of random variable. We say that X, Y are:

$X = Y$, identically distributed

- *Perfectly indistinguishable* if ∀ algorithms D, $\forall \lambda \in \mathbb{N}$,
$$\left|\Pr\left[D\left(1^\lambda, X_\lambda\right) = 1\right] - \Pr\left[D\left(1^\lambda, Y_\lambda\right) = 1\right]\right| = 0$$
- *Statistically indistinguishable* if ∀ algorithms D, $\forall \lambda \in \mathbb{N}$,
$$\left|\Pr\left[D\left(1^\lambda, X_\lambda\right) = 1\right] - \Pr\left[D\left(1^\lambda, Y_\lambda\right) = 1\right]\right| \le \mathrm{negl}(\lambda)$$
- *Computationally indistinguishable* if ∀ efficient algorithms D, $\forall \lambda \in \mathbb{N}$,
$$\left|\Pr\left[D\left(1^\lambda, X_\lambda\right) = 1\right] - \Pr\left[D\left(1^\lambda, Y_\lambda\right) = 1\right]\right| \le \mathrm{negl}(\lambda)$$

$X \approx_s Y$, can only distinguish if lucky

$X \approx_c Y$, can only distinguish if lucky or very hardworking

Statistical and computational zero-knowledge

PZK ⊆ SZK ⊆ CZK

Definition:

Let (P, V) be an IP for $\mathcal{L}$. We say (P, V) is

- *perfect zero-knowledge* if ∀ efficient V^*, ∃ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^{P}\right\} = \{S(V^*, x)\}$.
- *Statistical zero-knowledge* if ∀ efficient V^*, ∃ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^{P}\right\} \approx_s \{S(V^*, x)\}$.
- *Computational zero-knowledge* if ∀ efficient V^*, ∃ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^{P}\right\} \approx_c \{S(V^*, x)\}$.

Payoff: computational ZKP for all **NP**

~~expected~~ probabilistic polynomial time

SZK for **NP**-complete problems unlikely

“Hints” used to break soundness or ZK

Auxiliary inputs and states

Models efficient computation

Prover can take witness as input

Definition:

Let $P, V : \{0,1\}^* \to \{0,1\}^*$. Let $k : \{0,1\}^* \to \mathbb{N}$.

A $k(x)$-*move interaction* between P and V

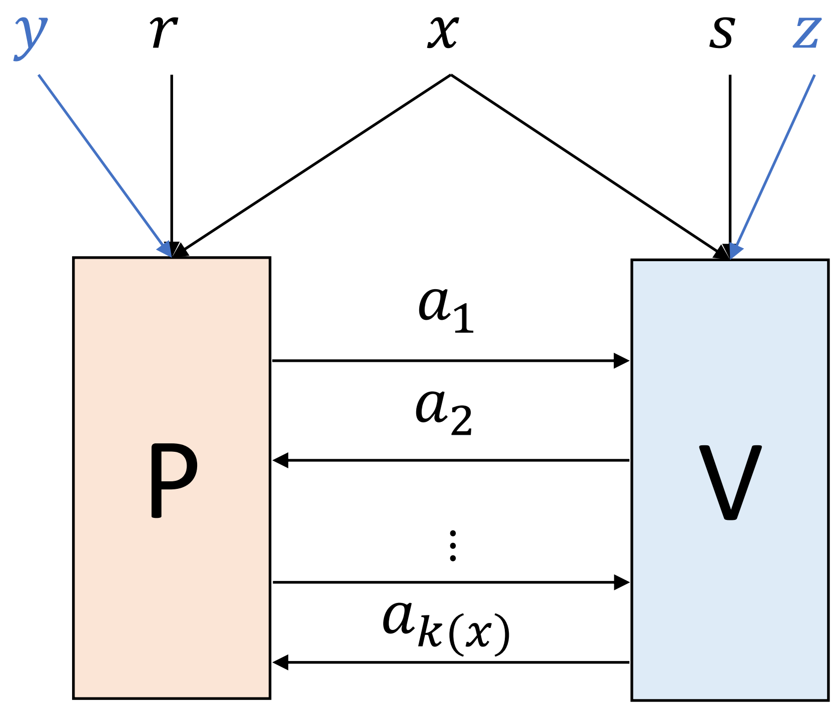
on input $x \in \{0,1\}^*$ is defined by:

- $a_1, st_{P,1} = P(x, y, r)$
- $a_2, st_{V,2} = V(x, z, a_1, s)$
- $a_3, st_{P,3} = P\left(x, y, st_{P,1}, a_1, a_2, r\right)$

  $\vdots$
- $a_{k(x)}, st_{V,k(x)} = V\left(x, z, st_{V,k(x)-2}, a_1, \dots, a_{k(x)-1}, s\right)$

The interaction is denoted $\langle P(y, r), V(z, s)\rangle(x) = a_{k(x)}$.

The *transcript* is $(x, a_1, \dots, a_{k(x)})$.



Security with auxiliary inputs

Definition:

(P, V) are an *IP with auxiliary inputs* if is V efficient (polynomial time in $|x|$) and they satisfy:

- *Completeness*

Completeness error $1 - \epsilon_C$

$\forall x \in \mathcal{L}, \exists y \in \{0,1\}^*$ *such that* $\forall z \in \{0,1\}^*, \Pr_{r,s}[\langle P(y,r), V(z,s)\rangle(x) = 1] \geq {}^3/_4$

y is often an **NP** witness

- *Soundness*

$\forall x \notin \mathcal{L}, \forall P^*, \forall y, z \in \{0,1\}^*, \Pr_{r,s}[\langle P^*(y,r), V(z,s)\rangle(x) = 1] \leq {}^1/_2$

No hint will help break soundness

Soundness error ϵ_S

Zero-knowledge with auxiliary inputs

Definition:

Idea: V^* might start with some knowledge

Let (P, V) be an IP for $\mathcal{L}$.

(P, V) is *perfect zero-knowledge with auxiliary inputs* if $\forall$ efficient V^*, $\forall x, y$ satisfying completeness, $\forall z$,

$\exists$ efficient simulator S such that $\forall x \in \mathcal{L}$, we have $\left\{\text{View}_{V^*}^{P}\right\} = \{S(V^*, x, z)\}$.

Normal ZK not generally preserved
under sequential composition!

Theorem: auxiliary-input ZK is preserved under sequential composition of protocols.

Auxiliary input ZK still not generally preserved
under parallel composition!

Agenda

- ZKP for graph isomorphism ✓
- Variations of zero-knowledge ✓
- **Variations of soundness**
- Commitment schemes
- ZK for an **NP**-complete problem

Interactive arguments

- Protocols whose soundness only holds against *efficient* provers
- So we can prove soundness using cryptographic assumptions!

Definition:

(P, V) is an *interactive argument* for L if P runs in polynomial time, (P, V) is complete; and

- (P, V) is *computationally sound* i.e.

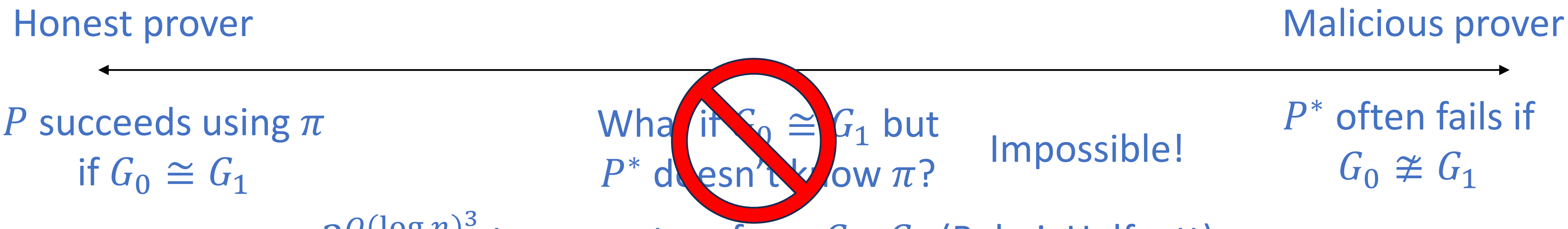
$\forall$ sufficiently long $x \notin \mathcal{L}$, $\forall$ efficient P^*, $\forall y, z \in \{0,1\}^*$,

$$\Pr[\langle P^*(y), V(z)\rangle(x) = 1] \leq {}^{1}\!/_{2}$$

ZK: a property of verifiers against different classes of provers

Knowledge soundness

- Can a GI prover succeed without knowing π such that $\pi(G_0) = G_1$?
- Completeness and soundness offer no guarantees…



Honest prover

Malicious prover

P succeeds using π if $G_0 \cong G_1$

What if $G_0 \cong G_1$ but P^* doesn't know π?

Impossible!

P^* often fails if $G_0 \ncong G_1$

$2^{O(\log n)^3}$ to compute π from G_0, G_1 (Babai, Helfgott)
So not considered knowledge

- Knowledge = efficient computation. Show that successful P^* can efficiently compute π.

Proofs of knowledge

Recall **NP** relations

$R = \{(x, w)\} \subseteq \{0,1\}^* \times \{0,1\}^*$

Definition:

Let $\mathcal{R}$ be a relation. Let $\kappa : \mathbb{N} \to [0,1]$. An IP (P, V) is a *proof of knowledge* for $\mathcal{R}$ with *knowledge-soundness error* κ if

Expected polynomial time

Not in $\mathcal{L}$ a priori

$\exists$ polynomial q and efficient *extractor* E such that $\forall P^*$, $\forall x, y \in \{0,1\}^*$,

if $\Pr_{r,s}[\langle P^*(y), V(s)\rangle(x) = 1] = \epsilon(x, y) \geq \kappa(|x|)$ then

$\Rightarrow x \in L$ so we have soundness

$E^{P^*}(x)$ outputs w with $(x, w) \in \mathcal{R}$ with probability at least

Oracle access to next message function

Can't assume P^* honest

$$\frac{\epsilon(x,y) - \kappa(|x|)}{q(|x|)}.$$

Dual to black-box ZK definition

S, E have similar 'superpowers'

Notes on knowledge soundness

Arguments of knowledge/computational knowledge soundness:

- Restrict to efficient P^*
- Allow E to output a witness or break a cryptographic assumption
- $\mathcal{R}' \coloneqq \left\{ (pp, x, w) : \begin{array}{c} pp \text{ cryptosystem parameters} \\ (x, w) \in \mathcal{R} \vee w \text{ breaks } pp \text{ somehow} \end{array} \right\}$

Useful even when soundness is trivial.

- Let $\mathbb{G}$ be a group of prime order p generated by $g \in \mathbb{G}$
- $\mathcal{L}_{DLOG} \coloneqq \{(\mathbb{G}, g, h, p) : \exists x \in \mathbb{Z}_p, h = x \cdot g\}$ is trivial. Why?
- Knowing x is non-trivial and useful for ID schemes. $(pk, sk) = (h, x)$.

GI is a proof of knowledge with $\kappa \equiv {}^1/_2$

Proof:

WLOG P^* is *deterministic* (exercise). $\Pr[V \text{ accepts}] \in \{0, {}^1/_2, 1\}$. Then if $\Pr[V \text{ accepts}] > {}^1/_2$, we have $\Pr[V \text{ accepts}] = 1$.

If $\leq {}^1/_2$ don't need to do anything

What does E do?

Black-box access to P^*.

```
                $E(G_0, G_1)$
1. $H \leftarrow_\$ P^*(G_0, G_1)$
2. $\tau_0 \leftarrow_\$ P^*(G_0, G_1, H, 0)$
3. $\tau_1 \leftarrow_\$ P^*(G_0, G_1, H, 1)$
4. Output $\pi \coloneqq \tau_0 \circ \tau_1^{-1}$.
```

$P^*(G_0, G_1)$

???

Runtime $\lesssim 2$ executions of P^*

Strict poly time!

$$\begin{tikzcd}
G_0 \arrow[dr, "\tau_0"'] & & G_1 \arrow[dl, "\tau_1"] \\
& H &
\end{tikzcd}$$

Why is E valid? (efficiency, outputs witness)

$\Pr[V \text{ accepts}] = 1$ so $H = \tau_0(G_0)$ and $H = \tau_1(G_1)$ hence $G_0 = \pi(G_1)$.

ETH Zurich, Department of Computer Science | Dr. Jonathan Bootle
HS 2023 | Karen Klein, Varun Maram, Antonio Merino Gallardo

Zero-Knowledge Proofs
Exercise 2

Note: you may want to read the unpresented slides on the interactive proof for graph isomorphism from Lecture 1 before attempting these questions.

2.1 Quadratic Residues

Let n be a positive integer. An element $x \in \mathbb{Z}_n^*$ is a *quadratic residue* modulo n if there exists $w \in \mathbb{Z}_n^*$ such that $x = w^2 \bmod n$. If n is composite, it is believed to be computationally hard to decide whether a random $x \in \mathbb{Z}_n^*$ is a quadratic residue modulo n without knowing the factorization of n.[1]

Consider the problem of proving that a value $x \in \mathbb{Z}_n^*$ is a quadratic residue modulo an integer n. Formally, the prover and verifier are given input (n, x) such that there exists w satisfying $x = w^2 \bmod n$.

For each of the following protocols, say whether it is complete, sound and zero-knowledge, and justify your answer.

Protocol 1.

$P \rightarrow V$ Generate $u \leftarrow_\$ \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send (u, y) to the verifier.

$V \rightarrow P$ Generate $b \leftarrow_\$ \{0, 1\}$ and send it to the prover.

$P \rightarrow V$ Compute $z \leftarrow uw^b$ and send it to the verifier. The verifier accepts if $z^2 = yx^b \bmod n$ and otherwise rejects.

Solution: Not zero knowledge, since the verifier can send back $b = 1$, receive $z = uw$ and compute $u^{-1}z = w$ using the u received in the first step to obtain the secret.

Formally, assume the protocol was zero knowledge. Then for every (potentially malicious) verifier V^* there exists an efficient simulator S that for every quadratic residue x can generate a transcript $((u, y), b, z)$ that is distributed equally to the transcript in the view of V^* in an execution of the protocol with an honest prover P. Now consider a cheating verifier V^* that sets $b = 1$, instead of sampling b uniformly at random. Then the simulator S on input x, n needs to generate $u, y, z \in \mathbb{Z}_n^*$ such that $u^2 = y$ and $z^2 = yx$. But if such a simulator exists, we can construct an efficient algorithm A that can decide quardatic residuosity: A first runs S on input $(V^*, (x, n))$. If x is a quadratic residue then A receives from S values $u, y, z \in \mathbb{Z}_n^*$ such that $u^2 = y$ and $z^2 = yx$. From those values one can compute a square root of x as $w = u^{-1}z$; thus A holds a proof for x being a quadratic residue. On the other hand, if x is a quadratic non-residue, then S can behave arbitrarily, but clearly cannot come up with values $u, y, z \in \mathbb{Z}_n^*$ such that $u^2 = y$ and $z^2 = yx$, since such values do not even exist in this case. (To see this, note that $z^2 = u^2x$ implies that

[1] Here, by "random x" we mean uniform among the elements x in $\mathbb{Z}_n^*$ for which $\mathcal{J}_n(x) = 1$, where $\mathcal{J}_n$ denotes the Jacobi symbol for composite n. (See e.g. [KL21, Section 15.4.3] for reference.)

x is a quadratic residue.) Hence, if S fails, then A deduces that x is a quadratic non-residue. Thus, assuming that quadratic residuosity is hard, the above protocol is not zero knowledge.

Protocol 2.

$P \to V$ Generate $u \leftarrow_\$ \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send y to the verifier.

$V \to P$ Set $b \leftarrow_\$ 0$ and send it to the prover.

$P \to V$ Compute $z \leftarrow uw^b$ and send it to the verifier. The verifier accepts if $z^2 = y \bmod n$ and otherwise rejects.

Solution: Not sound since a prover can just send a random u in the third round, indepentently of x, and thus fool the verifier.

Protocol 3.

$P \to V$ Generate $u \leftarrow_\$ \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send y to the verifier.

$V \to P$ Generate $b \leftarrow_\$ \{0,1\}$ and send it to the prover.

$P \to V$ Compute $z \leftarrow uw^b$ and send it to the verifier. The verifier accepts if $z^2 = yx^b \bmod n$, and otherwise rejects.

Solution: Complete, sound, and zero knowledge. For completeness, note that for a quadratic residue $x = w^2 \bmod n$ we have $z^2 = (uw^b)^2 = u^2(w^2)^b = yx^b \bmod n$, hence the verifier always accepts.

For soundness, note that for quadratic non-residue x there does not exist $w \in \mathbb{Z}_n^*$ such that $x = w^2 \bmod n$. Now consider the case $b = 1$ and assume $z^2 = yx \bmod n$ for some $y, z \in \mathbb{Z}_n^*$ of the (potentially cheating) prover's choice. Since x is a quadratic non-residue, this implies that also y must be a quadratic non-residue. Thus, if $b = 1$, then the prover can only succeed if they choose y to be a quadratic non-residue. On the other hand, if $b = 0$ and y is a quadratic non-residue, then the verifier will reject. Hence, in either case (y being a quadratic residue or non-residue), the prover succeeds with probability at most $1/2$, which proves soundness.

Finally, to show zero knowledge, we define a simulator as follows: The simulator can pick a random $B \leftarrow_\$ \{0,1\}$, and a random $z \leftarrow_\$ \mathbb{Z}_n^*$. It sets $y = z^2x^{-b}$. Then it sends y to the verifier V^*, which returns a bit b. If $B \neq b$ then the simulator restarts. Else, it outputs the simulated transcript (y, B, z). Note that $z^2 = yx^b$, i.e. this is indeed a verifying transcript. Clearly, in each run the simulator has to restart with probability $\frac{1}{2}$. Thus, the simulator needs to run V^* two times in expectation until the protocol terminates. The transcript produced by the simulator is identical to the verifier's view (y, b, z) in a real interaction, which shows that the protocol is zero knowledge.

Protocol 4.

$P \to V$ Generate $u \leftarrow_\$ \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send y to the verifier.

$V \to P$ Generate $b \leftarrow_\$ \{0,1\}$ and send it to the prover.

$P \to V$ Compute $z \leftarrow uw$ and send it to the verifier. The verifier accepts if $z^2 = yx^b \bmod n$ and otherwise rejects.

Solution: Not complete, since the prover will fail the verification check when $b = 0$ (and so the completeness bound is at most $\frac{1}{2}$).

2.2 Zero-Knowledge for NP with Cards.

Recall that a Hamiltonian cycle in an undirected graph is a cycle that visits each vertex exactly once. Karp [Kar72] proved that the problem of deciding whether a Hamiltonian cycle exists in a graph is NP-complete.

Suppose that the prover and verifier are given a deck of cards which are either red or black on one face and all indistinguishable on the other face. Assuming that there are as many cards of each kind as necessary, design a zero-knowledge protocol for proving that a graph has a Hamiltonian cycle, in which the prover only uses the cards and the verifier only an unbiased coin.[2]

Hint: Use adjacency matrices.

Solution: Suppose G is a graph with n vertices. Recall that in the $n \times n$ adjacency matrix corresponding to G, we have the entry $e_{i,j} = 1$ ($i, j \in \{1, \ldots, n\}$) in the ith row and jth column if and only if there is an edge between the ith and jth vertices in G;[3] otherwise, $e_{i,j} = 0$. The prover can now use the playing cards to commit to edges and "non-edges" in G by committing to 1s and 0s respectively across all n^2 entries in the adjacency matrix by laying either a red or black card face down, and run the following protocol:

$P \to V$: P randomly selects a permutation $\phi : G \to G$, then computes $\phi(G)$, which in this case denotes the permutation under ϕ of the adjacency matrix of G. Then, P commits to $\phi(G)$ as $c_{\phi(G)} = (c_{i,j})_{i,j}$, where $c_{i,j}$ is a commitment (i.e. a red or black card face down) to the (i, j)'th edge in $\phi(G)$. P also commits to the permutation ϕ (represented as a matrix) with $c_\phi = \text{Commit}(\phi)$. P sends those commitments over to the verifier.

$V \to P$: V generates $b \leftarrow_\$ \{0, 1\}$ and sends it to the prover.

$P \to V$: If $b = 0$, then P sends over $\phi(G)$ and ϕ[4], which V can verify by opening the commitments (flipping all the cards face up) and use to check that the received adjacency matrix is a genuine permutation of the adjacency matrix of G. If $b = 1$, then P shows a Hamiltonian cycle in $\phi(G)$, i.e. P applies the permutation ϕ to the Hamiltonian cycle H in G to obtain a Hamiltonian cycle $\phi(H)$ in $\phi(G)$ and sends only the edges in $\phi(H)$[5] to V. The verifier checks whether $\phi(H)$ is consistent with the commitment c_ϕ, i.e. for each edge (i, j) in $\phi(H)$ the verifier V flips the corresponding card $c_{i,j}$ and checks whether it indeed commits to red, and accepts if the received edges form a Hamiltonian cycle in $\phi(G)$.

Now we give a sketch on why the above protocol is sound and zero-knowledge (completeness can be shown in a straightforward manner). Regarding soundness, we claim that if the input graph G does not have a Hamiltonian cycle, then any cheating prover P^* should make the (honest) verifier V accept with probability at-most $1/2$. Conditioning on the first message (i.e., commitments $\bar{c} = (c_{\phi(G)}, c_\phi)$) that P^* sends to V, if P^* manages to make V accept with probability greater than $1/2$, it means that P^* successfully responds to *both* challenges of V (i.e., $b = 0$ and $b = 1$) w.r.t. the *same* first message $\bar{c}$. In other words, (1) $c_{\phi(G)}$ is indeed a commitment to a valid permutation of G ($b = 0$), and (2) $c_{\phi(G)}$ contains a permuted Hamiltonian cycle ($b = 1$). Combining the observations of (1), (2),

[2] More precisely, we assume a setting where the verifier can only flip a card (i.e., turn from face down to face up) given the prover's permission. This could be implemented by locked cardboxes, i.e. the verifier receives face down cards (the cards are in the posession of the verifier and cannot be changed by the prover), but in order to flip a card the verifier has to use a key provided by the prover.

[3] according to a pre-specified ordering of the n vertices in G

[4] In an implementation with locked boxes, by sending $\phi(G)$ we mean that P also sends the keys that allow V to flip the cards in $c_{\phi(G)}$; similarly for ϕ.

[5] I.e., in the implementation with locked boxes, P only sends keys for the cards associated to edges in $\phi(H)$.

we have that the original graph G does contain a Hamiltonian cycle, a contradiction. An important thing to note here is that to show soundness, we crucially rely on the *binding* property of the above card-based commitments which allows us to argue about the *same* underlying permuted graph within the commitments $\bar{c}$ (i.e., the committed values do not change) when we go from $b = 0$ to $b = 1$ (or vice-versa).

To show zero-knowledge, a simulator M (as usual) tries to guess the potentially dishonest verifier V^*'s challenge (i.e., bit b) in advance. M first chooses a uniformly random bit $B \leftarrow_\$ \{0, 1\}$. If $B = 0$, then M proceeds as the honest prover P by committing to a valid permutation of G (along with committing to the corresponding permutation). If $B = 1$, then denoting n to be the number of vertices in G, M generates a Hamiltonian cycle w.r.t. n vertices uniformly at random by itself and commits to the adjacency matrix corresponding to the graph with n vertices that *only* contains this Hamiltonian cycle (and no other edges); M also commits to an arbitrary permutation on n vertices. When M sends these commitments to V^* and obtains a challenge bit b, M restarts the simulation if $B \neq b$ (which happens with probability 1/2); otherwise, it proceeds as the honest P and opens the commitments accordingly (i.e., opens all commitments if $b = 0$, or only opens the commitments corresponding to the Hamiltonian cycle if $b = 1$). Finally, the simulator M outputs the simulated transcript consisting of the commitments, the bit B, and the openings. We claim that the simulated transcript is identical to the view of V^* in a real protocol with P: The only case we need to consider is when $b = 1$. Since we are permuting the graph G uniformly at random in the real protocol above, we're implicitly also permuting the underlying Hamiltonian cycle; hence when $b = 1$, the verifier V^* gets to see the opening to a *uniformly random* Hamiltonian cycle across n vertices. This is precisely what the simulator M does as well when $B(= b) = 1$. Again an important thing to note here is that we crucially rely on the *hiding* property of the card-based commitments, which allows M to also commit to arbitrary values (other than the cycle) when $B = 1$ that are not opened anyway.

2.3 Sudoku (*)

Sudoku is a game played on a 9×9 grid subdivided into nine 3×3 subgrids. Each game starts with just a few cells filled with numbers in $\{1, \ldots, 9\}$ and the rest are empty. The goal is to completely fill the grid with numbers in $\{1, \ldots, 9\}$ so that each row, column and subgrid contains all 9 digits. Note that the generalized problem on a $k^2 \times k^2$ board is NP-complete [YS03].

Suppose that for a given Sudoku instance, Peggy wants to prove to Victor that she knows the solution to the puzzle without disclosing any information beyond that fact. The following protocol from [GNPR07] using playing cards (seven decks of cards are needed in practice) allows her to do so. Peggy is assumed only to put a card on a cell if it holds the same value.

- For each cell, Peggy picks three cards, all with the number that is associated to this cell. Peggy places the three cards face down on each cell, except for those cells that are already filled, on which cards are placed face up.
- For each row and each cell in the row, Victor picks at random one of the three cards in the cell. Victor repeats the process for each column picking from the two remaining cards in each cell, and finally for each subgrid.
- For each row, column and subgrid, Peggy assembles in a packet the cards Victor chose and then shuffles each of the 27 packets independently. Peggy hands each packet to Victor.
- Victor looks at the cards in each packet and checks that the packet contains all the numbers.

a) Show that the protocol is complete.

SOLUTION: Assuming both the prover and the verifier behave honestly, the verifier will end the protocol with 27 packets containing the cards from 1 to 9 in order, and as such will always accept (perfect completeness).

b) Show that the protocol is zero-knowledge, i.e., design an efficient simulator.

SOLUTION: The simulator works as follows. In the setup step it uses *arbitrary* cards to fill the cells unknown to the verifier. After the verifier selects the cards for the packets the simulator takes them and shuffles them, but before giving them back it swaps each packet with a randomly shuffled pack of cards, in which each card appears once. Now note that the simulator has an "extra ability" to swap packets, that an honest prover does not have. However this does not contradict the soundness of our protocol. In cryptographic zero-knowledge protocols in the literature (some of which we will be seeing in later parts of the course), usually the simulator has an advantage over the prover – e.g., in terms of "rewinding" the verifier, possessing some trapdoor information, etc. So this "swapping packets" advantage replaces the ability of the simulator to rewind the verifier. As explained in [GNPR07], an appropriate analogy is that of editing a movie (first suggested in [QQQ+90]). When making a movie of the proof, one can swap the cards and edit the movie so that this action is not noticeable. The final version of the movie is then indistinguishable from what one would see in a real execution.

(Alternative to this solution one can also construct a simulator that guesses Victor's choices in advance and then distributes cards accordingly, as was done in the previous exercises. The probability to guess correctly is at least 6^{81}, so the simulator would have to rerun the experiment an expected number 6^{81} times. While 6^{81} is still a constant, it's questionable whether this runtime would still be considered efficient in practice, which is why we presented the above solution here.)

c) Follow the steps below to prove that the protocol is 1/9-sound.

1. Show that a cheating prover will have to cheat on at least two cells, say a and b, or get caught with probability 1.
   SOLUTION: We start by noticing that if the prover in any way changes the number of cards with each number, the verifier will be able to detect it with probability 1. This is because one packet will then have either duplicate or missing cards, which will be detected. If the prover cheats on a single cell, then it will necessarily change the number of cards and so will be detected. The only way it can remain undetected is if it cheats on at least two cells.
2. Suppose that Peggy cheats on exactly two cells, and she assigns the cards (x, x, y) to a and the cards (y, y, x) to b (it is otherwise even harder for Peggy to cheat). The three possible configurations for how a and b are positioned relative to each other are listed below. Show that in any case Peggy will get caught with probability at least 8/9.

   (a) a, b are not in the same row, column or subgrid.

   (b) a, b are in the same row, column or subgrid (exactly one of them).

   (c) a, b are in the same row (or column) and the same subgrid.

   HINT: Assume that Peggy has already allocated the cards to all cells but a and b, and determine the probability that she assigns the correct cards to these cells.
   SOLUTION: We first note that we can derive a looser soundness bound of $\frac{1}{3}$ with a quite simple argument. Assume that the Sudoku has no solution. Then, the prover cannot place three cards with the same value in each cell, or he will get caught. Suppose then that he places three cards, not all of the same value, on a cell a. This means that in the cell a value, say y, will be different from all others.

Suppose that for all other cells, the verifier has already assigned the cards to rows, columns and subgrids. Then, in order for the prover to win, it needs there to be exactly one row/column/subgrid that needs y to complete its set. The probability that the verifier assigns that row/column/subgrid to the cell a is then $\frac{1}{3}$.
Next, we refine this argument to get the claimed bound. We just handle the case when a, b are in the same row, column or subgrid (case (b)), the others follow similarly with a bit of care for the possible arising cases. Suppose that the verifier has assigned every card to a packet except for the cards of cell a and b. Then there are six packets that are still missing a card, 2 for row, 2 for column and 2 for subgrid. Each of these packets can have only one value that will yield a complete set, since it cannot be missing both x and y. Thus, the only way that the prover will not be caught is if the verifier assigns x to the rows/columns/subgrids that need an x, and y to the corresponding ones. This happens with probability at most $\frac{1}{9}$

References

[GNPR07] Ronen Gradwohl, Moni Naor, Benny Pinkas, and Guy N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. In *Fun with Algorithms*, pages 166–182, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Kar72] Richard M. Karp. *Reducibility among Combinatorial Problems.* 1972.

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2021.

[QQQ+90] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 628–631, New York, NY, 1990. Springer New York.

[YS03] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 86-A(5):1052–1060, 2003.

ETH Zurich, Department of Computer Science Dr. Jonathan Bootle
HS 2023 Karen Klein, Varun Maram, Antonio Merino Gallardo

Zero-Knowledge Proofs
Exercise 2

Note: you may want to read the unpresented slides on the interactive proof for graph isomorphism from Lecture 1 before attempting these questions.

2.1 Quadratic Residues

Let n be a positive integer. An element $x \in \mathbb{Z}_n^*$ is a *quadratic residue* modulo n if there exists $w \in \mathbb{Z}_n^*$ such that $x = w^2 \bmod n$. If n is composite, it is believed to be computationally hard to decide whether a random $x \in \mathbb{Z}_n^*$ is a quadratic residue modulo n without knowing the factorization of n.[1]

Consider the problem of proving that a value $x \in \mathbb{Z}_n^*$ is a quadratic residue modulo an integer n. Formally, the prover and verifier are given input (n, x) such that there exists w satisfying $x = w^2 \bmod n$.

For each of the following protocols, say whether it is complete, sound and zero-knowledge, and justify your answer.

Protocol 1.

$P \to V$ Generate $u \leftarrow_\$ \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send (u, y) to the verifier.

$V \to P$ Generate $b \leftarrow_\$ \{0, 1\}$ and send it to the prover.

$P \to V$ Compute $z \leftarrow uw^b$ and send it to the verifier. The verifier accepts if $z^2 = yx^b \bmod n$ and otherwise rejects.

Solution: Not zero knowledge, since the verifier can send back $b = 1$, receive $z = uw$ and compute $u^{-1}z = w$ using the u received in the first step to obtain the secret.

Formally, assume the protocol was zero knowledge. Then for every (potentially malicious) verifier V^* there exists an efficient simulator S that for every quadratic residue x can generate a transcript $((u, y), b, z)$ that is distributed equally to the transcript in the view of V^* in an execution of the protocol with an honest prover P. Now consider a cheating verifier V^* that sets $b = 1$, instead of sampling b uniformly at random. Then the simulator S on input x, n needs to generate $u, y, z \in \mathbb{Z}_n^*$ such that $u^2 = y$ and $z^2 = yx$. But if such a simulator exists, we can construct an efficient algorithm A that can decide quardatic residuosity: A first runs S on input $(V^*, (x, n))$. If x is a quadratic residue then A receives from S values $u, y, z \in \mathbb{Z}_n^*$ such that $u^2 = y$ and $z^2 = yx$. From those values one can compute a square root of x as $w = u^{-1}z$; thus A holds a proof for x being a quadratic residue. On the other hand, if x is a quadratic non-residue, then S can behave arbitrarily, but clearly cannot come up with values $u, y, z \in \mathbb{Z}_n^*$ such that $u^2 = y$ and $z^2 = yx$, since such values do not even exist in this case. (To see this, note that $z^2 = u^2x$ implies that

[1] Here, by "random x" we mean uniform among the elements x in $\mathbb{Z}_n^*$ for which $\mathcal{J}_n(x) = 1$, where $\mathcal{J}_n$ denotes the Jacobi symbol for composite n. (See e.g. [KL21, Section 15.4.3] for reference.)

x is a quadratic residue.) Hence, if S fails, then A deduces that x is a quadratic non-residue. Thus, assuming that quadratic residuosity is hard, the above protocol is not zero knowledge.

Protocol 2.

$P \to V$ Generate $u \leftarrow_{\$} \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send y to the verifier.

$V \to P$ Set $b \leftarrow_{\$} 0$ and send it to the prover.

$P \to V$ Compute $z \leftarrow uw^b$ and send it to the verifier. The verifier accepts if $z^2 = y \bmod n$ and otherwise rejects.

Solution: Not sound since a prover can just send a random u in the third round, indepentently of x, and thus fool the verifier.

Protocol 3.

$P \to V$ Generate $u \leftarrow_{\$} \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send y to the verifier.

$V \to P$ Generate $b \leftarrow_{\$} \{0, 1\}$ and send it to the prover.

$P \to V$ Compute $z \leftarrow uw^b$ and send it to the verifier. The verifier accepts if $z^2 = yx^b \bmod n$, and otherwise rejects.

Solution: Complete, sound, and zero knowledge. For completeness, note that for a quadratic residue $x = w^2 \bmod n$ we have $z^2 = (uw^b)^2 = u^2(w^2)^b = yx^b \bmod n$, hence the verifier always accepts.

For soundness, note that for quadratic non-residue x there does not exist $w \in \mathbb{Z}_n^*$ such that $x = w^2 \bmod n$. Now consider the case $b = 1$ and assume $z^2 = yx \bmod n$ for some $y, z \in \mathbb{Z}_n^*$ of the (potentially cheating) prover's choice. Since x is a quadratic non-residue, this implies that also y must be a quadratic non-residue. Thus, if $b = 1$, then the prover can only succeed if they choose y to be a quadratic non-residue. On the other hand, if $b = 0$ and y is a quadratic non-residue, then the verifier will reject. Hence, in either case (y being a quadratic residue or non-residue), the prover succeeds with probability at most $1/2$, which proves soundness.

Finally, to show zero knowledge, we define a simulator as follows: The simulator can pick a random $B \leftarrow_{\$} \{0, 1\}$, and a random $z \leftarrow_{\$} \mathbb{Z}_n^*$. It sets $y = z^2 x^{-b}$. Then it sends y to the verifier V^*, which returns a bit b. If $B \neq b$ then the simulator restarts. Else, it outputs the simulated transcript (y, B, z). Note that $z^2 = yx^b$, i.e. this is indeed a verifying transcript. Clearly, in each run the simulator has to restart with probability $\frac{1}{2}$. Thus, the simulator needs to run V^* two times in expectation until the protocol terminates. The transcript produced by the simulator is identical to the verifier's view (y, b, z) in a real interaction, which shows that the protocol is zero knowledge.

Protocol 4.

$P \to V$ Generate $u \leftarrow_{\$} \mathbb{Z}_n^*$, compute $y \leftarrow u^2$ and send y to the verifier.

$V \to P$ Generate $b \leftarrow_{\$} \{0, 1\}$ and send it to the prover.

$P \to V$ Compute $z \leftarrow uw$ and send it to the verifier. The verifier accepts if $z^2 = yx^b \bmod n$ and otherwise rejects.

Solution: Not complete, since the prover will fail the verification check when $b = 0$ (and so the completeness bound is at most $\frac{1}{2}$).

2.2 Zero-Knowledge for NP with Cards.

Recall that a Hamiltonian cycle in an undirected graph is a cycle that visits each vertex exactly once. Karp [Kar72] proved that the problem of deciding whether a Hamiltonian cycle exists in a graph is NP-complete.

Suppose that the prover and verifier are given a deck of cards which are either red or black on one face and all indistinguishable on the other face. Assuming that there are as many cards of each kind as necessary, design a zero-knowledge protocol for proving that a graph has a Hamiltonian cycle, in which the prover only uses the cards and the verifier only an unbiased coin.[2]

Hint: Use adjacency matrices.

Solution: Suppose G is a graph with n vertices. Recall that in the $n \times n$ adjacency matrix corresponding to G, we have the entry $e_{i,j} = 1$ ($i, j \in \{1, \ldots, n\}$) in the ith row and jth column if and only if there is an edge between the ith and jth vertices in G;[3] otherwise, $e_{i,j} = 0$. The prover can now use the playing cards to commit to edges and "non-edges" in G by committing to 1s and 0s respectively across all n^2 entries in the adjacency matrix by laying either a red or black card face down, and run the following protocol:

$P \to V$: P randomly selects a permutation $\phi : G \to G$, then computes $\phi(G)$, which in this case denotes the permutation under ϕ of the adjacency matrix of G. Then, P commits to $\phi(G)$ as $c_{\phi(G)} = (c_{i,j})_{i,j}$, where $c_{i,j}$ is a commitment (i.e. a red or black card face down) to the (i, j)'th edge in $\phi(G)$. P also commits to the permutation ϕ (represented as a matrix) with $c_\phi = \text{Commit}(\phi)$. P sends those commitments over to the verifier.

$V \to P$: V generates $b \leftarrow_\$ \{0, 1\}$ and sends it to the prover.

$P \to V$: If $b = 0$, then P sends over $\phi(G)$ and ϕ[4], which V can verify by opening the commitments (flipping all the cards face up) and use to check that the received adjacency matrix is a genuine permutation of the adjacency matrix of G. If $b = 1$, then P shows a Hamiltonian cycle in $\phi(G)$, i.e. P applies the permutation ϕ to the Hamiltonian cycle H in G to obtain a Hamiltonian cycle $\phi(H)$ in $\phi(G)$ and sends only the edges in $\phi(H)$[5] to V. The verifier checks whether $\phi(H)$ is consistent with the commitment c_ϕ, i.e. for each edge (i, j) in $\phi(H)$ the verifier V flips the corresponding card $c_{i,j}$ and checks whether it indeed commits to red, and accepts if the received edges form a Hamiltonian cycle in $\phi(G)$.

Now we give a sketch on why the above protocol is sound and zero-knowledge (completeness can be shown in a straightforward manner). Regarding soundness, we claim that if the input graph G does not have a Hamiltonian cycle, then any cheating prover P^* should make the (honest) verifier V accept with probability at-most $1/2$. Conditioning on the first message (i.e., commitments $\bar{c} = (c_{\phi(G)}, c_\phi)$) that P^* sends to V, if P^* manages to make V accept with probability greater than $1/2$, it means that P^* successfully responds to *both* challenges of V (i.e., $b = 0$ and $b = 1$) w.r.t. the *same* first message $\bar{c}$. In other words, (1) $c_{\phi(G)}$ is indeed a commitment to a valid permutation of G ($b = 0$), and (2) $c_{\phi(G)}$ contains a permuted Hamiltonian cycle ($b = 1$). Combining the observations of (1), (2),

[2] More precisely, we assume a setting where the verifier can only flip a card (i.e., turn from face down to face up) given the prover's permission. This could be implemented by locked cardboxes, i.e. the verifier receives face down cards (the cards are in the posession of the verifier and cannot be changed by the prover), but in order to flip a card the verifier has to use a key provided by the prover.

[3] according to a pre-specified ordering of the n vertices in G

[4] In an implementation with locked boxes, by sending $\phi(G)$ we mean that P also sends the keys that allow V to flip the cards in $c_{\phi(G)}$; similarly for ϕ.

[5] I.e., in the implementation with locked boxes, P only sends keys for the cards associated to edges in $\phi(H)$.

we have that the original graph G does contain a Hamiltonian cycle, a contradiction. An important thing to note here is that to show soundness, we crucially rely on the *binding* property of the above card-based commitments which allows us to argue about the *same* underlying permuted graph within the commitments $\bar{c}$ (i.e., the committed values do not change) when we go from $b = 0$ to $b = 1$ (or vice-versa).

To show zero-knowledge, a simulator M (as usual) tries to guess the potentially dishonest verifier V^*'s challenge (i.e., bit b) in advance. M first chooses a uniformly random bit $B \leftarrow_{\$} \{0, 1\}$. If $B = 0$, then M proceeds as the honest prover P by committing to a valid permutation of G (along with committing to the corresponding permutation). If $B = 1$, then denoting n to be the number of vertices in G, M generates a Hamiltonian cycle w.r.t. n vertices uniformly at random by itself and commits to the adjacency matrix corresponding to the graph with n vertices that *only* contains this Hamiltonian cycle (and no other edges); M also commits to an arbitrary permutation on n vertices. When M sends these commitments to V^* and obtains a challenge bit b, M restarts the simulation if $B \neq b$ (which happens with probability 1/2); otherwise, it proceeds as the honest P and opens the commitments accordingly (i.e., opens all commitments if $b = 0$, or only opens the commitments corresponding to the Hamiltonian cycle if $b = 1$). Finally, the simulator M outputs the simulated transcript consisting of the commitments, the bit B, and the openings. We claim that the simulated transcript is identical to the view of V^* in a real protocol with P: The only case we need to consider is when $b = 1$. Since we are permuting the graph G uniformly at random in the real protocol above, we're implicitly also permuting the underlying Hamiltonian cycle; hence when $b = 1$, the verifier V^* gets to see the opening to a *uniformly random* Hamiltonian cycle across n vertices. This is precisely what the simulator M does as well when $B(= b) = 1$. Again an important thing to note here is that we crucially rely on the *hiding* property of the card-based commitments, which allows M to also commit to arbitrary values (other than the cycle) when $B = 1$ that are not opened anyway.

2.3 Sudoku (*)

Sudoku is a game played on a 9×9 grid subdivided into nine 3×3 subgrids. Each game starts with just a few cells filled with numbers in $\{1, \ldots, 9\}$ and the rest are empty. The goal is to completely fill the grid with numbers in $\{1, \ldots, 9\}$ so that each row, column and subgrid contains all 9 digits. Note that the generalized problem on a $k^2 \times k^2$ board is NP-complete [YS03].

Suppose that for a given Sudoku instance, Peggy wants to prove to Victor that she knows the solution to the puzzle without disclosing any information beyond that fact. The following protocol from [GNPR07] using playing cards (seven decks of cards are needed in practice) allows her to do so. Peggy is assumed only to put a card on a cell if it holds the same value.

- For each cell, Peggy picks three cards, all with the number that is associated to this cell. Peggy places the three cards face down on each cell, except for those cells that are already filled, on which cards are placed face up.
- For each row and each cell in the row, Victor picks at random one of the three cards in the cell. Victor repeats the process for each column picking from the two remaining cards in each cell, and finally for each subgrid.
- For each row, column and subgrid, Peggy assembles in a packet the cards Victor chose and then shuffles each of the 27 packets independently. Peggy hands each packet to Victor.
- Victor looks at the cards in each packet and checks that the packet contains all the numbers.

a) Show that the protocol is complete.

SOLUTION: Assuming both the prover and the verifier behave honestly, the verifier will end the protocol with 27 packets containing the cards from 1 to 9 in order, and as such will always accept (perfect completeness).

b) Show that the protocol is zero-knowledge, i.e., design an efficient simulator.

SOLUTION: The simulator works as follows. In the setup step it uses *arbitrary* cards to fill the cells unknown to the verifier. After the verifier selects the cards for the packets the simulator takes them and shuffles them, but before giving them back it swaps each packet with a randomly shuffled pack of cards, in which each card appears once. Now note that the simulator has an "extra ability" to swap packets, that an honest prover does not have. However this does not contradict the soundness of our protocol. In cryptographic zero-knowledge protocols in the literature (some of which we will be seeing in later parts of the course), usually the simulator has an advantage over the prover – e.g., in terms of "rewinding" the verifier, possessing some trapdoor information, etc. So this "swapping packets" advantage replaces the ability of the simulator to rewind the verifier. As explained in [GNPR07], an appropriate analogy is that of editing a movie (first suggested in [QQQ$^+$90]). When making a movie of the proof, one can swap the cards and edit the movie so that this action is not noticeable. The final version of the movie is then indistinguishable from what one would see in a real execution.

(Alternative to this solution one can also construct a simulator that guesses Victor's choices in advance and then distributes cards accordingly, as was done in the previous exercises. The probability to guess correctly is at least 6^{81}, so the simulator would have to rerun the experiment an expected number 6^{81} times. While 6^{81} is still a constant, it's questionable whether this runtime would still be considered efficient in practice, which is why we presented the above solution here.)

c) Follow the steps below to prove that the protocol is 1/9-sound.

1. Show that a cheating prover will have to cheat on at least two cells, say a and b, or get caught with probability 1.
   SOLUTION: We start by noticing that if the prover in any way changes the number of cards with each number, the verifier will be able to detect it with probability 1. This is because one packet will then have either duplicate or missing cards, which will be detected. If the prover cheats on a single cell, then it will necessarily change the number of cards and so will be detected. The only way it can remain undetected is if it cheats on at least two cells.
2. Suppose that Peggy cheats on exactly two cells, and she assigns the cards (x, x, y) to a and the cards (y, y, x) to b (it is otherwise even harder for Peggy to cheat). The three possible configurations for how a and b are positioned relative to each other are listed below. Show that in any case Peggy will get caught with probability at least 8/9.
   (a) a, b are not in the same row, column or subgrid.
   (b) a, b are in the same row, column or subgrid (exactly one of them).
   (c) a, b are in the same row (or column) and the same subgrid.
   HINT: Assume that Peggy has already allocated the cards to all cells but a and b, and determine the probability that she assigns the correct cards to these cells.
   SOLUTION: We first note that we can derive a looser soundness bound of $\frac{1}{3}$ with a quite simple argument. Assume that the Sudoku has no solution. Then, the prover cannot place three cards with the same value in each cell, or he will get caught. Suppose then that he places three cards, not all of the same value, on a cell a. This means that in the cell a value, say y, will be different from all others.

Suppose that for all other cells, the verifier has already assigned the cards to rows, columns and subgrids. Then, in order for the prover to win, it needs there to be exactly one row/column/subgrid that needs y to complete its set. The probability that the verifier assigns that row/column/subgrid to the cell a is then $\frac{1}{3}$.
Next, we refine this argument to get the claimed bound. We just handle the case when a, b are in the same row, column or subgrid (case (b)), the others follow similarly with a bit of care for the possible arising cases. Suppose that the verifier has assigned every card to a packet except for the cards of cell a and b. Then there are six packets that are still missing a card, 2 for row, 2 for column and 2 for subgrid. Each of these packets can have only one value that will yield a complete set, since it cannot be missing both x and y. Thus, the only way that the prover will not be caught is if the verifier assigns x to the rows/columns/subgrids that need an x, and y to the corresponding ones. This happens with probability at most $\frac{1}{9}$

References

[GNPR07] Ronen Gradwohl, Moni Naor, Benny Pinkas, and Guy N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. In *Fun with Algorithms*, pages 166–182, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Kar72] Richard M. Karp. *Reducibility among Combinatorial Problems.* 1972.

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2021.

[QQQ$^+$90] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 628–631, New York, NY, 1990. Springer New York.

[YS03] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 86-A(5):1052–1060, 2003.