Lecture 4: MPC in the Head and Transformations of Σ-protocols

Zero-knowledge proofs

263-4665-00L

Lecturer: Jonathan Bootle

Announcements

- Exercise sheet 5 posted on Moodle
- Graded, 10% of final grade
- Submit through Moodle on or before 23:59, 20/10/2023
- Please email if you think you've found a typo or mistake

- 20/10/2023 exercise session used for optional/starred exercises

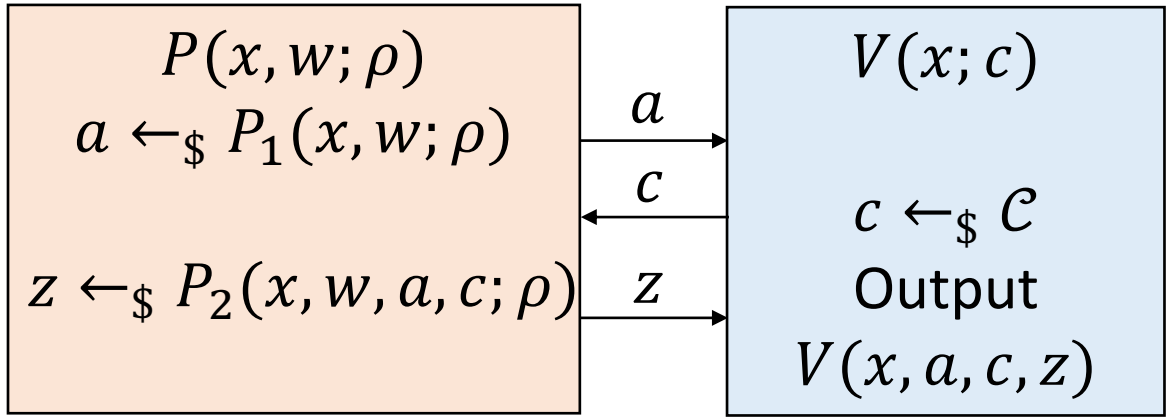
Last time

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

Agenda

- **Composition methods for Σ-protocols**
- Σ-protocols from MPC in the Head
- The Fiat-Shamir Transformation
- Making Σ-protocols zero-knowledge against malicious verifiers

Σ-protocols



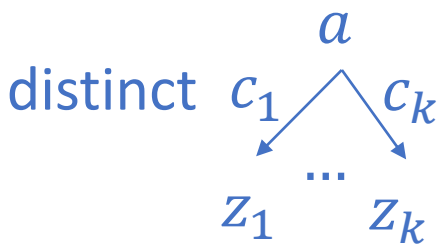
challenge space

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



$\xrightarrow{E}$ $(x, w) \in \mathcal{R}$

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

- SHZVK

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

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

Examples:

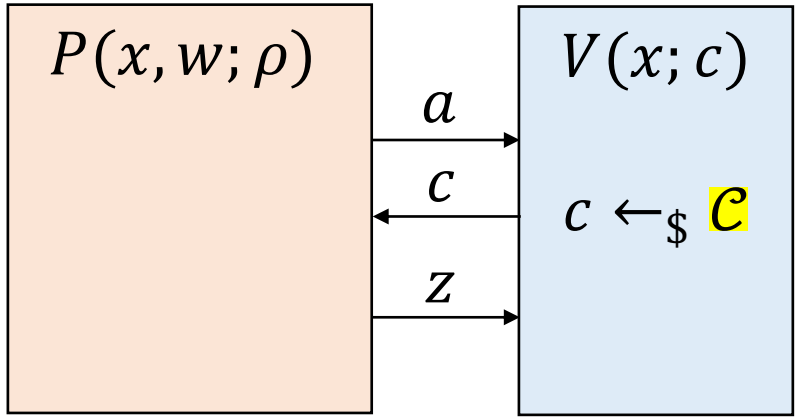
GI, QR protocol from Sheet 2

statistical, computational variants

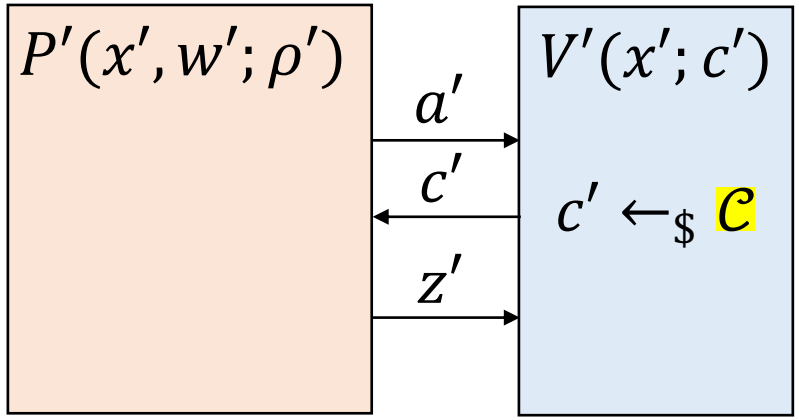
AND composition of Σ-protocols

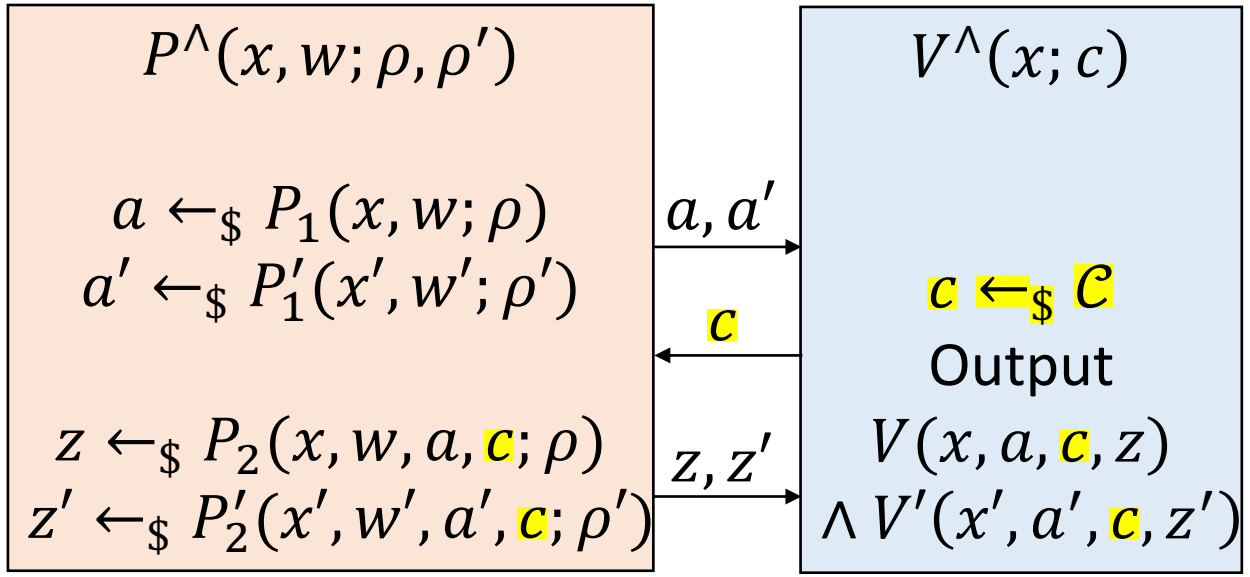
- $\mathcal{R}_\wedge := \{((x,x'),(w,w')) : (x,w) \in R \wedge (x',w') \in \mathcal{R}'\}$

$\mathcal{R}$



$\mathcal{R}'$





Use the same challenge c for both
(both protocols must use same $\mathcal{C}$)

Analysis of AND composition of Σ-protocols

Claim: $(P, V), (P', V')$ k-sound Σ-protocols ⇒ $(P^\wedge, V^\wedge)$ a k-sound Σ-protocol

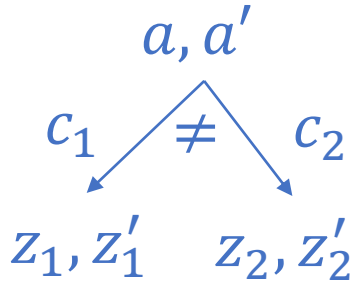
Proof:

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

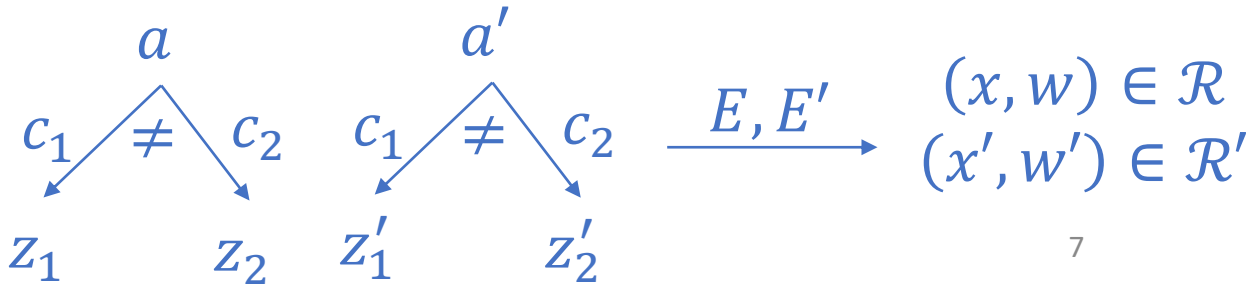
$$\{S^\wedge(x, x', c)\} = \{S(x, c), S'(x', c)\} \approx \{\mathrm{View}_V^P(x, c), \mathrm{View}_{V'}^{P'}(x', c)\} = \{\mathrm{View}_{V^\wedge}^{P^\wedge}(x, x', c)\}$$

- k-soundness:

accepting 2-tree for $(P^\wedge, V^\wedge)$



accepting 2-trees for $(P, V), (P', V')$

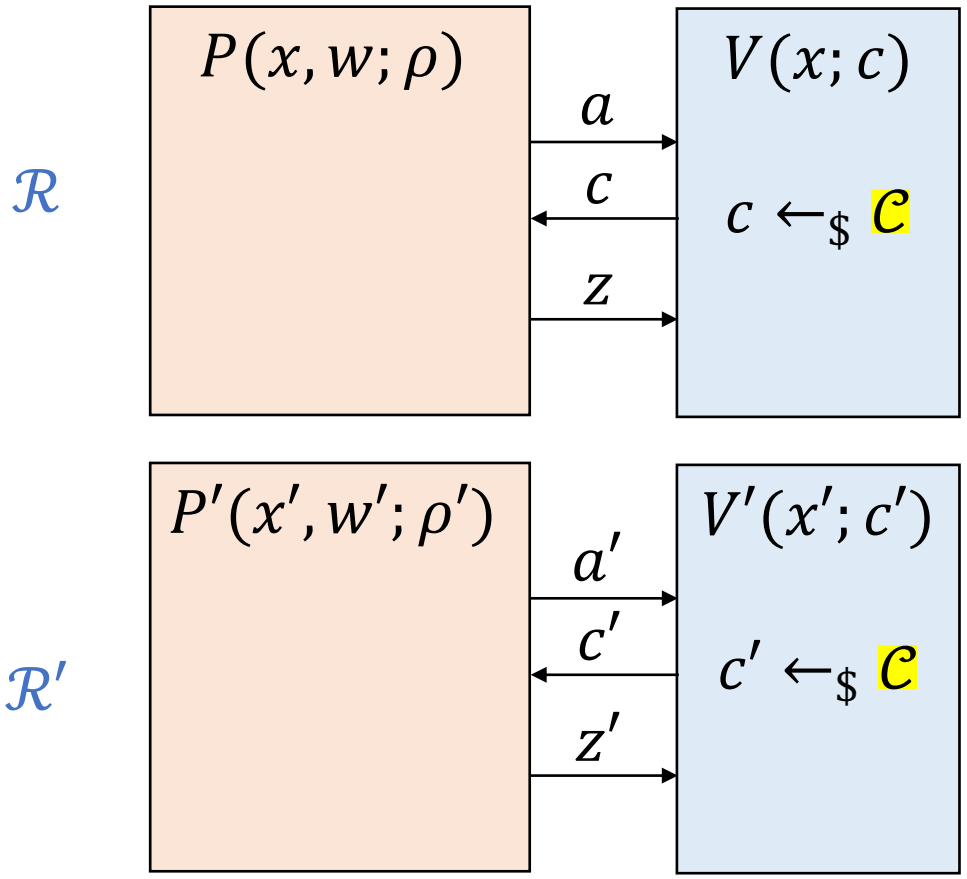


OR composition of Σ-protocols

$\mathcal{C}$ must be an additive group

Important for completeness

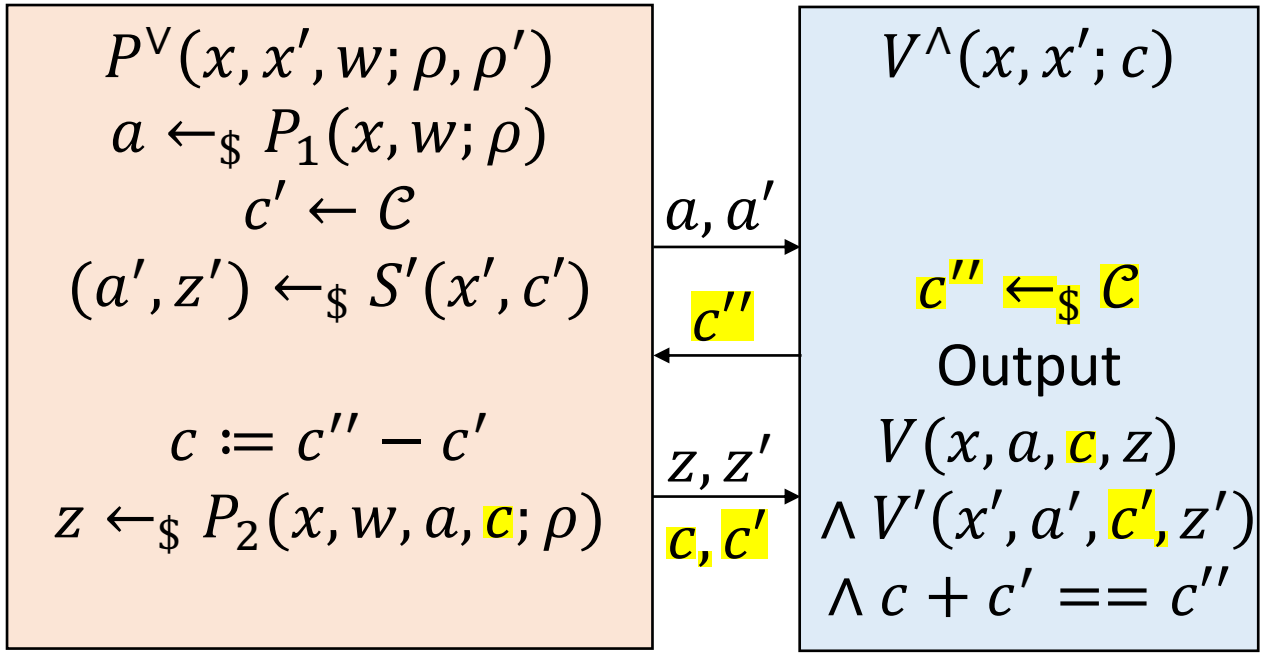
- $\mathcal{R}_{\mathsf{V}} \coloneqq \{((x,x'),w) : x \in \mathcal{L} \wedge x' \in \mathcal{L}', (x,w) \in R \vee (x',w) \in \mathcal{R}'\}$



Use a simulator to cheat in one of the protocols

Presented for $(x,w) \in \mathcal{R}$, simulate for x'

Otherwise replace P, S' with S, P'.



Analysis of OR composition of Σ-protocols

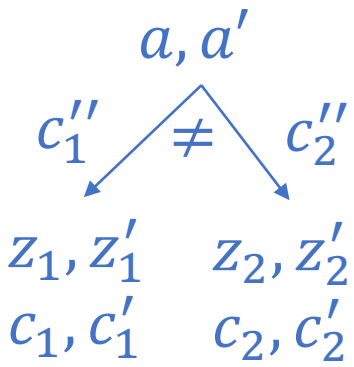
Claim: $(P, V), (P', V')$ 2-sound Σ-protocols $\Rightarrow$ $(P^{\vee}, V^{\vee})$ a 2-sound Σ-protocol

Proof:

- Completeness: $c + c' = c''$. W.L.O.G $(x, w) \in \mathcal{R} \Rightarrow V(x, a, c, z) = 1$.
- $x' \in \mathcal{L}' \Rightarrow S'(x, c')$ is perfectly simulated so $V'(x', a', c', z') = 1$.
- k-soundness:

Why we need $x \in \mathcal{L}, x' \in \mathcal{L}'$

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



$c_1'' \neq c_2''$

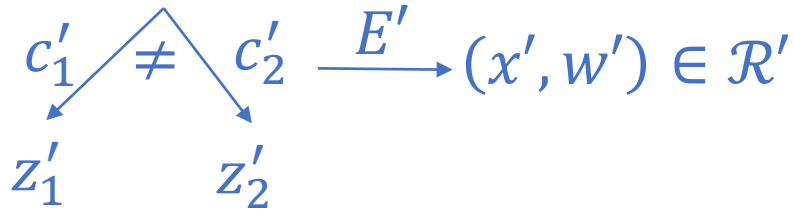
$c_1 + c_1' \neq c_2 + c_2'$

$\Downarrow$

$c_1 \neq c_2$ or $c_1' \neq c_2'$

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

accepting 2-tree for (P', V')



SHVZK of OR composition of Σ-protocols

- SHVZK:
- $S^{\vee}(x, x', c'') \coloneqq (S(x, c), S'(x', c'), c, c')$ with $c, c' \leftarrow_{\$} \mathcal{C}$ uniform conditioned on $c + c' = c''$.
- W.L.O.G. $(x, w) \in \mathcal{R}$
- $\{S^{\vee}(x, x', c'')\} = \{S(x, c), S'(x', c'), c, c'\}$

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

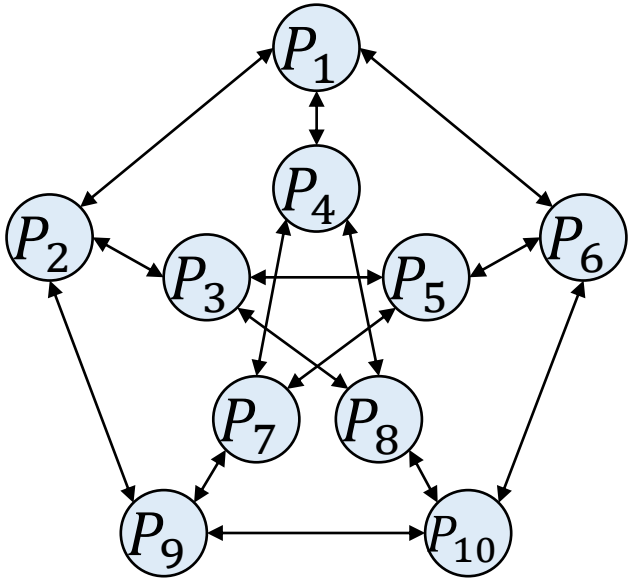
By SHVZK of the Σ-protocol for $\mathcal{R}$

Agenda

- Composition methods for Σ-protocols ✓
- **Σ-protocols from MPC in the Head**
- The Fiat-Shamir Transformation
- Making Σ-protocols zero-knowledge against malicious verifiers

MPC protocols

- Multiple players $P_1, \ldots, P_n$ want to compute a joint function of their private inputs w_i without leaking information on w_i
- Players interact and exchange messages over many rounds
- All players should get the output
- Players check each other's messages
- See Cryptographic Protocols course



MPC protocols

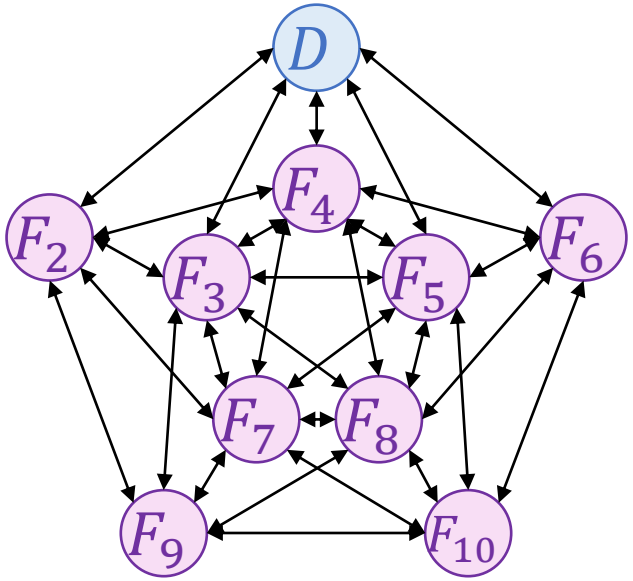
Example:

- Danish Sugar Beet auction (2008)
- https://en.wikipedia.org/wiki/Danish_Sugar_Beet_Auction

Output: contract allocation

Sugar manufacturer

Input: money



Sugar beet farmers

Input: production capacity and cost

Zero-Knowledge from Secure Multiparty Computation
Ishai, Kushilevitz, Ostrovsky, Sahai, 2007

MPC-in-the-head

Shares of a witness
$w \coloneqq w_1 \oplus \cdots \oplus w_n$

For any **NP**-relation $\mathcal{R}$,

$$f_{\mathcal{R}}(x, w_1, \ldots, w_n) \coloneqq [(x, w_1 \oplus \cdots \oplus w_n) \in \mathcal{R}] \in \{0,1\}.$$

MPC for $f_{\mathcal{R}}(x, w_1, \ldots, w_n)$

Commitments

Σ-protocol for $\mathcal{R}$

Idea: like G3C protocol but on MPC communication graph

Definitions of an MPC protocol

Can output a special end symbol to terminate the protocol, then P_i returns its local output

Definition:

A *multi-party computation protocol* Π

- Between players $P_1, \dots, P_n$
- On private inputs $w_1, \dots, w_n$
- On public input x
- With private randomness $r_1, \dots, r_n$

is specified by a *next message function*

$$\Pi\left(i, x, w_i, r_i, \left(M_1, \dots, M_j\right)\right) \rightarrow \left(m_{j+1, i \rightarrow k}\right)_{k \neq i}$$

$M_j = \left(m_{j,k \rightarrow i}\right)_{k \neq i}$

Messages from P_k, $k \neq i$

Output consists of all messages sent by P_i in round $j + 1$.

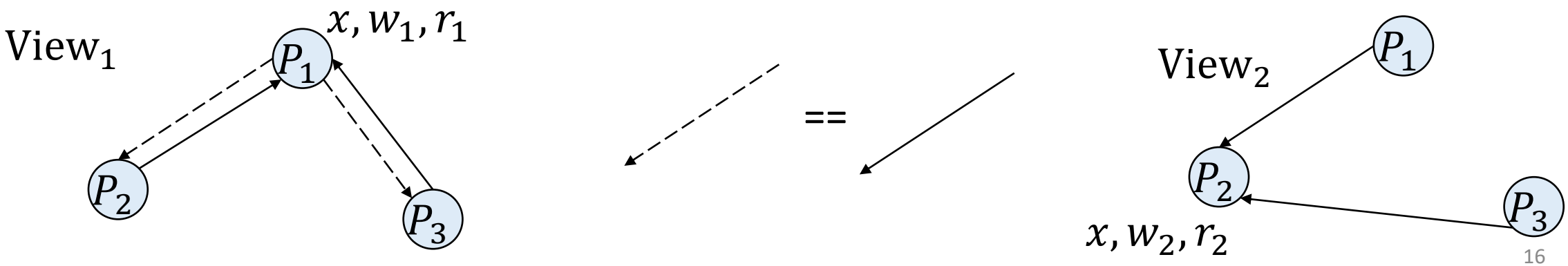
Player views

Definitions:

All messages received from $P_1, \ldots, P_{j-1}, P_{j+1}, \ldots, P_n$ in all rounds

- The *view* View_i of player P_i is $\text{View}_i := (x, w_i, r_i, M_1, \ldots, M_k)$.

- View_i , View_j are *consistent* if the messages sent from P_j to P_i (computed from View_i using Π) are the same as the messages received by P_j from P_i (according to View_j).



Local vs global consistency

Claim:

- Let Π be an n-party protocol and x a public input. Let $\mathrm{View}_1, \ldots, \mathrm{View}_n$ be a tuple of (possibly incorrect) views. Then

$\forall$ distinct i, j, View_i and View_j are consistent (with respect to Π, x) $\Leftrightarrow$ $\exists$ an honest execution of Π with input x (for some w_i, r_i) which matches each View_j

Proof:

($\Rightarrow$): by induction. If $\mathrm{View}_1, \ldots, \mathrm{View}_n$ match an honest execution up to round j, apply Π. By pairwise consistency, the execution extends to round $j + 1$.

($\Leftarrow$): follows by definition.

Properties of MPC protocols

"Honest but curious"
Parties in T follow Π but try to learn from messages received, like HVZK.

Definition:

An MPC protocol Π securely computes a function f in the semi-honest model if it satisfies

"All players get the correct output"

- *Correctness*: $\forall x, w_1, \dots, w_n$,

$$\Pr_{r_1,\dots r_n}[\exists i : \text{final output of } P_i \neq f(x, w_1, \dots, w_n)] = 0$$

- *t-privacy* (for some $t \in [n-1]$):

$\exists$ efficient simulator S^{MPC} such that $\forall T \subset [n]$ with $|T| \leq t$,

$$\{\text{Views}_{i\in T}(x, w_1, \dots, w_n)\} = \left\{S^{MPC}\left(T, x, \{w_i\}_{i\in T}, f(x, w_1, \dots, w_n)\Big|_T\right)\right\}$$

Views of players in T leak no information on $\{w_i\}_{i\notin T}$

Visual Σ-Protocol from MPC in the Head

Communication: 2 views,decommitments and n commitments

Computation: n commitments and one run of Π

Think $O(N)$ for a circuit with N gates

Randomly share witness

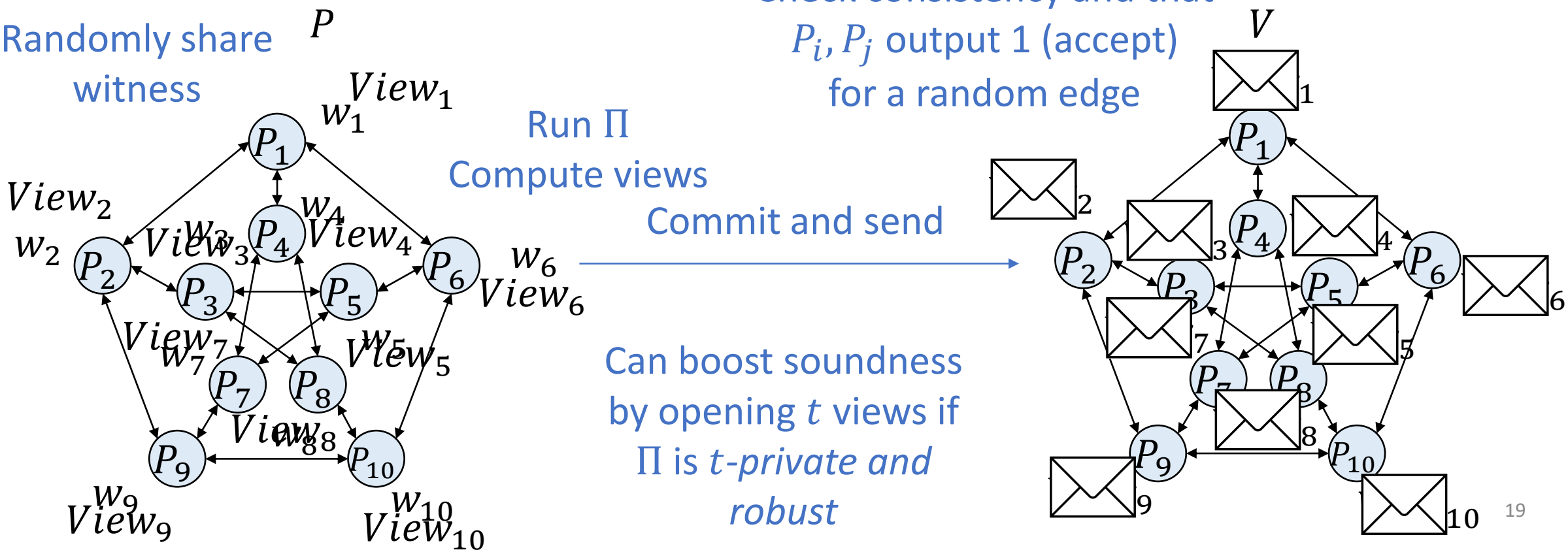
Run Π

Compute views

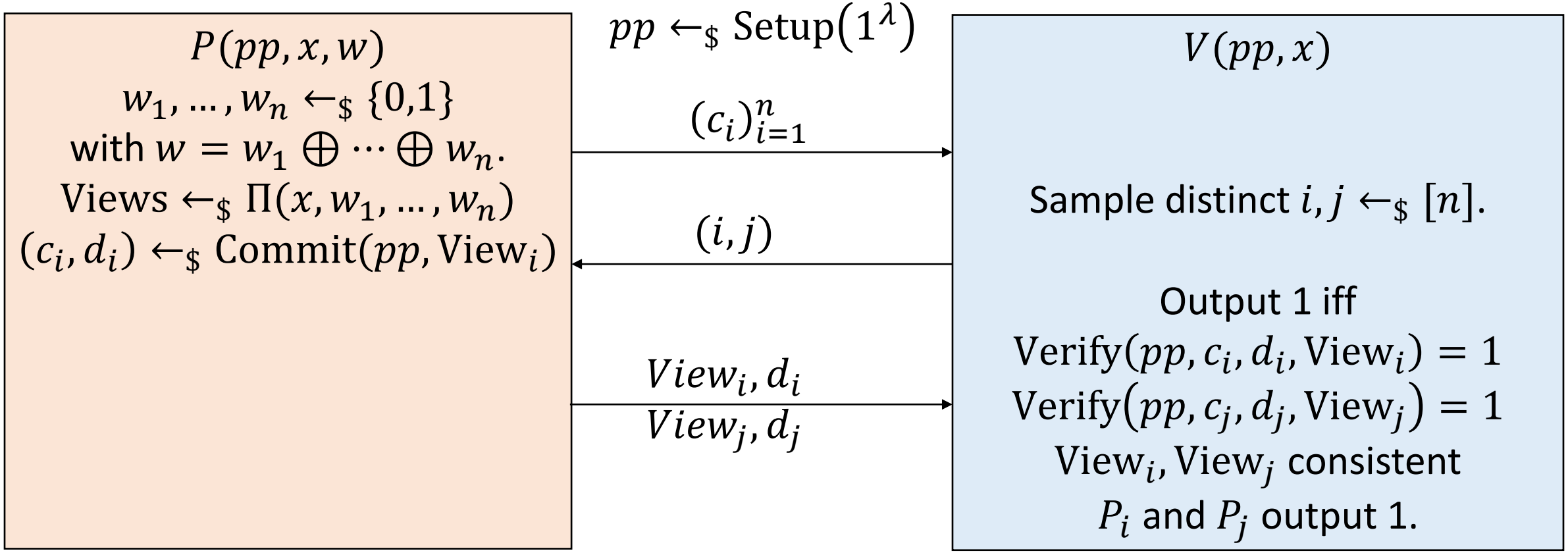
Commit and send

Check consistency and that P_i, P_j output 1 (accept) for a random edge

Can boost soundness by opening t views if Π is *t-private and robust*



Proper Σ-Protocol from MPC in the Head



Completeness, $\binom{n}{2}$-special soundness analysis

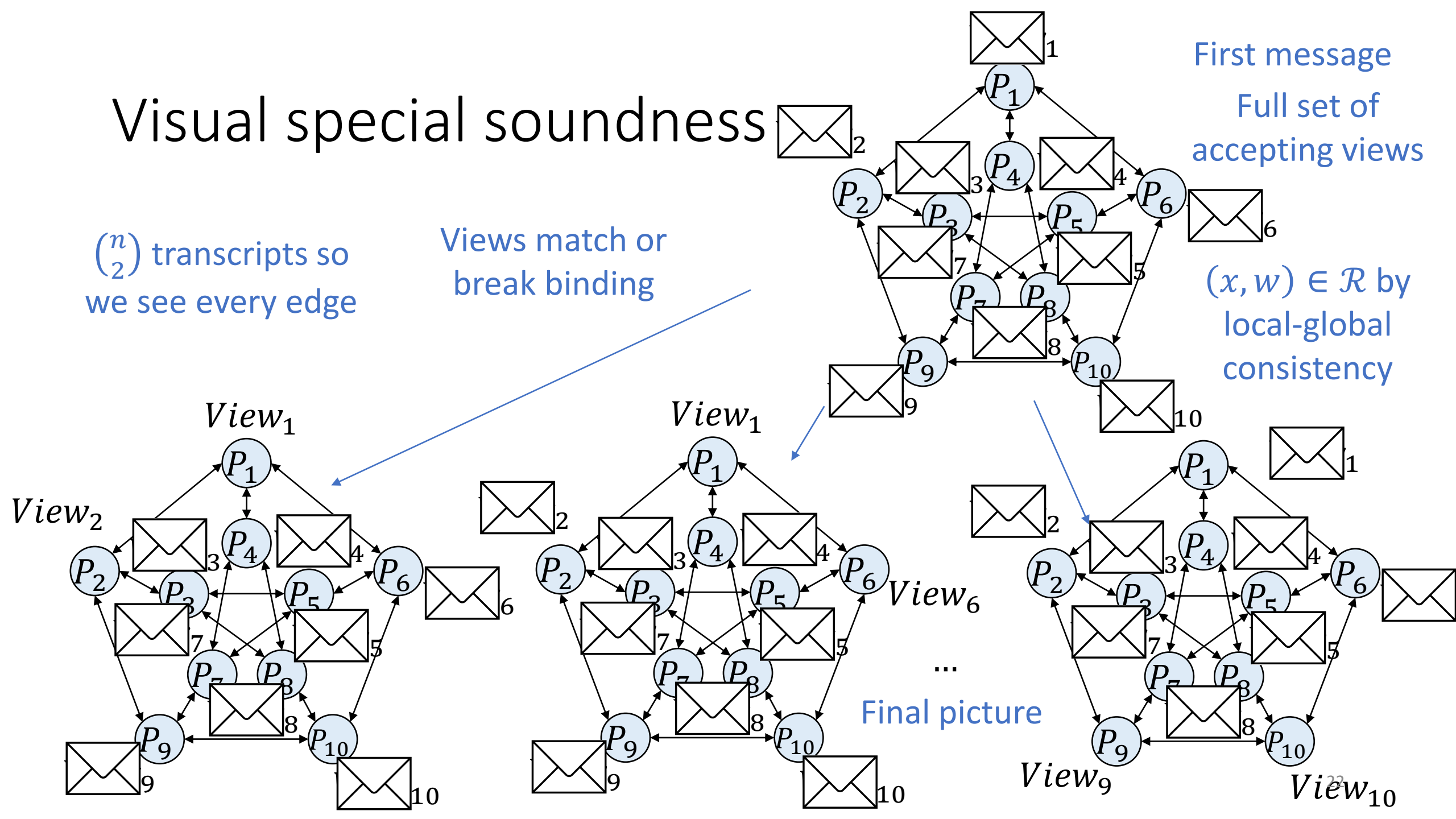
Completeness:

- If $(x, w) \in \mathcal{R}$ then every P_i computes $f_{\mathcal{R}} = 1$ by correctness of Π and views are consistent
- By the correctness of the commitment scheme, $\text{Verify}(pp, c_i, d_i, \text{View}_i) = 1$, and similarly for c_j.

$\binom{n}{2}$-special soundness: commitments edge challenges views and decommitments

- Given accepting transcripts $\left((c_i)_{i=1}^{n}, (i_k, j_k), \text{View}_{i_k}, d_{i_k}, \text{View}_{j_k}, d_{j_k}\right)$ for $\binom{n}{2}$ distinct edges (i_k, j_k), we cover every edge!
- If $i_k = j_l$ but $\text{View}_{i_k} \neq \text{View}_{j_l}$ then we break binding.
- Single set of consistent accepting views. ∃ honest execution of Π with $f(x, w_1, \dots, w_n) = 1$ by local-global consistency, hence $(x, w) \in \mathcal{R}$.

Visual special soundness



SHVZK analysis

Commit, decommit distributions from Commit

What is the verifier's view?

- $\left((c_i)_{i=1}^n, (i,j), \text{View}_i, d_i, \text{View}_j, d_j\right)$
- $\text{Verify}(pp, c_i, d_i, \text{View}_i) = 1$
- $\text{Verify}\left(pp, c_j, d_j, \text{View}_j\right) = 1$
- w_i, w_j random, P_i, P_j output 1.

Views consistent

Need $n \geq 3$ or w_1, w_2 correlated $w_1 \oplus w_2 = w$

$S\left(pp, x, (i,j)\right)$

1. $w_i, w_j \leftarrow_\$ \{0,1\}$.
2. $\text{View}_i, \text{View}_j$ sampled with $S^{MPC}\left(i, j, x, w_i, w_j, 1,1\right)$.
3. $(c_i, d_i) \leftarrow_\$ \text{Commit}(pp, \text{View}_i)$.
4. $\left(c_j, d_j\right) \leftarrow_\$ \text{Commit}(pp, \text{View}_j)$.
5. $\forall u \in [n] \setminus \{i,j\}$, $(c_u, d_u) \leftarrow_\$ \text{Commit}(pp, 0)$.
6. Output $\left((c_i)_{i=1}^n, (i,j), \text{View}_i, d_i, \text{View}_j, d_j\right)$

Why is the simulator valid? (efficient, indistinguishable)

- $\text{View}_i, \text{View}_j$ distributed as in a real execution by 2-privacy.
- c_i, d_i, c_j, d_j have the same distribution as a real execution.
- $\text{Verify}(pp, c_i, d_i, \text{View}_i) = 1$ by correctness, and similarly for c_j.
- $\forall u \in [n] \setminus \{i,j\}, c_u$ is indistinguishable from honest one by hiding.

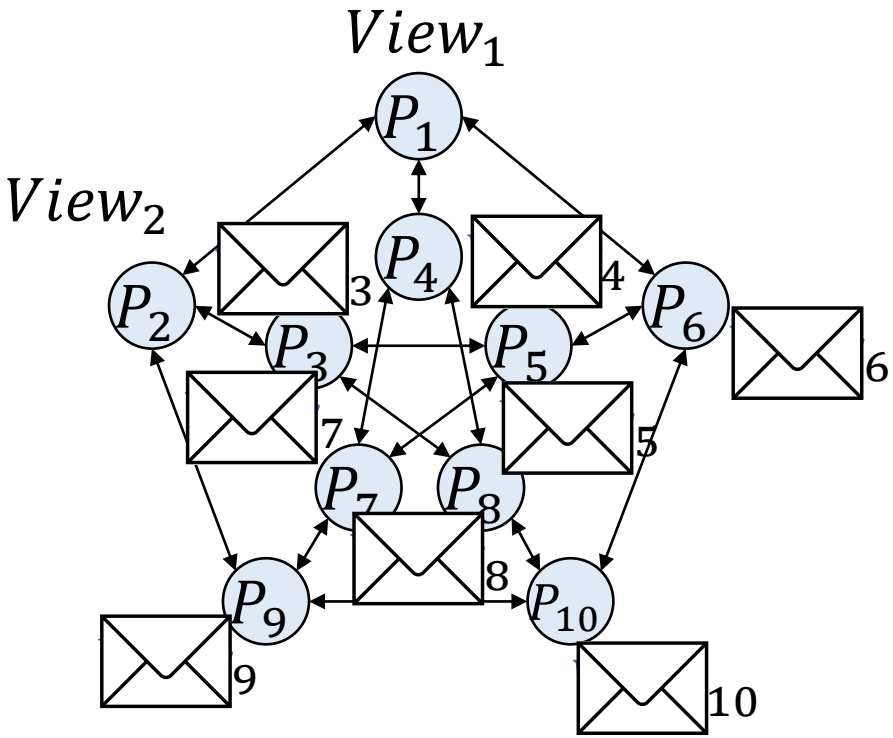
S^{MPC} is efficient

Visual SHVZK

- Real protocol is ZK even for malicious V^* (similar subtlety to G3C protocol)

Given an edge, simulate accepting views $View_1$, $View_2$ using $S^{MPC}(i, j, x, w_i, w_j, 1,1)$

Set the other views to 0 and commit



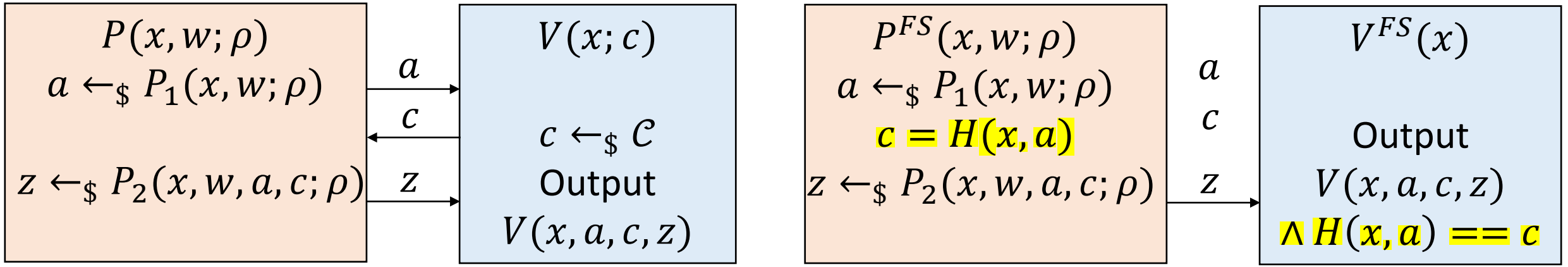
V just sees random simulated accepting views

Agenda

- Composition methods for Σ-protocols ✓
- Σ-protocols from MPC in the Head ✓
- **The Fiat-Shamir Transformation**
- Making Σ-protocols zero-knowledge against malicious verifiers

The Fiat-Shamir Heuristic

- Makes a Σ-protocol non-interactive (i.e. just one message)



$c \leftarrow_\$ \mathcal{C}$ on each *new* input to H

H 'remembers' outputs for earlier inputs.

- Hash function H (secure if H modelled as a random oracle)
- Generalises to public-coin protocols with more rounds
- Gives full ZK for free

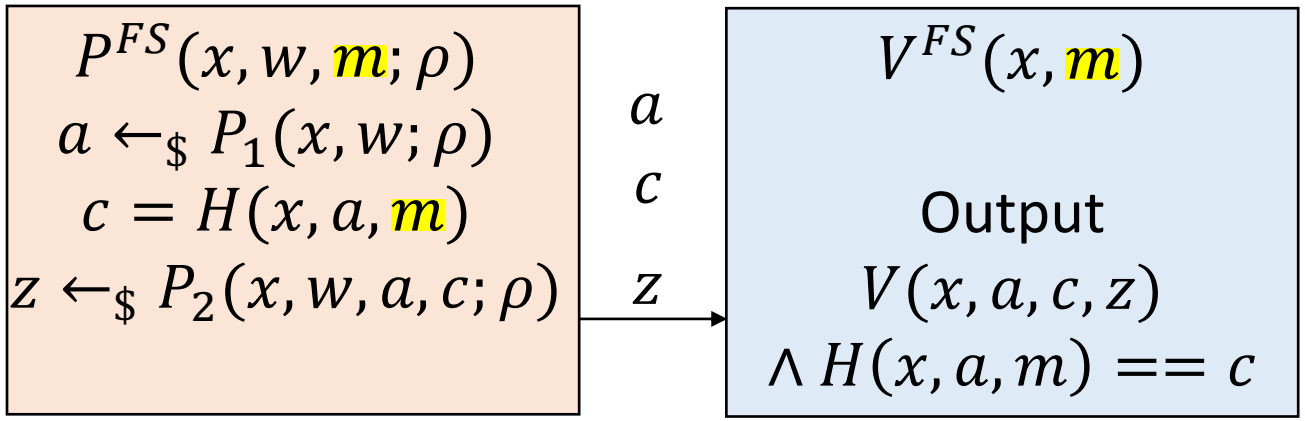
S may have control over H

Notes on the Fiat-Shamir Heuristic

- Hash pp if the instance involves commitments, hash more not less!
- Not secure for real hash functions in general
- Provably secure for certain hash-functions

Peikert, Shiehian, 2019, Noninteractive Zero Knowledge for NP from (Plain) Learning With Errors

- Works for public coin protocols (i.e. not GNI). Why?
- Signature scheme with $x = pk$, $w = sk$, Sign = prover, verify = verifier



Keygen: Easy to sample (x, w)
Hard to compute w from x

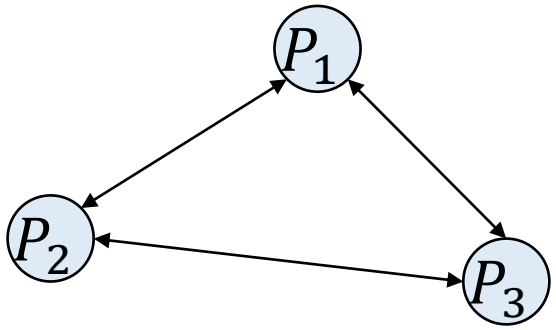
Adversaries may hash many values of a, m

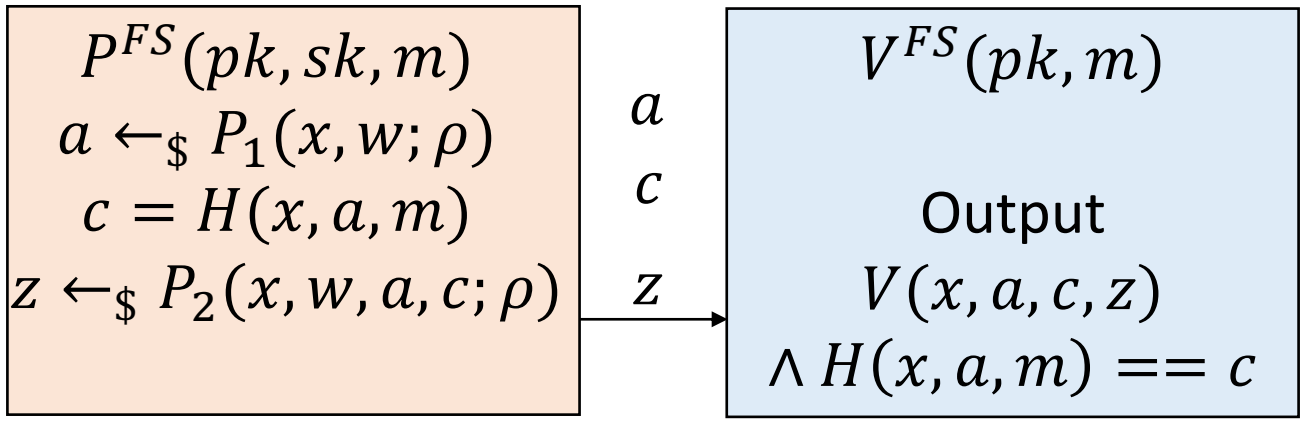
Pointcheval, Stern, Security Proofs for Signature Schemes

Application: post-quantum signature schemes

- Picnic signature scheme (NIST standardisation candidate)
- Language: $pk = f(sk)$ for a 'friendly' block cipher f
- MPC protocol with 3 parties
- Hash-based commitments

Uses MPC-in-the-Head and Fiat-Shamir
MPC protocol on later exercise sheet





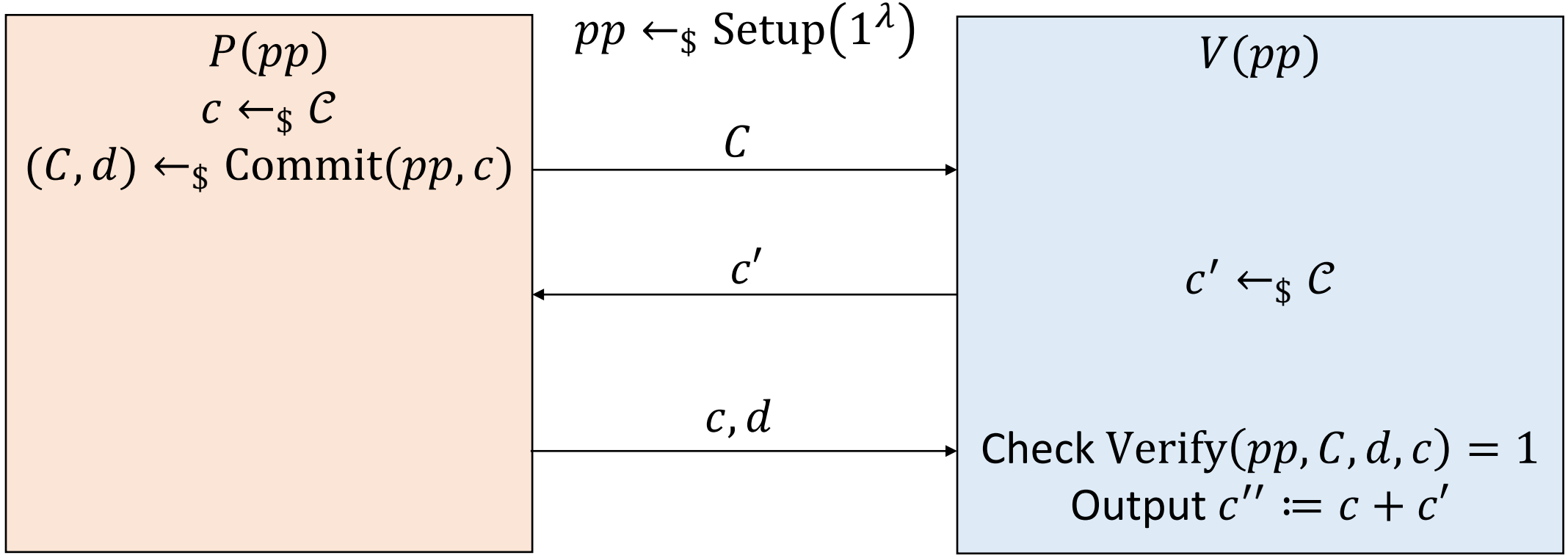
Agenda

- Composition methods for Σ-protocols ✓
- Σ-protocols from MPC in the Head ✓
- The Fiat-Shamir Transformation ✓
- **Making Σ-protocols zero-knowledge against malicious verifiers**

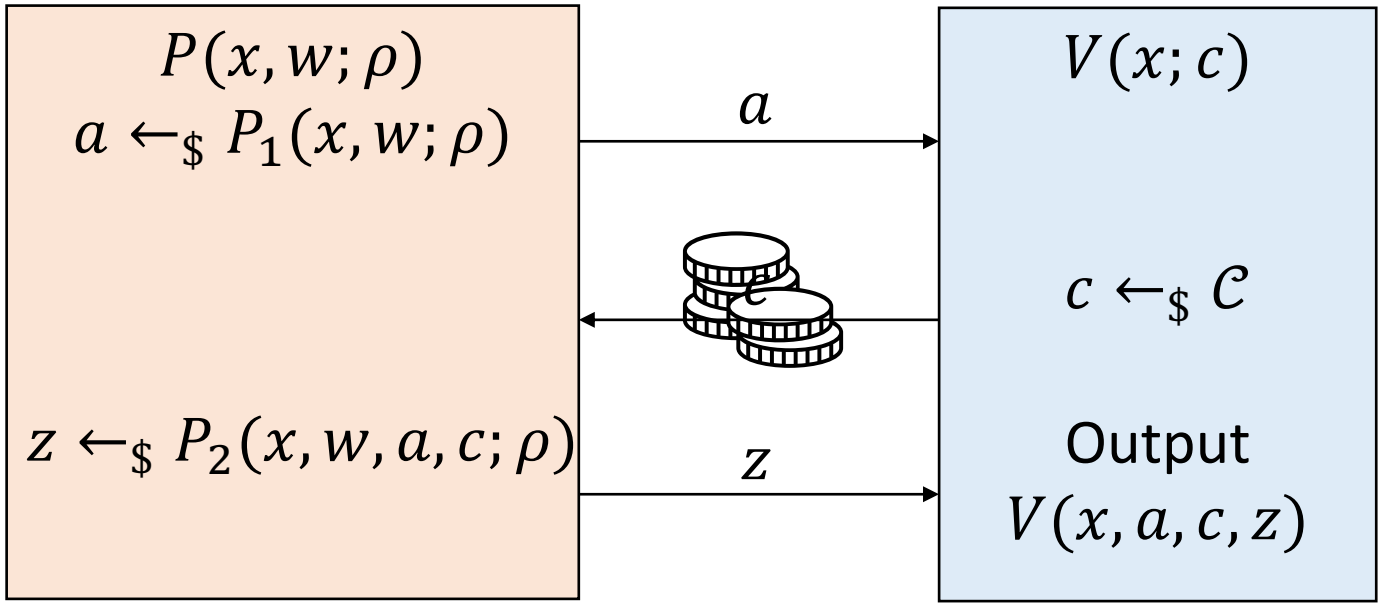
Coin flip protocol

Forces honest V by generating challenges together

$\mathcal{C}$ a group



Compiling Σ-protocols to fully ZK protocols

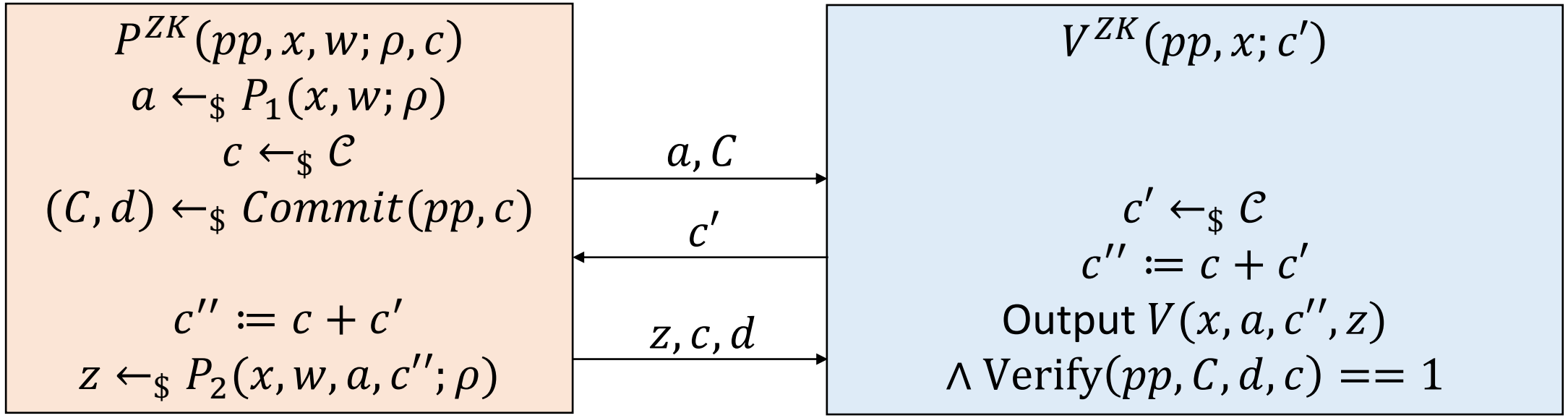


Compiling Σ-protocols to fully ZK protocols

Variant where (P, V) have opposite roles in coin flip protocol

First assume $\mathcal{C}$ is polynomially bounded

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



Theorem:

Against malicious verifiers

(P^{ZK}, V^{ZK}) has completeness, k-special soundness and zero-knowledge.

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

Zero-Knowledge Proofs
Exercise 4

4.1 Necessity of Prover's Randomness in ZK Protocols

Let (P, V) be an interactive proof system *with auxiliary inputs* for a language L. Now suppose (P, V) is (computational) zero-knowledge such that P is *deterministic*. Then show that $L \in \mathbf{BPP}$.[1]

4.2 Zero-Knowledge Protocol for Graph Non-Isomorphism

a) Explain why the GNI protocol given in the 1st lecture is not zero-knowledge.

b) Show that the same protocol is honest-verifier zero-knowledge.

c) Design a statistical zero-knowledge protocol for the GNI problem.

HINT: Use the GI$[k]$ protocol in Task 4.3 as a *sub-protocol*. You can rely on the fact that GI$[k]$ is knowledge sound with knowledge error 2^{-k} as proven in Task 4.3.

4.3 (Zero-Knowledge) Proof of Knowledge for Graph Isomorphism (*)

Let $\mathrm{GI} = (P_{\mathrm{GI}}, V_{\mathrm{GI}})$ be the zero-knowledge protocol for graph isomorphism seen in the 2nd lecture. Now let $\mathrm{GI}[k] = (P_{\mathrm{GI}[k]}, V_{\mathrm{GI}[k]})$ be the *k-sequential repetition* of the GI protocol such that the verifier $V_{\mathrm{GI}[k]}$ accepts if and only if the verifier V_{GI} in every single execution of the basic GI protocol accepts. Prove that GI$[k]$ is knowledge sound with knowledge error 2^{-k}.

References

[1] Roughly speaking, **BPP** is essentially a *randomized* version of class **P**, and still contains "easy" problems. We say that $L \in \mathbf{BPP}$ if there exists a *randomized* algorithm M and a polynomial q such that,

- if $x \in L$, then $\Pr[M(x) = 1] \geq 3/4$,
- if $x \notin L$, then $\Pr[M(x) = 1] \leq 1/2$,
- for all x, M terminates in at-most $q(|x|)$ steps on input x.

Also see https://en.wikipedia.org/wiki/BPP_(complexity), https://complexityzoo.net/Complexity_Zoo:B#bpp.

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

Zero-Knowledge Proofs
Exercise 4

4.1 Necessity of Prover's Randomness in ZK Protocols

Let (P, V) be an interactive proof system *with auxiliary inputs* for a language L. Now suppose (P, V) is (computational) zero-knowledge such that P is *deterministic*. Then show that $L \in \mathbf{BPP}$.[1]

Solution: Given such a proof system (P, V), we construct a *randomized* polynomial time algorithm M_L which decides L. At a high-level, $M_L(x)$ tries to simulate an interaction between *deterministic* prover P and *honest* verifier V on common input x by invoking the assumed zero-knowledge simulator M_{sim} w.r.t. (P, V) in multiple rounds with different auxiliary inputs. Without loss of generality, let P be the party that sends the first message in the proof system (e.g., the first message can just be "start" which denotes P asking V to start its proof). Now consider a cheating verifier V^* which given an auxiliary input $(z_1, \ldots, z_i)$ sends z_j as its j-th message for all $j \leq i$ (the remaining messages $(z_j)_{j>i}$ of V^* are chosen arbitrarily). Note that because P is deterministic, in a real interaction with V^*, the first $i \leq t$ responses of P are completely determined by $z_1, \ldots, z_i$. For any x in the language, this must also be the case in the corresponding simulation by $M_{sim}(x, V^*(z_1, \ldots, z_i))$ with overwhelming probability, because of (computational) indistinguishability between the real and simulated views of V^* for all instances in the language.

Using the above observation, $M_L(x)$ proceeds as follows. It first chooses uniform randomness r for the *honest* verifier V and runs $M_{sim}(x, V(r))$ to obtain a simulated transcript, from which it extracts prover messages $y_1, \ldots, y_t$ and verifier messages $z_1, \ldots, z_t$. Then $M_L(x)$ tries to simulate a real interaction between P and V on input x and verifier randomness r. For $i = 1, \ldots, t$, to simulate the ith message of P in this interaction, it runs M_{sim} for cheating verifier $V^*(z_1, \ldots, z_{i-1})$ and extracts P's answers y'_j for $1 \leq j \leq i$ from the transcript it obtained. $M_L(x)$ stops the and outputs the bit $b = 0$ if it observes *inconsistencies* in P's messages across different invocations of M_{sim}, i.e. if $y'_j \neq y_j$ for some $j \leq i$. Otherwise it uses y_i as P's ith answer in the protocol execution with V. Finally, $M_L(x)$ outputs $b = z_t$.

Now if $x \in L$, $M_L(x)$'s simulated interaction between P and V on input x matches the real interaction with overwhelming probability. If the statistical distance between the interactions is some negligible quantity ε, then from completeness of (P, V) we have

[1] Roughly speaking, **BPP** is essentially a *randomized* version of class **P**, and still contains "easy" problems. We say that $L \in \mathbf{BPP}$ if there exists a *randomized* algorithm M and a polynomial q such that,

- if $x \in L$, then $\Pr[M(x) = 1] \geq 3/4$,
- if $x \notin L$, then $\Pr[M(x) = 1] \leq 1/2$,
- for all x, M terminates in at-most $q(|x|)$ steps on input x.

Also see https://en.wikipedia.org/wiki/BPP_(complexity), https://complexityzoo.net/Complexity_Zoo:B#bpp.

$\Pr[M_L(x) = 1] \geq 3/4 - t \cdot \varepsilon$. (From Exercise 3.1 (a) on the definition **IP** with "flexible" completeness and soundness errors, it's not hard to see that "$\Pr[M(x) = 1] \geq 3/4 - \varepsilon$" also suffices for the definition of **BPP**, if $1/2 < 3/4 - \varepsilon$.) If $x \notin L$, note that there's no guarantee on the zero-knowledge simulator M_{sim} correctly generating the honest prover P's messages. However, since $M_L(x)$ still runs the honest verifier V on independent randomness r and on arbitrary (or "cheating") prover's messages generated by M_{sim}, because of the soundness guarantee of (P, V) we have $\Pr[M_L(x) = 1] \leq 1/2$.

Remark: Note that in the above proof the auxiliary input to malicious prover V^* depends on the instance x, which is common input to the protocol and could be sampled from a superpolynomially large language. Thus, a similar argument would not go through for interactive proof systems *without* auxiliary inputs.

4.2 Zero-Knowledge Protocol for Graph Non-Isomorphism

a) Explain why the GNI protocol given in the 1st lecture is not zero-knowledge.

Solution: The protocol is not zero-knowledge as a malicious verifier could send an arbitrary graph to the honest prover and learn whether it is isomorphic to one of the two input graphs.

More formally, assume the protocol was zero-knowledge. Then for every efficient verifier V^* there exists an efficient simulator S that on input V^* and an instance (G_0, G_1) in the language (i.e. $G_0 \ncong G_1$) can perfectly simulate V^*'s view in an execution of the protocol with the honest prover P. We can use this simulator S to solve the graph isomorphism problem as follows: On input an instance (G_0^*, G_1^*), set $G_0 := G_0^*$ and sample an arbitrary graph $G_1 \ncong G_0$. Then run $S(V^*(G_1^*), (G_0, G_1))$ for a cheating verifier $V^*(G_1^*)$ that instead of sampling H honestly, sets $H = G_1^*$. Since the instance (G_0, G_1) is in the language, the simulator has to output a transcript that just looks like a real transcript between $V^*(G_1^*)$ and P. Now if $G_0^* \cong G_1^*$, then the P's bit in this transcript is $B = 0$, otherwise P either outputs $B = 1$ or aborts. Thus, one can conclude that $G_1 \cong G_0$ if $B = 0$ and $G_1 \ncong G_0$ otherwise. This shows that the GNI protocol is not zero-knowledge unless the graph isomorphism problem is easy.

b) Show that the same protocol is honest-verifier zero-knowledge.

Solution: Consider an algorithm which, on the input of two non-isomorphic graphs G_0 and G_1 on n vertices, generates uniformly at random a bit b and a permutation σ of $[\![n]\!]$, computes $H \leftarrow \sigma G_b$ and returns a transcript $((G_0, G_1), (b, \sigma), H, b, 1)$. The distribution of this transcript is the same as in an honest protocol execution, and the protocol is thus honest-verifier zero-knowledge. (In fact, it is even SHVZK, as can be shown by a similar argument.)

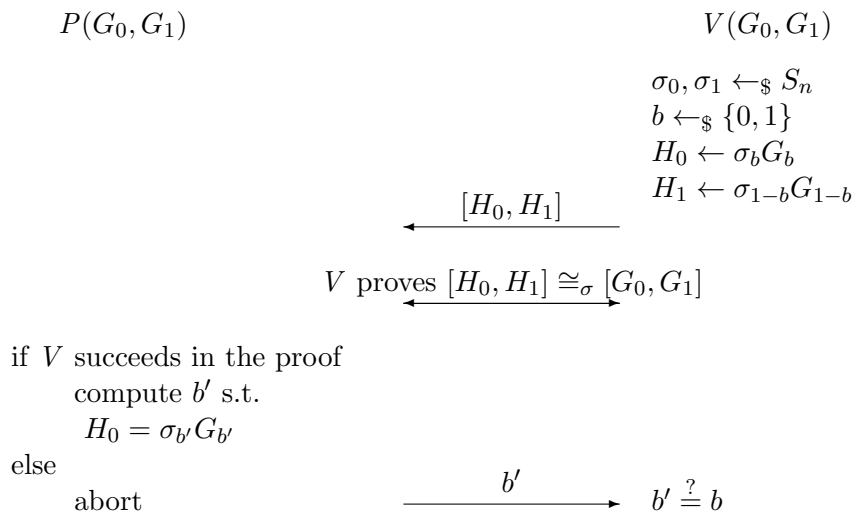
c) Design a statistical zero-knowledge protocol for the GNI problem.

HINT: Use the GI[k] protocol in Task 4.3 as a *sub-protocol*. You can rely on the fact that GI[k] is knowledge sound with knowledge error 2^{-k} as proven in Task 4.3.

Solution: In the standard GNI protocol, if the prover were additionally guaranteed that the graph the verifier sends in its first flow is isomorphic to one of the two input graphs, the verifier would be forced to follow the protocol and since the standard GNI protocol is honest-verifier zero-knowledge, the overall protocol would be fully zero-knowledge.

The main idea of the new protocol is then to have the verifier permute both input graphs and concatenate them in a random order, send the resulting graph to the prover, give a zero-knowledge proof that the new graph is isomorphic to the concatenation of

G_0 and G_1. The prover would be then able to tell in which order the permuted graphs were concatenated if and only if G_0 and G_1 are non-isomorphic.



For the zero-knowledge proof that $[H_0, H_1] \cong_\sigma [G_0, G_1]$, V uses the k-sequential repetition of the GI protocol, i.e., the GI$[k]$ protocol discussed in Task 4.3 above.

Completeness. The protocol is complete as the prover can tell which order the input graphs are concatenated in if the input graphs are non-isomorphic.

Soundness. To show that the protocol is sound, consider a malicious prover P^*. The idea now is to use the zero-knowledge property of GI$[k]$.[2] Let $S^{P^*}_{\mathrm{GI}[k]}$ be the ZK simulator for GI$[k]$ with oracle access to P^*, which for each round $i \in [\![k]\!]$ runs the simulator of GI as seen in the lecture. If we run this simulator on the instance $([H_0, H_1], [G_0, G_1])$, then it will output a transcript that is identically distributed to a real interaction of P^* and V in GI$[k]$, and P^*'s state after this simulation will be identically distributed to its state after a real execution of GI$[k]$. We can now extend this simulator to a simulator for the entire transcript of our GNI protocol. For this, the simulator simply has to execute the initial phase of the honest verifier V and then after the simulation of GI$[k]$ simply continue running P^*. We will now change the initial phase of V to make this simulation independent of the bit b until the final check $b' \stackrel{?}{=} b$: When simulating V, the simulator samples σ_0, σ_1 uniformly at random and sets $H_0 := \sigma_0(G_0)$ as well as $H_1 := \sigma_1(G_1)$. Since G_0 and G_1 are isomorphic, $[H_0, H_1]$ are identically distributed as before, and as the simulation of GI$[k]$ is independent of σ also the rest of the transcript is identically distributed as before. But this means that there is no way for the prover P^* to infer any information about the bit b and it hence succeeds with probability at most $1/2$.

Statistical Zero-Knowledge. To simulate the transcript of an interaction with a malicious verifier V^*, the key idea is to use a knowledge-extractor for GI$[k]$ to recover the order in which the graphs are concatenated. More precisely, consider a simulator that runs V^* until it proves that it knows an isomorphism between $[H_0, H_1]$ and $[G_0, G_1]$. If the proof fails then the simulator aborts, otherwise it rewinds the protocol until immediately after $[H_0, H_1]$ was sent, runs the extractor for GI$[k]$ as explained in

[2] It is easy to see that the GI protocol is perfect *fully* black-box zero-knowledge with auxiliary inputs, which is a strengthening of the respective black-box notion in the sense that simulation should work even against *inefficient* verifier V^*. Furthermore, recall from the lecture that sequential composition of zero-knowledge with auxiliary inputs protocols preserves zero-knowledge with auxiliary inputs of the overall protocol; the proof of this statement is out of scope of this exercise, but interested students can refer to [Gol01, Section 4.3.4]. In this exercise we use that GI$[k]$ is perfect fully black-box zero-knowledge with auxiliary inputs (as long as k is polynomial).

the solution to Task 4.3 below to extract the isomorphism σ between the two graphs, determines b,[3] and returns $([H_0, H_1], \mathsf{trans}, b)$, where trans denotes the transcript of GI$[k]$.

Namely, the simulator proceeds as follows.

1. The simulator runs V^* to get a first message $[H_0, H_1]$.
2. The simulator runs the extractor for GI$[k]$, using V^* to generate the messages for $P^*_{GI[k]}$. If the extractor aborts, the simulator outputs $([H_0, H_1], \mathsf{trans}, \perp)$. If the extractor produces an isomorphism between $[G_0, G_1]$ and $[H_0, H_1]$, then the simulator infers b and outputs $([H_0, H_1], \mathsf{trans}, b)$.

We explain why the output of the simulator is statistically indistinguishable from a real execution between P and V^*. First, note that the first transcript trans produced by the extractor has exactly the same distribution as in a real protocol execution. The issue is that when trans is accepting, the extractor must produce a witness that will allow the simulator to continue and simulate an entire protocol transcript.

Let $\varepsilon([H_0, H_1])$ be the probability that if the first message of the protocol is $[H_0, H_1]$, then V^* convinces P in the GI$[k]$ protocol. Fix any choice of $[H_0, H_1]$. If $\varepsilon > 2^{-k}$ and trans is accepting, the extractor always produces a witness, and the simulation is distributed exactly as in a real protocol execution. If $\varepsilon \leq 2^{-k}$ and trans is accepting then we do not know how the extractor will behave. However, since $\varepsilon \leq 2^{-k}$, the probability (over $[H_0, H_1]$ and the GI$[k]$ protocol) that $\varepsilon \leq 2^{-k}$ and trans is accepting is at most 2^{-k}. If we set k to be the number of vertices in G_0 and G_1, then the probability that the simulated transcript differs from a real execution is negligible. □

4.3 (Zero-Knowledge) Proof of Knowledge for Graph Isomorphism (*)

Let GI $= (P_{\text{GI}}, V_{\text{GI}})$ be the zero-knowledge protocol for graph isomorphism seen in the 2nd lecture. Now let GI$[k] = (P_{\text{GI}[k]}, V_{\text{GI}[k]})$ be the *k-sequential repetition* of the GI protocol such that the verifier $V_{\text{GI}[k]}$ accepts if and only if the verifier V_{GI} in every single execution of the basic GI protocol accepts. Prove that GI$[k]$ is knowledge sound with knowledge error 2^{-k}.

Solution: We describe an extractor for GI$[k]$ (with prover $P^*_{\text{GI}[k]}$ and verifier $V_{\text{GI}[k]}$) which has knowledge error 2^{-k}.

1. The extractor runs the GI$[k]$ protocol (between $P^*_{\text{GI}[k]}$ and verifier $V_{\text{GI}[k]}$) once to produce a first transcript trans. If $V_{\text{GI}[k]}$ rejects trans, then the extractor aborts.
2. If $V_{\text{GI}[k]}$ does accept, then for $j = 1, \ldots, k$, the extractor rewinds $P^*_{\text{GI}[k]}$ to the computation step right after it sent its first message in the j-th repetition of GI$[k]$, and runs the j-th repetition with the *other* challenge value (using the same randomness for $P^*_{\text{GI}[k]}$).
3. If there is some j for which $P^*_{\text{GI}[k]}$ can respond to *both* challenges in GI$[k]$, the extractor extracts a witness via the 2-special soundness of the GI protocol. If not, the extractor aborts.

The running time of the extractor is at most twice that of the GI$[k]$ protocol, so the extractor runs in *strict* (rather than expected) polynomial time.

Now we analyse the success probability of the extractor. The probability that the extractor runs beyond step 1 is the same as the probability that $V_{\text{GI}[k]}$ accepts. If $P^*_{\text{GI}[k]}$ can convince $V_{\text{GI}[k]}$ with probability greater than 2^{-k}, then there must be more than one sequence of

[3] Here we assume G_0 and G_1 are connected graphs, so the extracted isomorphism σ has to map nodes $[\![n]\!]$ either to $[\![n]\!]$ (in case $b = 0$) or to $[\![n+1, 2n]\!]$ (in case $b = 1$), hence it is easy to extract b from σ.

challenges which lead to $V_{\mathrm{GI}[k]}$ accepting, and so there must be some repetition j for which $P^*_{\mathrm{GI}[k]}$ can answer both challenges. So if $V_{\mathrm{GI}[k]}$ accepts in step 1, then the extractor will always succeed in steps 2 and 3.

References

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques.* Cambridge University Press, 2001.