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

Zero-Knowledge Proofs
Exercise 3

3.1 Definitions of Interactive Proofs

a) Show that the constants 3/4 and 1/2 in the completeness and soundness definitions for the class **IP** are arbitrary, i.e., that any other $0 < p < q \leq 1$ lead to an equivalent definition of **IP**.

HINT: Given an interactive proof (P, V), build an interactive proof (P', V') with completeness and soundness parameters q and p. You may use Hoeffding's inequalities: let $X_1, \ldots, X_n$ be i.i.d. Bernoulli random variables with parameter μ and $\bar{X} := (\sum_i X_i)/n$. For all $\varepsilon > 0$,

$$\Pr\left[\bar{X} \leq \mu - \varepsilon\right] \leq \exp(-2n\varepsilon^2) \text{ and}$$
$$\Pr\left[\bar{X} \geq \mu + \varepsilon\right] \leq \exp(-2n\varepsilon^2).$$

Solution: For $n \in \mathbb{Z}_{>0}$, let (P_n, V_n) be the interactive protocol defined as follows: on common input x, (P, V) is repeated n times and V_n accepts if and only if V accepts at least $5n/8$ times. (Note that $5/8 = (1/2 + 3/4)/2$.)

For $i \in [\![n]\!]$, let X_i be the Bernoulli random variable that indicates whether V accepts in the ith repetition.

Let $(Y_i)_{i=1}^n$ be i.i.d. Bernoulli random variables with $\Pr[Y_i = 1] = 3/4$ for each i. Let $\mu_Y := \Pr[Y_i = 1]$, and $\bar{Y} := (\sum_i Y_i)/n$. Note that $\mu_Y = \mathbb{E}[\bar{Y}]$.

Let $(Z_i)_{i=1}^n$ be i.i.d. Bernoulli random variables with $\Pr[Z_i = 1] = 1/2$ for each i. Let $\mu_Z := \Pr[Z_i = 1]$, and $\bar{Z} := (\sum_i Z_i)/n$, so that $\mu_Z = \mathbb{E}[\bar{Z}]$.

For $x \in L$, $\Pr[X_i = 1] \geq \Pr[Y_i = 1] = 3/4$ for each i, and

$$\begin{aligned}
\Pr[V_n = 0] &\leq \Pr[\textstyle\sum X_i < 5n/8] \\
&\leq \Pr[\textstyle\sum Y_i < 5n/8] \\
&\leq \Pr[\bar{Y} \leq 3/4 - 1/8] \\
&\leq \Pr[\bar{Y} \leq \mu_Y - 1/8] \\
&\leq \exp(-n/32).
\end{aligned}$$

For $x \notin L$, $\Pr[X_i = 1] \leq \Pr[Z_i = 1] = 1/2$ for each i, and

$$\begin{aligned}
\Pr[V_n = 1] &\leq \Pr[\textstyle\sum X_i \geq 5n/8] \\
&\leq \Pr[\textstyle\sum Z_i \geq 5n/8] \\
&\leq \Pr[\bar{Z} \geq 1/2 + 1/8] \\
&\leq \Pr[\bar{Z} \geq \mu_Z + 1/8] \\
&\leq \exp(-n/32).
\end{aligned}$$

If $n \geq 32\log(1/\min(p,(1-q)))$, these upper bounds are respectively lower than $(1-q)$ and p, and as long as $1/\min(p,(1-q)) = O(\exp \operatorname{poly}(\lambda))$, integer n is polynomial in λ. It then suffices to set (P', V') as (P_n, V_n) for $n := \lceil 32\log(1/\min(p,(1-q)))\rceil$.

b) Show that a language L for which there exists an interactive proof (P, V) with V deterministic is in **NP**.

Solution: If $x \in L$ then completeness implies that there is a $3/4$ fraction of the prover randomness that makes the verifier accept. Therefore, there exists a transcript of the interaction between P and V that is accepting, and it then serves as an **NP** witness. Conversely, if $x \notin L$, then there cannot exist a transcript that makes V accept as otherwise an algorithm P^* that runs P on every possible choice of prover randomness would recover it and could simply run P on the corresponding choice of randomness and always make V accept (thereby contradicting the soundness of (P, V)). $\square$

c) Show that for every interactive proof (P, V), with a probabilistic prover, there exists an interactive proof (P', V) for the same language L such that P' is deterministic.

Hint: Recall that the prover is computationally unbounded.

Solution: Given an interactive proof (P, V), consider an interactive proof (P', V) in which P' proceeds as follows: for common input x, algorithm P' computes the randomness $r_{\max}$ for P that maximises the probability that V accepts and then runs P on $r_{\max}$. P' can determine $r_{\max}$ by running the protocol for every possible choice of randomness for both P and V.

Let $p(x)$ denote the probability that V accepts in an interaction with P, and $p(x, r)$ the probability that V accepts in an interaction with P on randomness r. Let R be a random variable with uniform distribution over the randomness set of the prover. Then, if $x \in L$,

$$3/4 \leq p(x) \leq \sum_r p(x, r) \Pr[R = r] \leq \sum_r p(x, r_{\max}) \Pr[R = r] \leq p(x, r_{\max}).$$

Conversely, if $x \notin L$ then since no prover P^* can make V accept with probability larger than $1/2$ by the soundness property of (P, V), protocol (P', V) is sound. $\square$

d) Consider a language L for which there exists an interactive proof (P, V) such that V always rejects when the input is not in L. Show that L is in **NP**.

Solution: If $x \in L$, completeness implies that there exists an accepting transcript of an interaction between P and V which then serves as a witness for x. If $x \notin L$ then by assumption V always rejects no matter the computation of a potentially malicious prover and no such accepting transcript exists. In other words, V is a polynomial-time verifier for L, which implies that $L \in \mathbf{NP}$. $\square$

3.2 Commitment Schemes

a) Explain why a commitment scheme cannot be both perfectly hiding and perfectly binding.

Solution: For parameter pp and messages x, x', let $\mathsf{Commit}(pp, x) = (c_x, d_x)$ and $\mathsf{Commit}(pp, x') = (c_{x'}, d_{x'})$. For a commitment scheme to be perfectly hiding, the distributions of c_x and $c_{x'}$ must be the same for all parameters pp and committed messages x and x'. For a commitment scheme to be perfectly binding, the supports of c_x and $c_{x'}$ must be disjoint as soon as $x \neq x'$. It follows that a commitment scheme cannot be both perfectly hiding and perfectly binding.

b) The RSA assumption is that no efficient algorithm can compute a value x such that $g = x^e \bmod N$ given an RSA public key (N, e) and a value $g \in \{0, \ldots, N-1\}$.

Consider the following commitment scheme [Fis01]:

$\mathsf{Setup}\left(1^\lambda\right) \to pp$**:** Choose an RSA modulus $N = pq$ with $2^{\lambda-1} \leq N < 2^\lambda$, a prime $e \geq 2^\lambda$, and let $g := x^e \bmod N$ for $x \leftarrow \mathbb{Z}_N^*$. Return public parameters $pp = (N, e, g)$.

$\mathsf{Commit}(pp, m, r) \to (c, d)$: To commit to a message $m \in \mathbb{Z}_e$, compute $c \leftarrow g^m r^e \bmod N$ for a random $r \leftarrow_\$ \mathbb{Z}_N^*$, set $d \leftarrow r$ and return (c, d).

$\mathsf{Verify}(pp, c, d, m) \to b \in \{0, 1\}$: Return 1 if $c = g^m d^e \bmod N$ and 0 otherwise.

Show that this commitment scheme is perfectly hiding and computationally binding under the RSA assumption.

Show that the commitment scheme is also *equivocable*: there is a trapdoor which makes it possible to open a commitment to any message.

Solution:

Hiding Property. Since e is a prime greater than N, it is coprime with $\varphi(N)$, and there exists an integer f such that $ef = 1 \bmod \varphi(N)$ by Bézout's identity. Therefore, the map $x \mapsto x^e \bmod N$ is a permutation of $\mathbb{Z}_N^*$. It follows that a commitment to any $m \in \mathbb{Z}_e$ is uniformly distributed over $\mathbb{Z}_N^*$ and the scheme is thus perfectly hiding.

Binding Property. As for the binding property, note that for pairs (m, d) and (m', d') in $\mathbb{Z}_e \times \mathbb{Z}_N^*$ such that $m \neq m'$, the equality $g^m d^e = g^{m'} (d')^e \bmod N$ implies that $g^{m'-m} = \left(d' d^{-1}\right)^e \bmod N$. Since $m' - m \neq 0 \bmod e$, there exist integers u and v such that $u(m' - m) + ve = 1$ by Bézout's identity, and then $g = \left(\left(d' d^{-1}\right)^u g^v\right)^e \bmod N$, i.e., it is possible to compute an eth root of g. To reduce the binding property of the scheme to the RSA assumption, it then suffices to set the public parameters of the scheme to an instance (N, e, g) of the RSA problem upon receiving it.

(Note that in the setup, we have $g \in \mathbb{Z}_N^*$. However if we have $g \notin \mathbb{Z}_N^*$ in the RSA problem instance, then we can simply recover the prime factors of N by computing $\mathsf{gcd}(g, N)$ and solve the RSA problem in a straightforward way.)

Equivocability. A trapdoor for this scheme is an integer f such that $ef = 1 \bmod \varphi(N)$, i.e., a piece of information that allows to efficiently compute e-th roots. Indeed, a value $c \in \mathbb{Z}_N^*$ can be opened to any value $m \in Z_e$ if and only if a value $d \in \mathbb{Z}_N^*$ such that $c = g^m d^e \bmod N$ can be computed. To do so, it suffices to set $d := \left(cg^{-m}\right)^f \bmod N$.

3.3 Implementing ZKP for Graph Isomorphism (*)

Note: This exercise is not examinable. But the intention is to get students to practice implementing ZKPs from the lectures.

Implement the (perfect) zero-knowledge proof for graph isomorphism (GI) presented in Lecture 2; you can use your favorite programming language and libraries.

To be more precise, your code should implement the following helper functions:

- instance_generator
  - Input: n, an integer
  - Output: $((G_0, G_1), \pi)$, where G_1 is a random graph with n vertices, π is a random $n \times n$ permutation matrix and $G_0 = \pi(G_1)$. (You can use any valid representation of graphs – e.g., as adjacency matrices, members of a "graph" class in certain libraries, etc.)
- first_prover_msg
  - Input: $(n, (G_0, G_1), \pi)$, where n is an integer, (G_0, G_1) is a pair of graphs with n vertices and π is an $n \times n$ permutation matrix.
  - Output: (H, σ), where H is a graph with n vertices and σ is an $n \times n$ permutation matrix. The computation of (H, σ) follows from the prover's first message in the GI ZKP from Lecture 2.
- verifier_msg

  - Input: None.
  - Output: b, a bit. The computation of b follows from the verifier's first message in the GI ZKP from Lecture 2.

- second_prover_msg
  - Input: $(n, (G_0, G_1), \pi, \sigma, H, b)$, where $n, (G_0, G_1), \pi$ are as defined w.r.t. the input of first_prover_msg, σ is an $n \times n$ permutation matrix, H is a graph with n vertices and b is a bit.
  - Output: τ, an $n \times n$ permutation matrix. The computation of τ follows from the prover's second message in the GI ZKP from Lecture 2.

- verifier_checks
  - Input: $(n, (G_0, G_1), H, b, \tau)$, where $(n, (G_0, G_1), H, b)$ are as defined w.r.t. the input of second_prover_msg and τ is an $n \times n$ permutation matrix.
  - Output: b, a bit. The computation of b follows from the verifier's final output in the GI ZKP from Lecture 2. (Here you can have $b = 0$ and $b = 1$ to be synonymous with the verifier "rejecting" and "accepting" respectively.)

Your code should then take as input an integer n from a user and – using the above helper functions – display a pair of graphs with n vertices, the messages exchanged between an honest prover and an honest verifer in the GI ZKP on the aforementioned pair of graphs, and the verifier's final output.

Bonus: Can you demonstrate soundness of the GI ZKP using your implementation?

Solution: You can find an example implementation in SageMath[1] on Moodle in the form of a Jupyter Notebook[2] titled "Graph Isomorphism Implementation.ipynb".

We currently demonstrate soundness by using a "false_instance_generator(n)" function which outputs two independent random graphs G_0, G_1 on n vertices and a random $n \times n$ permutation matrix π; the graphs G_0 and G_1 are non-isomorphic with high probability (i.e., *not* in the language being considered). In our demonstration, we are using an honest prover, but we encourage you to "code" your own malicious prover! (You can do this by creating arbitrary "first prover message" and "second prover message" functions.)

References

[Fis01] Marc Fischlin. *Trapdoor commitment schemes and their applications.* PhD thesis, Goethe University Frankfurt, Frankfurt am Main, Germany, 2001.

[1] `https://www.sagemath.org/`
[2] `https://jupyter.org/`

Lecture 3: Sigma Protocols

Zero-knowledge proofs

263-4665-00L

Lecturer: Jonathan Bootle

Last time

- ZKP for graph isomorphism
- Variations of zero-knowledge
- Variations of soundness

Course Outline (13 lectures)

1. **Introduction and definitions** ~2 lectures

2. **Sigma protocols** ~3 lectures

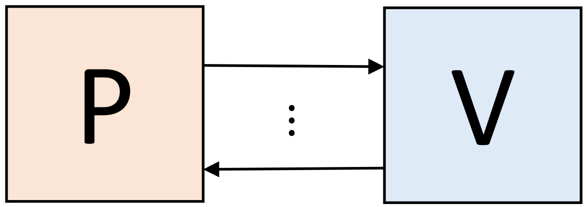
3. **ZK arguments with short proofs** ~4 lectures

4. **Non-interactive zero-knowledge** ~3 lectures

5. **Bonus material?** ~1 lecture

Efficiency targets

$\mathcal{L} \in \mathbf{NP}$



1. Introduction and definitions		**NP** proofs	$\mathrm{poly}(\lvert x \rvert)$ bits $\leftrightarrow$	$\mathrm{poly}(\lvert x \rvert)$ V-time	Not ZK
2. Sigma protocols	Practical, useful techniques		$\mathrm{poly}(\lvert x \rvert)$	$\mathrm{poly}(\lvert x \rvert)$	(SHV)ZK
3. ZK arguments with short proofs	Standard assumptions		$\mathrm{poly}\log(\lvert x \rvert)$	$\mathrm{poly}\log(\lvert x \rvert)$	(SHV)ZK
				Main targets	
4. Non-interactive zero-knowledge	Strong assumptions		$O(1)$	$O(1)$	ZK

5. Bonus material?

Agenda

- **Variations of soundness**
- Σ-protocols
- Commitment schemes
- Σ-protocol for an **NP**-complete problem
- Composition methods for Σ-protocols

Proofs of knowledge

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

$\exists$ 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

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

Public randomness and private randomness

Definition:

An IP (P, V) is *public coin* if V's messages are exactly its random inputs and nothing else. In this case, verifier messages are called *challenges*.

Goldwasser, Sipser 1986: Every language with an IP has a public coin IP.

Example: ZKP for GI

$P(G_0, G_1)$

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

$H \coloneqq \sigma(G_0)$

$\xrightarrow{H}$

$V(G_0, G_1)$

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

$\xleftarrow{b}$

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

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

$\xrightarrow{\tau}$

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

Non-example: IP for GNI

Not sound if b is leaked

$V(G_0, G_1)$

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

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

$H = \pi(G_b)$

$\xleftarrow{H}$

$P(G_0, G_1)$

Test $H \cong G_0$ or $H \cong G_1$

$\xrightarrow{B \text{ s.t. } H \cong G_B}$

Output $(B == b)$

Trees of transcripts

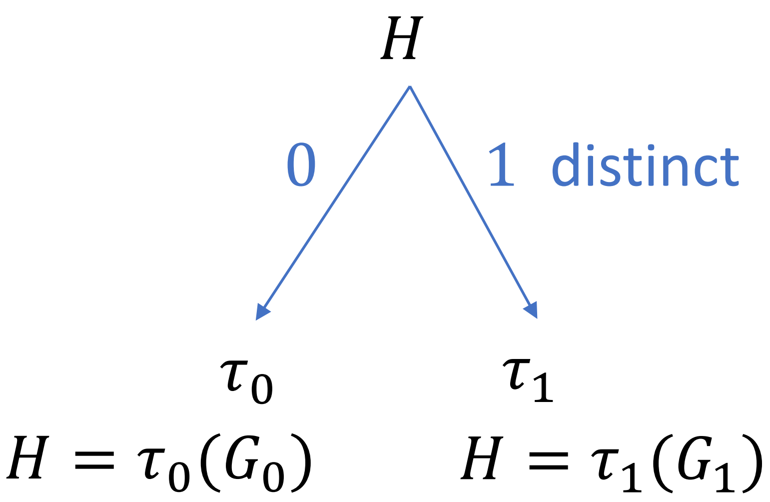
Definition:

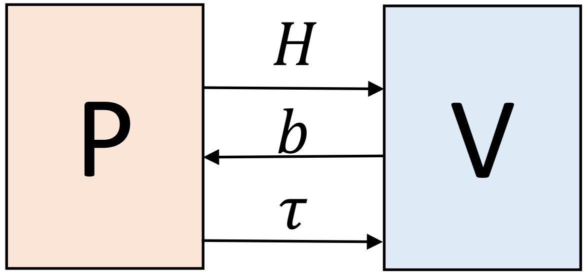
An $(n_1, \ldots, n_k)$-tree of transcripts for a $(2k + 1)$-move public-coin protocol is a set of $\prod_{i=1}^{k} n_i$ transcripts arranged in a tree such that:

- Vertices correspond to prover messages.
- Edges correspond to verifier challenges.
- Each node at depth i has n_i child edges labelled with distinct challenges.
- Each transcript corresponds to exactly one root-to-leaf path.
- The tree is *accepting* if the verifier would have accepted every transcript.

Example accepting trees

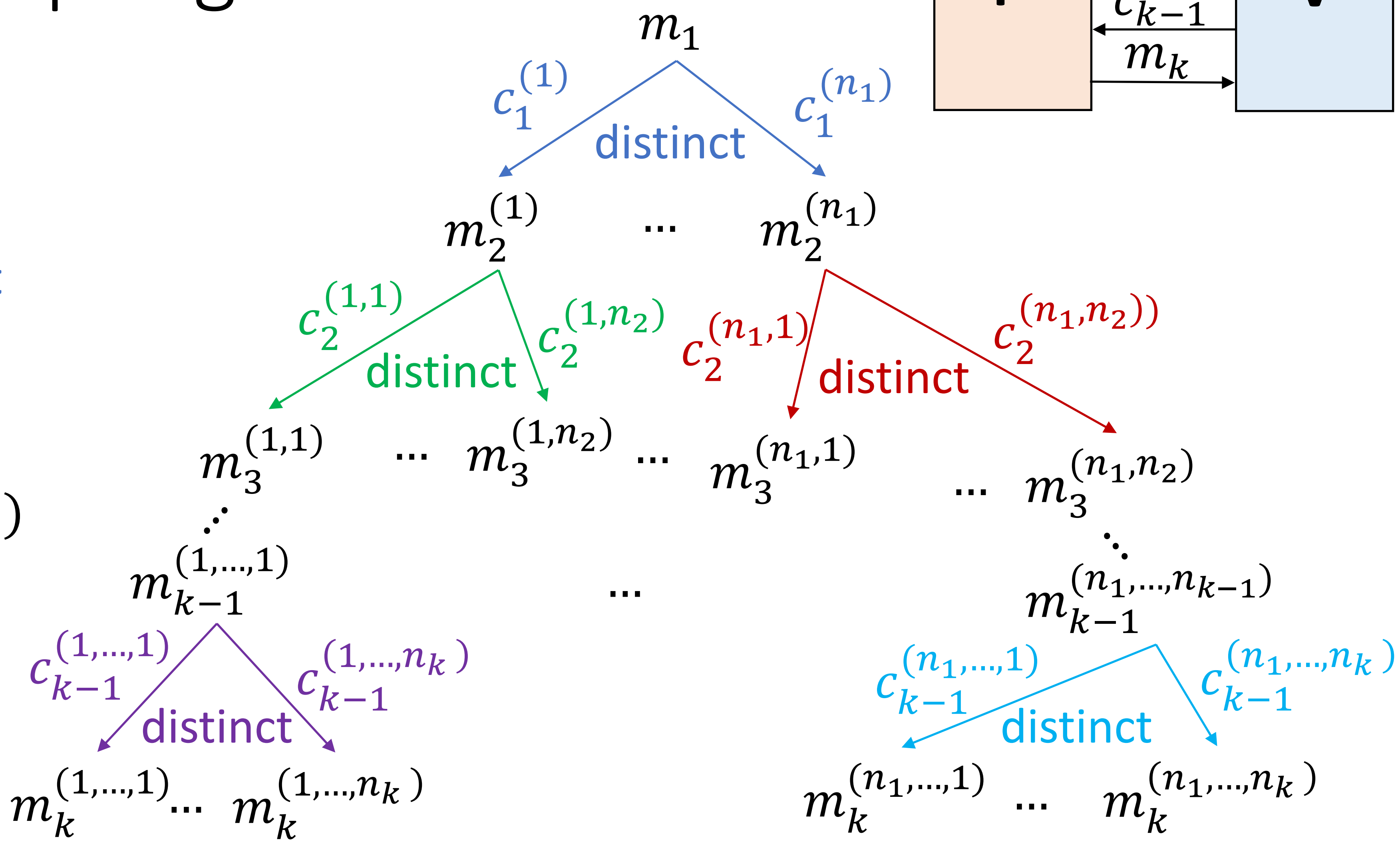
2-tree for GI

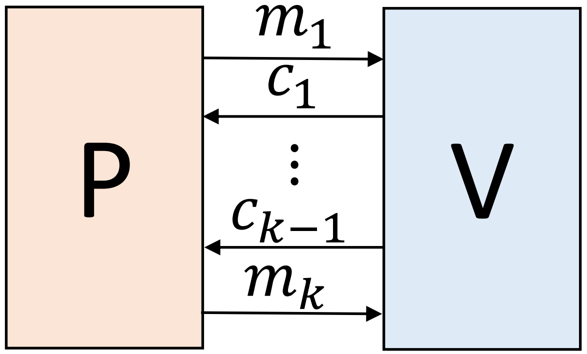




General

$(n_1, \ldots, n_k)$-tree





Special soundness

Usually easier to prove special soundness

Definition:

$(n_1, \dots, n_k)$-sound for short

A $(2k+1)$-move public coin protocol is $(n_1, \dots, n_k)$-*special sound* if $\exists$ an efficient *extractor* E that takes x and an $(n_1, \dots, n_k)$-tree of *accepting* transcripts for x and produces a witness w with $(x, w) \in \mathcal{R}$.

Theorem: (Attema, Cramer, Kohl 2021)

See paper for proofs

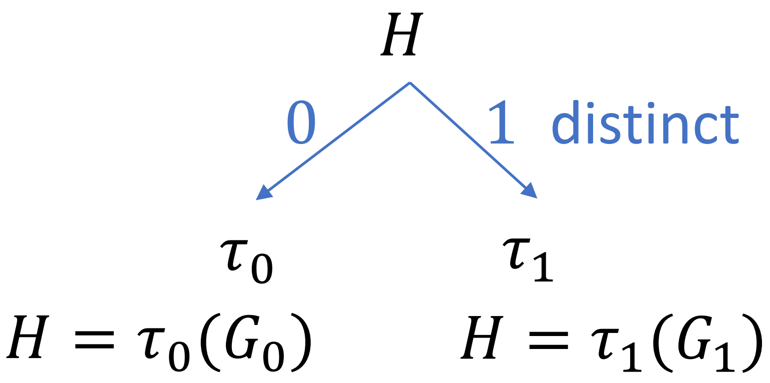
Let (P, V) be $(n_1, \dots, n_k)$-special sound with uniformly random verifier messages from a set of size N, and $\prod_{i=1}^{k} n_i$ be polynomially bounded in $|x|$. Then (P, V) is knowledge sound with knowledge error

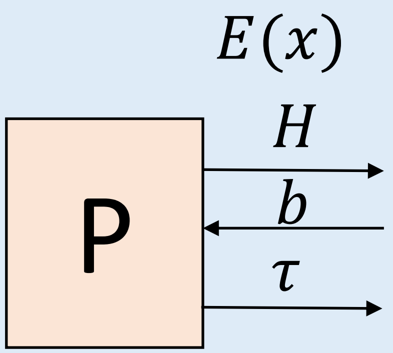
$n_i \leq N$

$$\kappa = \frac{N^k - \prod_{i=1}^{k}(N - n_i - 1)}{N^k} \leq \frac{\sum_{i=1}^{k}(n_i - 1)}{N}$$

Intuition behind proof

2-tree for GI

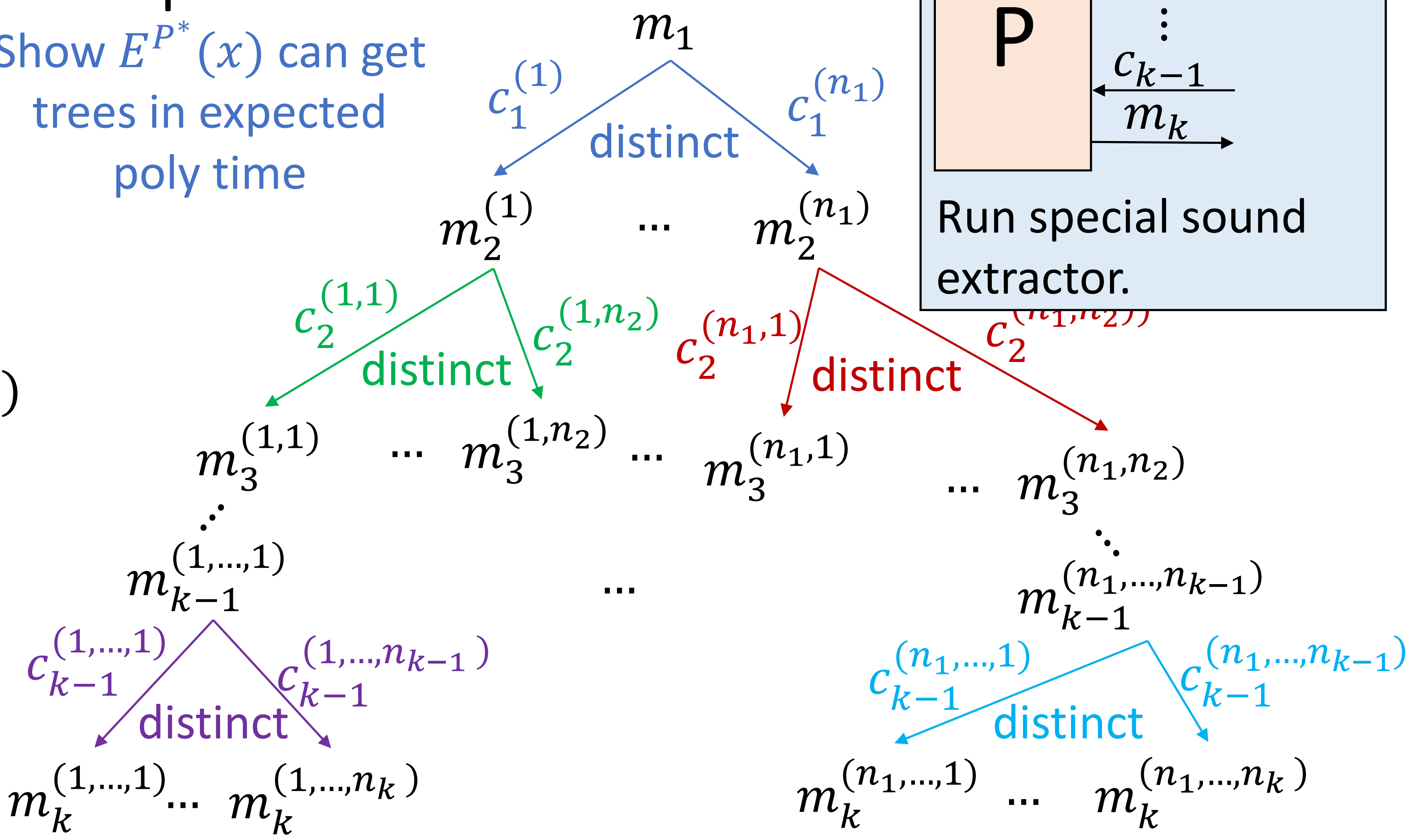


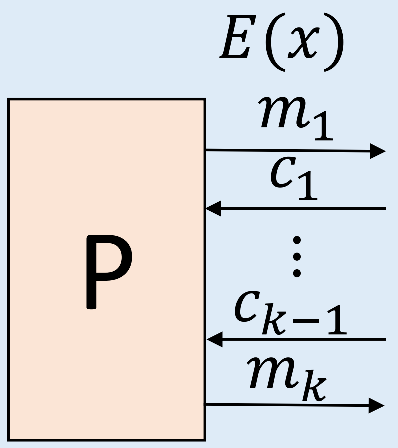


Run special sound extractor.

Show $E^{P^*}(x)$ can get trees in expected poly time

General $(n_1, \ldots, n_k)$-tree





Run special sound extractor.

Summary of variants

- Can't have perfect soundness and ZK together.

Short-term secrecy

	Soundness	ZK	Proof sizes
Proofs	Perfect/statistical	Computational	Not compressing
Arguments	Computational	Perfect/statistical	$\ll \lvert x\rvert, \lvert w\rvert$

Short window for cheating

Everlasting secrecy

- Knowledge soundness good for "trivial" languages.
- We will almost always prove special soundness and SHVZK.
- Rely on transformations for knowledge soundness and full ZK.

Agenda

- Variations of soundness ✓
- **Σ-protocols**
- Commitment schemes
- Σ-protocol for an **NP**-complete problem
- Composition methods for Σ-protocols

Σ-protocols

$P(x, w; \rho)$		$V(x; c)$
$a \leftarrow_\$ P_1(x, w; \rho)$	$\xrightarrow{a}$	
	$\xleftarrow{c}$	$c \leftarrow_\$ \mathcal{C}$
$z \leftarrow_\$ P_2(x, w, a, c; \rho)$	$\xrightarrow{z}$	Output $V(x, a, c, z)$

challenge space

Definition:

A Σ-protocol for a relation $\mathcal{R}$ is a 3-move, public-coin protocol satisfying

- Completeness with no errors $(x, w) \in \mathcal{R} \Rightarrow V$ accepts
- k-special soundness

accepting k-tree

$$\begin{array}{c} a \\ c_1 \swarrow \quad \searrow c_k \\ z_1 \quad \cdots \quad z_k \end{array} \xrightarrow{E} (x, w) \in \mathcal{R}$$

distinct $c_1 \ldots c_k$

$\Rightarrow$ soundness error $(k-1)/|\mathcal{C}|$

- SHZVK

$\exists$ efficient $S : \forall x \in \mathcal{L}, c \in \mathcal{C}, \mathrm{View}_V^P(x, c) \approx S(x, c)$

Can *try* to get full ZK similarly to GI if $\mathcal{C}$ is not too big

Examples:

GI, QR protocol from Sheet 2

statistical, computational variants

Nomenclature

Cramer, 1996, Modular Design of Secure, Yet Practical, Cryptographic Protocols

$P(x, w; \rho)$

$a \leftarrow_{\$} P_1(x, w; \rho)$

$V(x; c)$

x

a

$c \leftarrow_{\$} \mathcal{C}$

c

$z \leftarrow_{\$} P_2(x, w, a, c; \rho)$

z

Output

$V(x, a, c, z)$

Agenda

- Variations of soundness ✓
- Σ-protocols ✓
- **Commitment schemes**
- Σ-protocol for an **NP**-complete problem
- Composition methods for Σ-protocols

Commitment schemes

Cryptographic envelopes

Values can be sealed and revealed later

Values are hidden until revealed

Values can’t be changed

Syntax of commitment schemes

Definition:

A *commitment scheme* is a collection of 3 p.p.t. algorithms $(\mathrm{Setup}, \mathrm{Commit}, \mathrm{Verify})$ such that $\forall\lambda \in \mathbb{N}$,

- $\mathrm{Setup}(1^\lambda)$ outputs public parameters pp describing a message space $\mathfrak{M}$, randomness space $\mathfrak{R}$, decommitment space $\mathfrak{D}$ and commitment space $\mathfrak{C}$.

- $\mathrm{Commit}(pp, m \in \mathfrak{M}, r \leftarrow_\$ \mathfrak{R})$ outputs a pair $(c, d) \in \mathfrak{C} \times \mathfrak{D}$.
- $\mathrm{Verify}(pp, c \in \mathfrak{C}, d \in \mathfrak{D}, m \in \mathfrak{M})$ outputs a bit $b \in \{0,1\}$.

Signals whether m was stored inside c or not.

Correctness and security of commitments

Definition:

$\forall \lambda \in \mathbb{N}$, a commitment scheme $(\text{Setup}, \text{Commit}, \text{Verify})$ satisfies

- *Correctness* if $\forall pp \leftarrow_\$ \text{Setup}(1^\lambda), m \in \mathfrak{M}, r \in \mathfrak{R}$,

$$\text{Verify}(pp, Commit(pp, m, r), m) = 1$$

- *Perfect hiding* if $\forall A$,

Linked to ZK

$$\left\{\begin{array}{c} c_1 : pp \leftarrow_\$ \text{Setup}(1^\lambda) \\ (m_1, m_2) \leftarrow_\$ A(pp) \\ (c_1, d_1) \leftarrow_\$ \text{Commit}(pp, m_1, r_1) \end{array}\right\} = \left\{\begin{array}{c} c_2 : pp \leftarrow_\$ \text{Setup}(1^\lambda) \\ (m_1, m_2) \leftarrow_\$ A(pp) \\ (c_2, d_2) \leftarrow_\$ \text{Commit}(pp, m_2, r_2) \end{array}\right\}$$

- *Perfect binding* if $\forall A$,

A outputs in correct spaces

$$\Pr\left[\begin{array}{c} pp \leftarrow_\$ \text{Setup}(1^\lambda), (c, d_1, d_2, m_1, m_2) \leftarrow A(pp) \\ m_1 \neq m_2 \wedge \text{Verify}(pp, c, d_1, m_1) = \text{Verify}(pp, c, d_2, m_2) = 1 \end{array}\right] = 0$$

Linked to soundness

Correctness and security of commitments

Definition:

$\forall \lambda \in \mathbb{N}$, a commitment scheme $(\text{Setup}, \text{Commit}, \text{Verify})$ satisfies

- *Computational hiding* if $\forall$ efficient A,

$$\left\{\begin{matrix} c_1 : pp \leftarrow_\$ \text{Setup}(1^\lambda) \\ (m_1, m_2) \leftarrow_\$ A(pp) \\ (c_1, d_1) \leftarrow_\$ \text{Commit}(pp, m_1, r_1) \end{matrix}\right\} \approx_c \left\{\begin{matrix} c_2 : pp \leftarrow_\$ \text{Setup}(1^\lambda) \\ (m_1, m_2) \leftarrow_\$ A(pp) \\ (c_2, d_2) \leftarrow_\$ \text{Commit}(pp, m_2, r_2) \end{matrix}\right\}$$

- *Computational binding* if $\forall$ efficient A,

A outputs in correct spaces

$$\Pr\left[\begin{matrix} pp \leftarrow_\$ \text{Setup}(1^\lambda), (c, d_1, d_2, m_1, m_2) \leftarrow A(pp) \\ m_1 \neq m_2 \wedge \text{Verify}(pp, c, d_1, m_1) = \text{Verify}(pp, c, d_2, m_2) = 1 \end{matrix}\right] \leq negl(\lambda)$$

	Perfect hiding	Computational hiding
Perfect binding	NO	YES
Computational binding	YES	YES

Commitments from Elgamal encryption

Setup$(\mathbf{1}^{\lambda})$**:**

- Choose group $\mathbb{G}$ of prime order $p \approx 2^{\lambda}$.
- Sample $h \leftarrow_{\$} \mathbb{G}$, $s \leftarrow_{\$} \mathbb{Z}_p$ and compute $g = s \cdot h$.
- Output $pp \coloneqq (\mathbb{G}, g, h, p)$.

Computationally hiding assuming DDH

Commit$(\boldsymbol{pp}, \boldsymbol{m} \in \mathbb{G}, \boldsymbol{r} \leftarrow_{\$} \mathbb{Z}_{\boldsymbol{p}})$**:**

- Output $\big((c_1, c_2), d\big) \coloneqq \big((m + r \cdot g, r \cdot h), r\big)$.

Perfectly binding as s, decryption $c_1 - s \cdot c_2$ are unique

Verify$(\boldsymbol{pp}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{m})$**:**

- Output $(\text{Commit}(pp, m, d) == c)$.

Pedersen commitments

Setup $(\mathbf{1}^\lambda)$**:**

Choose group $\mathbb{G}$ of prime order $p \approx 2^\lambda$.

- Sample $h \leftarrow_\$ \mathbb{G}$, $s \leftarrow_\$ \mathbb{Z}_p$ and compute $g = s \cdot h$.
- Output $pp \coloneqq (\mathbb{G}, g, h, p)$.

Perfectly hiding as $r \cdot h$ is uniformly random in $\mathbb{G}$

Commit $(\boldsymbol{pp}, \boldsymbol{m} \in \mathbb{Z}_{\boldsymbol{p}}, \boldsymbol{r} \leftarrow_\$ \mathbb{Z}_{\boldsymbol{p}})$**:**

- Output $(c, d) \coloneqq (m \cdot g + r \cdot h, r)$.

Computationally binding or we break the DLOG assumption

Verify $(\boldsymbol{pp}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{m})$**:**

- Output $(\mathrm{Commit}(pp, m, d) == c)$.

Agenda

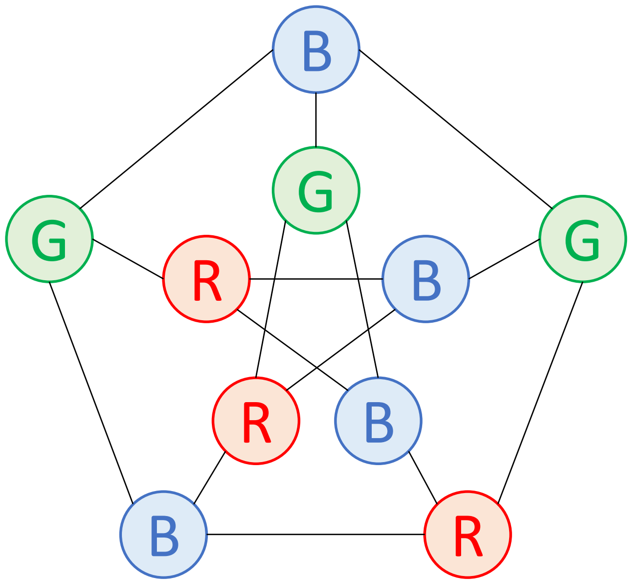
- Variations of soundness ✓
- Σ-protocols ✓
- Commitment schemes ✓
- **Σ-protocol for an NP-complete problem**
- Composition methods for Σ-protocols

Visual Σ-Protocol for Graph 3-Colouring

NP-complete: Convert any NP x into x_{G3C} in poly time

- $\mathcal{L}_{G3C} \coloneqq \{3\text{−colourable graphs } G = (V, E)\}$. $x \in \mathcal{L} \Leftrightarrow x_{G3C} \in \mathcal{L}_{G3C}$
- $\mathcal{R}_{G3C} \coloneqq \{(G, c) : G \in \mathcal{L}_{G3C}, c: V \to \{R, G, B\}, (u, v) \in E \Rightarrow c(u) \neq c(v)\}$.

P Efficiently checkable so $\mathcal{L}_{G3C} \in$ **NP** V



Petersen graph

Visual Σ-Protocol for Graph 3-Colouring

- $\mathcal{L}_{G3C} \coloneqq \{3-\text{colourable graphs } G = (V, E)\}$.
- $\mathcal{R}_{G3C} \coloneqq \{(G, c) : G \in \mathcal{L}_{G3C}, c: V \to \{R, G, B\}, (u, v) \in E \Rightarrow c(u) \neq c(v)\}$.

- See also https://web.mit.edu/~ezyang/Public/graph/svg.html

Proper Σ-Protocol for Graph 3-Colouring

- $\mathcal{R}_{G3C} \coloneqq \{(G, c) : G \in \mathcal{L}_{G3C}, c: V \to \{R, G, B\}, (u, v) \in E \Rightarrow c(u) \neq c(v)\}$.

$pp \leftarrow_\$ \text{Setup}(1^\lambda)$

$P(pp, G, c)$		$V(pp, G)$
$\sigma \leftarrow_\$ \text{Perm}(R, G, B)$ $col \coloneqq \sigma(c(V))$ $(c_u, d_u) \leftarrow_\$ Commit(pp, col_u)$	$\xrightarrow{(c_u)_{u \in V}}$	
	$\xleftarrow{(a, b)}$	$(a, b) \leftarrow_\$ E$
	$\xrightarrow{col_a, d_a \quad col_b, d_b}$	Output 1 iff $Verify(pp, c_a, d_a, col_a) = 1$ $Verify(pp, c_b, d_b, col_b) = 1$ $col_a \neq col_b$

Completeness, $|E|$-special soundness analysis

Completeness:

- If c is a 3-colouring then so is $\sigma \circ c$, and $\forall (a,b) \in E, col_a \neq col_b$.
- By the correctness of the commitment scheme, $Verify(pp, c_a, d_a, col_a) = 1$, and similarly for c_b.

$|E|$-special soundness:

commitments

edge challenges

colours and decommitments

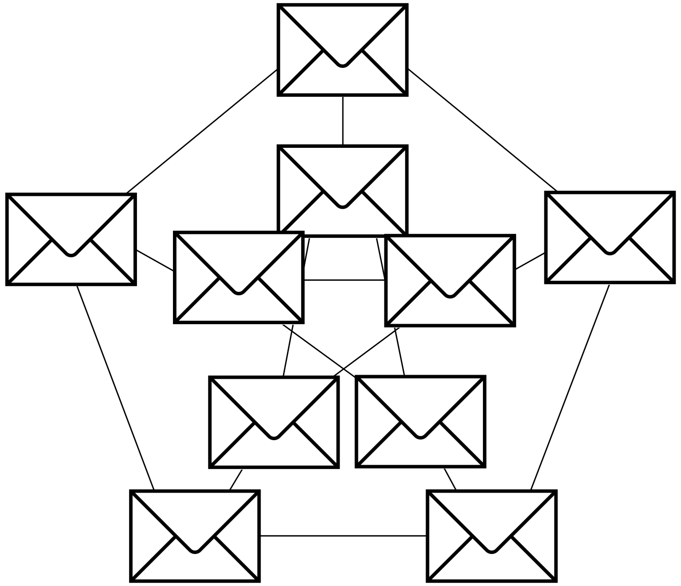
- Given accepting transcripts $\left((c_u)_{u \in V}, (a_i, b_i), col_{a_i}, d_{a_i}, col_{b_i}, d_{b_i}\right)$ for $|E|$ distinct edges $(a_i, b_i) \in E$, we cover every edge!
- If $a_i = b_j$ but $col_{a_i} \neq col_{b_j}$ then we break binding.

Elgamal: impossible

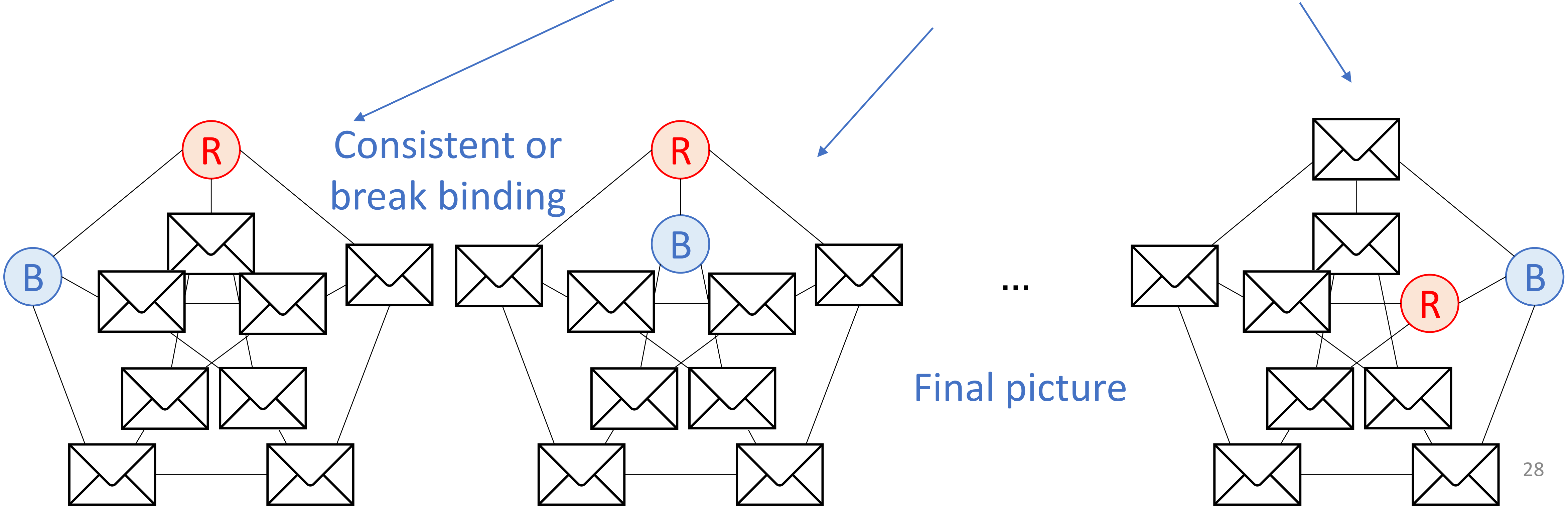
Pedersen: computationally impossible

- So $\left(col_{a_i}\right)_i$ are *consistent* with $\left(col_{b_j}\right)_j$ and define a colouring
- $col_{a_i} \neq col_{b_i}$ so this is a 3-colouring

Visual special soundness



$|E|$ transcripts so
we see every edge



SHVZK analysis

$S(pp, G, (a,b))$
1.$col_a, col_b \leftarrow_{\$} \{R, G, B\}$ with $col_a \neq col_b$.
2. $(c_a, d_a) \leftarrow_{\$} \text{Commit}(pp, col_a)$.
3. $(c_b, d_b) \leftarrow_{\$} \text{Commit}(pp, col_b)$.
4. $\forall u \in V \setminus \{a,b\}$, $(c_u, d_u) \leftarrow_{\$} \text{Commit}(pp, R)$.
5. Output $\left((c_u)_{u \in V}, (a,b), col_a, d_a, col_b, d_b\right)$

Commit, decommit distributions from Commit

What is the verifier's view?

$$\left((c_u)_{u \in V}, (a,b), col_a, d_a, col_b, d_b\right)$$

- $\text{Verify}(pp, c_a, d_a, col_a) = 1$
- $\text{Verify}(pp, c_b, d_b, col_b) = 1$
- $col_a \neq col_b$

Colours are random

Why is the simulator valid? (efficient, indistinguishable)

- $col_a \neq col_b$ and they are random as in a real execution
- c_a, d_a, c_b, d_b have the same distribution as a real execution
- $\text{Verify}(pp, c_a, d_a, col_a) = 1$ by correctness, and similarly for c_b
- $\forall u \in V \setminus \{a,b\}, c_u$ is indistinguishable from honest one by hiding

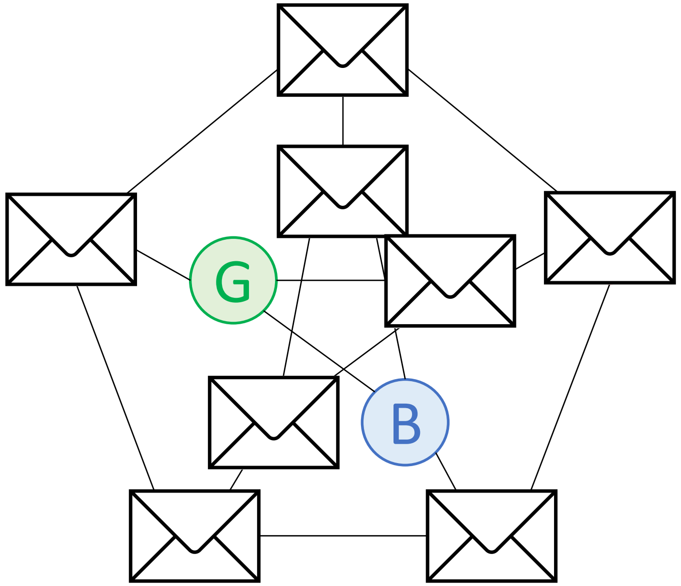
Elgamal: computational
Pedersen: perfect

Visual SHVZK

- Real protocol is ZK even for malicious V^*. See Goldreich textbook.
- Proof is subtle and shows d_a, d_b don't leak contents of unopened commitments.

Given an edge, colour the vertices randomly

Colour the other vertices arbitrarily



V just sees random mismatched colours

Discussion: S, E superpowers

- Black-box access (for forking and rewinding)
- Expected polynomial time
- S often simulates backwards and guesses challenges
- E often sees multiple related transcripts
- Alternatively, S, E might have trapdoors for the commitments

Why are these necessary?

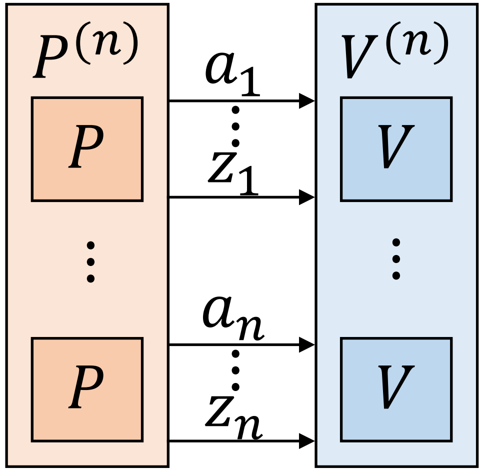
Agenda

- Variations of soundness ✓
- Σ-protocols ✓
- Commitment schemes ✓
- Σ-protocol for an NP-complete problem ✓
- **Composition methods for Σ-protocols**

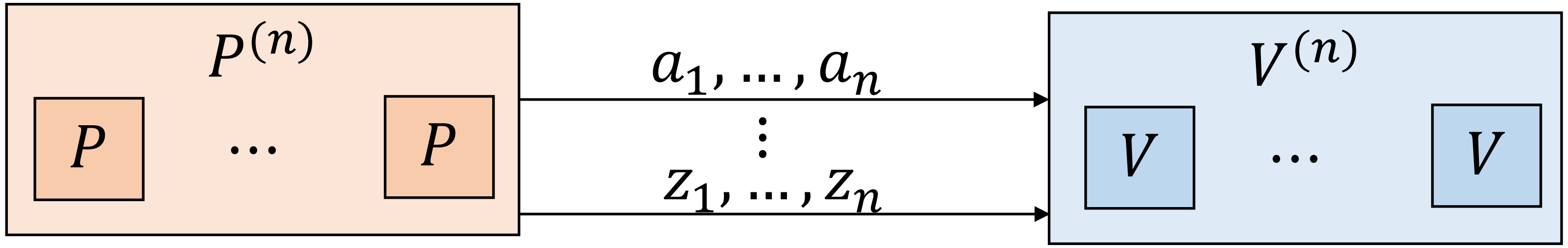
Security under composition

Σ-protocols still k-special sound, SHVZK under *parallel* composition

- Build protocols for complex relations
- Repeat protocols to improve completeness/soundness errors



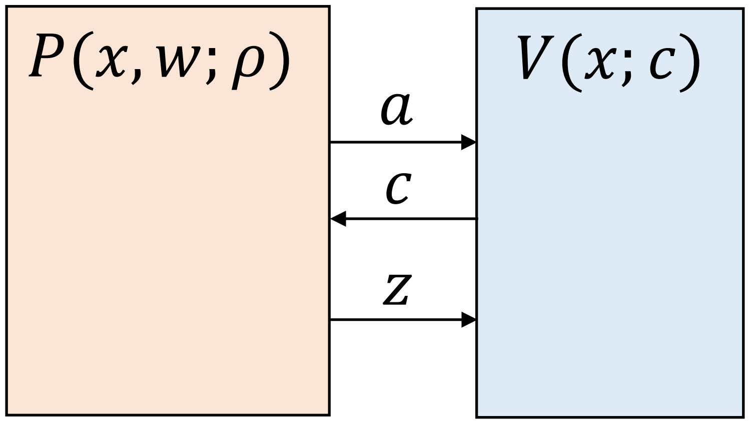
Various accept conditions possible e.g. majority, unanimous

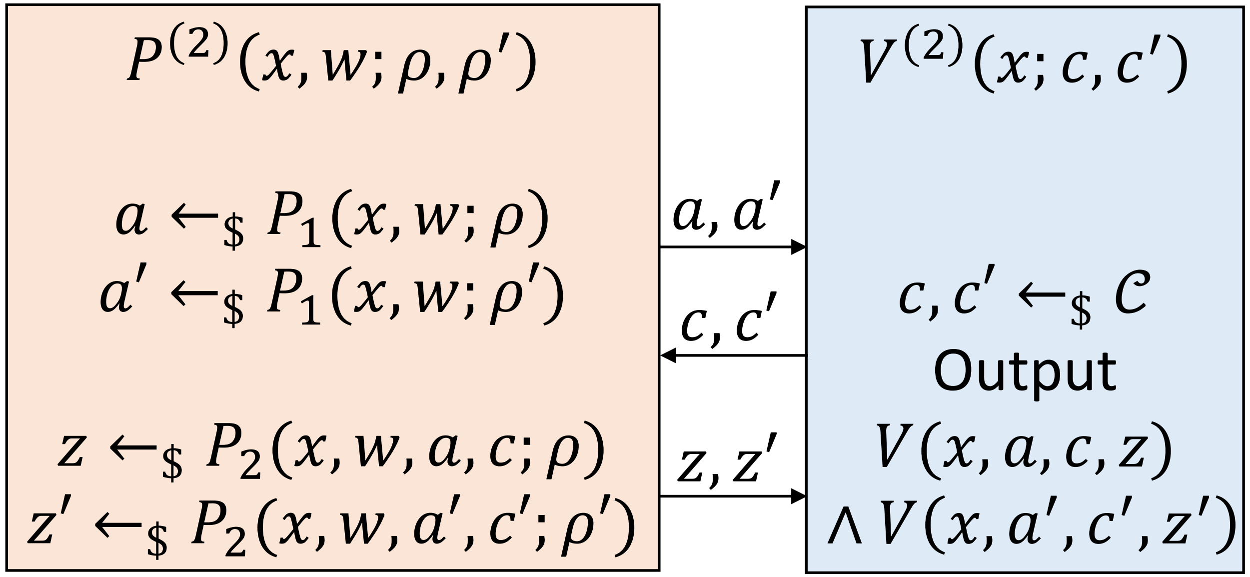


Preserved?	# Rounds	Soundness	ZK
Sequential	NO	YES – exercise 3.1	YES
Parallel	YES	YES for proofs NO for arguments	NO

Parallel repetition of Σ-protocols

- $\mathcal{R} = \{(x, w)\}$





Unanimous

Analysis of parallel repetition of Σ-protocols

Claim: (P, V) a 2-sound Σ-protocol ⇒ $\left(P^{(2)}, V^{(2)}\right)$ a 2-sound Σ-protocol

soundness error from $^1/_{|\mathcal{C}|}$ to $^1/_{|\mathcal{C}|^2}$

Proof:

- Completeness: $(x, w) \in \mathcal{R} \Rightarrow V(x, a, c, z), V(x, a', c', z') = 1$
- SHVZK: $S^{(2)}(x, c, c') \coloneqq \left(S(x, c), S(x, c')\right)$

$$\{S^{(2)}(x, c, c')\} = \{S(x, c), S(x, c')\} \approx \{\mathrm{View}_V^P(x, c), \mathrm{View}_V^P(x, c)\} = \{\mathrm{View}_{V^{(2)}}^{P^{(2)}}(x, c, c')\}$$

- 2-soundness:

accepting 2-tree for $\left(P^{(2)}, V^{(2)}\right)$

$$\begin{tikzcd}[column sep=small] & a, a' \arrow[dl, "{(c_1, c_1')}"'] \arrow[dr, "{(c_2, c_2')}"] & \\ z_1, z_1' \arrow[rr, phantom, "\neq"] & & z_2, z_2' \end{tikzcd}$$

$$(c_1, c_1') \neq (c_2, c_2') \\ \Downarrow \\ c_1 \neq c_2 \text{ or } c_1' \neq c_2'$$

W.L.O.G. $c_1 \neq c_2$.

accepting 2-tree for (P, V)

$$\begin{tikzcd}[column sep=small] & a \arrow[dl, "c_1"'] \arrow[dr, "c_2"] & \\ z_1 \arrow[rr, phantom, "\neq"] & & z_2 \end{tikzcd} \xrightarrow{E} (x, w) \in \mathcal{R}$$

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

Zero-Knowledge Proofs
Exercise 3

3.1 Definitions of Interactive Proofs

a) Show that the constants 3/4 and 1/2 in the completeness and soundness definitions for the class **IP** are arbitrary, i.e., that any other $0 < p < q \leq 1$ lead to an equivalent definition of **IP**.

HINT: Given an interactive proof (P, V), build an interactive proof (P', V') with completeness and soundness parameters q and p. You may use Hoeffding's inequalities: let $X_1, \ldots, X_n$ be i.i.d. Bernoulli random variables with parameter μ and $\bar{X} := (\sum_i X_i)/n$. For all $\varepsilon > 0$,

$$\Pr\left[\bar{X} \leq \mu - \varepsilon\right] \leq \exp(-2n\varepsilon^2) \text{ and}$$
$$\Pr\left[\bar{X} \geq \mu + \varepsilon\right] \leq \exp(-2n\varepsilon^2).$$

b) Show that a language L for which there exists an interactive proof (P, V) with V deterministic is in **NP**.

c) Show that for every interactive proof (P, V), with a probabilistic prover, there exists an interactive proof (P', V) for the same language L such that P' is deterministic.

HINT: Recall that the prover is computationally unbounded.

d) Consider a language L for which there exists an interactive proof (P, V) such that V always rejects when the input is not in L. Show that L is in **NP**.

3.2 Commitment Schemes

a) Explain why a commitment scheme cannot be both perfectly hiding and perfectly binding.

b) The RSA assumption is that no efficient algorithm can compute a value x such that $g = x^e \bmod N$ given an RSA public key (N, e) and a value $g \in \{0, \ldots, N-1\}$.

Consider the following commitment scheme [Fis01]:

$\mathsf{Setup}\left(1^\lambda\right) \to pp$**:** Choose an RSA modulus $N = pq$ with $2^{\lambda-1} \leq N < 2^\lambda$, a prime $e \geq 2^\lambda$, and let $g := x^e \bmod N$ for $x \leftarrow \mathbb{Z}_N^*$. Return public parameters $pp = (N, e, g)$.

$\mathsf{Commit}(pp, m, r) \to (c, d)$: To commit to a message $m \in \mathbb{Z}_e$, compute $c \leftarrow g^m r^e \bmod N$ for a random $r \leftarrow_\$ \mathbb{Z}_N^*$, set $d \leftarrow r$ and return (c, d).

$\mathsf{Verify}(pp, c, d, m) \to b \in \{0, 1\}$: Return 1 if $c = g^m d^e \bmod N$ and 0 otherwise.

Show that this commitment scheme is perfectly hiding and computationally binding under the RSA assumption.

Show that the commitment scheme is also *equivocable*: there is a trapdoor which makes it possible to open a commitment to any message.

3.3 Implementing ZKP for Graph Isomorphism (*)

<u>Note</u>: This exercise is not examinable. But the intention is to get students to practice implementing ZKPs from the lectures.

Implement the (perfect) zero-knowledge proof for graph isomorphism (GI) presented in Lecture 2; you can use your favorite programming language and libraries.

To be more precise, your code should implement the following helper functions:

- instance_generator
  - Input: n, an integer
  - Output: $((G_0, G_1), \pi)$, where G_1 is a random graph with n vertices, π is a random $n \times n$ permutation matrix and $G_0 = \pi(G_1)$. (You can use any valid representation of graphs – e.g., as adjacency matrices, members of a "graph" class in certain libraries, etc.)
- first_prover_msg
  - Input: $(n, (G_0, G_1), \pi)$, where n is an integer, (G_0, G_1) is a pair of graphs with n vertices and π is an $n \times n$ permutation matrix.
  - Output: (H, σ), where H is a graph with n vertices and σ is an $n \times n$ permutation matrix. The computation of (H, σ) follows from the prover's first message in the GI ZKP from Lecture 2.
- verifier_msg
  - Input: None.
  - Output: b, a bit. The computation of b follows from the verifier's first message in the GI ZKP from Lecture 2.
- second_prover_msg
  - Input: $(n, (G_0, G_1), \pi, \sigma, H, b)$, where $n, (G_0, G_1), \pi$ are as defined w.r.t. the input of first_prover_msg, σ is an $n \times n$ permutation matrix, H is a graph with n vertices and b is a bit.
  - Output: τ, an $n \times n$ permutation matrix. The computation of τ follows from the prover's second message in the GI ZKP from Lecture 2.
- verifier_checks
  - Input: $(n, (G_0, G_1), H, b, \tau)$, where $(n, (G_0, G_1), H, b)$ are as defined w.r.t. the input of second_prover_msg and τ is an $n \times n$ permutation matrix.
  - Output: b, a bit. The computation of b follows from the verifier's final output in the GI ZKP from Lecture 2. (Here you can have $b = 0$ and $b = 1$ to be synonymous with the verifier "rejecting" and "accepting" respectively.)

Your code should then take as input an integer n from a user and – using the above helper functions – display a pair of graphs with n vertices, the messages exchanged between an honest prover and an honest verifer in the GI ZKP on the aforementioned pair of graphs, and the verifier's final output.

Bonus: Can you demonstrate soundness of the GI ZKP using your implementation?

References

[Fis01] Marc Fischlin. *Trapdoor commitment schemes and their applications.* PhD thesis, Goethe University Frankfurt, Frankfurt am Main, Germany, 2001.