Modern Public Key Cryptography

Commitments and Zero-Knowledge

Daniel Kales
based on slides by Sebastian Ramacher and David Derler

Graz, April 28, 2021

SCIENCE PASSION TECHNOLOGY

Outline

- Commitment Schemes
- Interactive Proofs
- Zero-Knowledge Proofs

Commitment Schemes

Commitments - Informal Idea

Imagine two parties A and B

- A makes some (secret) decision m
- A wants to later convince B that decision m was made
  - A must not be able to change m later on
  - B must not be able to learn anything about m before

Real world analogy

- A writes m on a piece of paper,
  - puts it in a box and locks the box
- A hands the locked box to B
  - Later, A can give the key for the box to B

Commitments - Informal Idea

Imagine two parties A and B

- A makes some (secret) decision m
- A wants to later convince B that decision m was made
  - A must not be able to change m later on
  - B must not be able to learn anything about m before

Real world analogy

- A writes m on a piece of paper,
  - puts it in a box and locks the box
- A hands the locked box to B
  - Later, A can give the key for the box to B

Commitment Scheme

Commitment Scheme

Gen(1^κ): This probabilistic algorithm on input of κ, outputs (public) parameters pp.

Commit(pp, m): This (probabilistic) algorithm on input pp and message $m \in \mathcal{M}$, outputs commitment C and opening information O.

Open(pp, C, O) : This deterministic algorithm on input pp C and O returns $m \in M \cup \{\perp\}$.

- pp may be generated by a trusted third party (TTP) or one of the parties

Security

Binding

- Recall: A must not be able to change m later on

More formally: $\forall$ PPT $\mathcal{A}$ $\exists$ negl. $\epsilon(\cdot)$ such that

$$\Pr\left[\begin{array}{c} \text{pp} \leftarrow \mathit{Gen}(1^\kappa), (C^*, O^*, O'^*) \leftarrow \mathcal{A}(\text{pp}), \\ m \leftarrow \mathit{Open}(\text{pp}, C^*, O^*), \\ m' \leftarrow \mathit{Open}(\text{pp}, C^*, O'^*) : \\ m \neq m' \ \wedge \ m \neq \bot \ \wedge \ m' \neq \bot \end{array}\right] \leq \epsilon(\kappa).$$

Security II

Hiding

- Recall: B must not be able to learn anything about m

More formally: $\forall$ PPT $\mathcal{A}$ $\exists$ negl. $\epsilon(\cdot)$ such that

$$\Pr\left[\begin{array}{c} \text{pp} \leftarrow \mathit{Gen}(1^\kappa), (m_0, m_1, \text{state}) \leftarrow \mathcal{A}(\text{pp}), \\ b \xleftarrow{R} \{0,1\}, C_b \leftarrow \mathit{Commit}(\text{pp}, m_b), \\ b^* \leftarrow \mathcal{A}(\text{state}, C_b) : b = b^* \end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa).$$

Discrete Log Commitment

Scheme

$$Gen(1^\kappa): \text{ Set pp} \leftarrow \mathcal{G}^\kappa = (\mathbb{G}, p, g) \text{ and return pp.}$$

$$Commit(\text{pp}, m): \text{ Return } C \leftarrow g^m, O \leftarrow m$$

$$Open(\text{pp}, C, O): \text{ If } C = g^m \text{ return } m \text{ and } \perp \text{ otherwise.}$$

- Binding holds unconditional (only single m satisfies $C = g^m$)
- Hiding holds computational under DL (clearly, only for unpredictable messages)

Discrete Log Commitment

Scheme

$Gen(1^\kappa):$ Set pp $\leftarrow \mathcal{G}^\kappa = (\mathbb{G}, p, g)$ and return pp.

$Commit(\mathrm{pp}, m):$ Return $C \leftarrow g^m, O \leftarrow m$

$Open(\mathrm{pp}, C, O):$ If $C = g^m$ return m and $\perp$ otherwise.

- Binding holds unconditional (only single m satisfies $C = g^m$)
- Hiding holds computational under DL (clearly, only for unpredictable messages)

Pedersen Commitment

Scheme

$Gen(1^\kappa)$: Choose $\mathcal{G}^\kappa = (\mathbb{G}, p, g), h \xleftarrow{R} \mathbb{G}$ and return $\text{pp} \leftarrow (\mathcal{G}^\kappa, h)$.

$Commit(\text{pp}, m)$: Choose $r \xleftarrow{R} \mathbb{Z}_p$ and return $C \leftarrow g^m h^r, O \leftarrow (m, r)$

$Open(\text{pp}, C, O)$: If $C = g^m h^r$ return m and $\perp$ otherwise.

- Binding holds under DL (recall first lecture & exercise)
- Hiding holds unconditional ($\forall C\, \forall m\, \exists$unique $r : C = g^m h^r$)
- Who can generate the pp?

Pedersen Commitment

Scheme

$$Gen(1^\kappa): \text{ Choose } \mathcal{G}^\kappa = (\mathbb{G}, p, g), h \xleftarrow{R} \mathbb{G} \text{ and return pp} \leftarrow (\mathcal{G}^\kappa, h).$$

$$Commit(\mathsf{pp}, m): \text{ Choose } r \xleftarrow{R} \mathbb{Z}_p \text{ and return } C \leftarrow g^m h^r, O \leftarrow (m, r)$$

$$Open(\mathsf{pp}, C, O): \text{ If } C = g^m h^r \text{ return } m \text{ and } \perp \text{ otherwise.}$$

- Binding holds under DL (recall first lecture & exercise)
- Hiding holds unconditional ($\forall C\, \forall m\, \exists$unique $r : C = g^m h^r$)
- Who can generate the pp?

Unconditional vs. Computational Security

There is no scheme (in the classical setting) providing

- unconditional hiding and
- unconditional binding

at the same time.

Why? (Recall exercises.)

Unconditional vs. Computational Security

There is no scheme (in the classical setting) providing

- unconditional hiding and
- unconditional binding

at the same time.

Why? (Recall exercises.)

Commitments from Encryption Schemes

Assume an IND-CPA secure encryption scheme

- $\Pi = (Gen, Enc, Dec)$

Commitment Scheme from Π

$Gen(1^\kappa)$: Run $(\mathrm{sk}, \mathrm{pk}) \leftarrow Gen(1^\kappa)$ and return pk.

$Commit(\mathrm{pp}, m)$: Randomly choose r and return $C \leftarrow Enc(\mathrm{pk}, m;\ r), O \leftarrow (m, r)$

$Open(\mathrm{pp}, C, O)$: If $C = Enc(\mathrm{pk}, m;\ r)$ return m and $\perp$ otherwise.

Commitments from Encryption Schemes II

- Binding follows from perfect correctness
  - Correctness states

$$\forall(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathit{Gen}(1^\kappa), \forall m : m = \mathit{Dec}(\mathrm{sk}, \mathit{Enc}(\mathrm{pk}, m))$$

  - Breaking binding implies that
    - $\mathit{Enc}(\mathrm{pk}, m_0) = \mathit{Enc}(\mathrm{pk}, m_1)$, for a fixed pk
    - But then we have $m_1 = \mathit{Dec}(\mathrm{sk}, \mathit{Enc}(\mathrm{pk}, m_0))$
- Hiding follows from IND-CPA security
  - $\mathcal{A}$ who breaks hiding can be used to break IND-CPA

Commitments from Encryption Schemes II

- Binding follows from perfect correctness
  - Correctness states

$$\forall(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathit{Gen}(1^\kappa), \forall m : m = \mathit{Dec}(\mathrm{sk}, \mathit{Enc}(\mathrm{pk}, m))$$

  - Breaking binding implies that
    - $\mathit{Enc}(\mathrm{pk}, m_0) = \mathit{Enc}(\mathrm{pk}, m_1)$, for a fixed pk
    - But then we have $m_1 = \mathit{Dec}(\mathrm{sk}, \mathit{Enc}(\mathrm{pk}, m_0))$
- Hiding follows from IND-CPA security
  - $\mathcal{A}$ who breaks hiding can be used to break IND-CPA

Interactive Proofs

Efficiently Verifiable Proofs

- $\mathcal{NP}$ is the set of decision problems
  - where valid instances have efficiently verifiable proofs
- For any such problem S there is a deterministic polynomial time verifier
  - such that for any instance $x \in S$
  - there exists an algorithm (the prover) that provides a polynomial sized witness w ($\mathcal{NP}$ witness)
  - such that the verifier accepts on input (x, w) iff $x \in S$

Example: Graph Isomorphism (GI)

- Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a bijection $\pi : V_1 \mapsto V_2$ s.t.

$$\{u, v\} \in E_1 \iff \{\pi(u), \pi(v)\} \in E_2$$

- Language $L_{GI} = \{(G_1, G_2) | G_1 \cong G_2\}$ is in $\mathcal{NP}$ (witness π)

Interactive Proofs

- What if we allow the verifier to adaptively ask the prover?
  - Does not give a benefit (we can define an equivalent non-interactive verifier that takes a transcript)
- Allow an interactive verifier to be probabilistic?
  - Gives more power - yields the class $\mathcal{IP}$ ($\mathcal{IP}$ = PSPACE)
- Consider game between computationally bounded verifier $\mathcal{V}$ (PPT) and computationally unbounded prover $\mathcal{P}$
  - Prover convinces the verifier of the validity of some assertion ($x \in S$)

Interactive Proofs

- What if we allow the verifier to adaptively ask the prover?
  - Does not give a benefit (we can define an equivalent non-interactive verifier that takes a transcript)
- Allow an interactive verifier to be probabilistic?
  - Gives more power - yields the class $\mathcal{IP}$ ($\mathcal{IP} =$ PSPACE)
- Consider game between computationally bounded verifier $\mathcal{V}$ (PPT) and computationally unbounded prover $\mathcal{P}$
  - Prover convinces the verifier of the validity of some assertion ($x \in S$)

Interactive Proofs

- What if we allow the verifier to adaptively ask the prover?
  - Does not give a benefit (we can define an equivalent non-interactive verifier that takes a transcript)
- Allow an interactive verifier to be probabilistic?
  - Gives more power - yields the class $\mathcal{IP}$ ($\mathcal{IP} = \text{PSPACE}$)
- Consider game between computationally bounded verifier $\mathcal{V}$ (PPT) and computationally unbounded prover $\mathcal{P}$
  - Prover convinces the verifier of the validity of some assertion ($x \in S$)

Interactive Proofs: Formalization

Interactive Proof System (IPS)

An IPS for a language L is an interactive protocol between an unrestricted prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$ such that on input x the following conditions hold:

Completeness: $\forall x \in L\colon \Pr[(\mathcal{P}, \mathcal{V})(x) \text{ accepts}] = 1$

Soundness: $\forall x \notin L, \forall \mathcal{P}^*\colon \Pr[(\mathcal{P}^*, \mathcal{V})(x) \text{ accepts}] \leq \frac{1}{2}$

- Perfect completeness (imperfect may have error probability)
- Interactive arguments: computational soundness ($\mathcal{P}^*$ is PPT)
- Reduce soundness error by sequential/parallel repetition

Interactive Proofs: Formalization

Interactive Proof System (IPS)

An IPS for a language L is an interactive protocol between an unrestricted prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$ such that on input x the following conditions hold:

Completeness: $\forall x \in L$: $\Pr[(\mathcal{P}, \mathcal{V})(x) \ \texttt{accepts}] = 1$

Soundness: $\forall x \notin L, \forall \mathcal{P}^*$: $\Pr[(\mathcal{P}^*, \mathcal{V})(x) \ \texttt{accepts}] \leq \frac{1}{2}$

- Perfect completeness (imperfect may have error probability)
- Interactive arguments: computational soundness ($\mathcal{P}^*$ is PPT)
- Reduce soundness error by sequential/parallel repetition

Example: Graph Non-Isomorphism (GNI)

- $L_{GNI} = \{(G_1, G_2) \mid |G_1| = |G_2|, G_1 \ncong G_2\}$
- Unknown if $L_{GNI} \in \mathcal{NP}$ (clearly in co-$\mathcal{NP}$), but it is in $\mathcal{IP}$

IP for L_{GNI}

Let $x = (G_1, G_2)$ be the common input

$\mathcal{V}$: Pick $i \xleftarrow{R} \{1, 2\}$, randomly permute vertices of G_i and send to $\mathcal{P}$

$\mathcal{P}$: Find $b \in \{1, 2\}$ s.t. $G_i \cong G_b$ and send b (note that $\mathcal{P}$ is unbounded)

$\mathcal{V}$: Accept if $b = i$

- If $G_1 \ncong G_2$, any permutation of G_i uniquely determines i
- If $G_1 \cong G_2$, distribution of G_i independent of i

Example: Graph Non-Isomorphism (GNI)

- $L_{GNI} = \{(G_1, G_2) \mid |G_1| = |G_2|, G_1 \ncong G_2\}$
- Unknown if $L_{GNI} \in \mathcal{NP}$ (clearly in co-$\mathcal{NP}$), but it is in $\mathcal{IP}$

IP for L_{GNI}

Let $x = (G_1, G_2)$ be the common input

$\mathcal{V}$: Pick $i \xleftarrow{R} \{1, 2\}$, randomly permute vertices of G_i and send to $\mathcal{P}$

$\mathcal{P}$: Find $b \in \{1, 2\}$ s.t. $G_i \cong G_b$ and send b (note that $\mathcal{P}$ is unbounded)

$\mathcal{V}$: Accept if $b = i$

- If $G_1 \ncong G_2$, any permutation of G_i uniquely determines i
- If $G_1 \cong G_2$, distribution of G_i independent of i

Example: Graph Non-Isomorphism (GNI)

- $L_{GNI} = \{(G_1, G_2) \mid |G_1| = |G_2|, G_1 \ncong G_2\}$
- Unknown if $L_{GNI} \in \mathcal{NP}$ (clearly in co-$\mathcal{NP}$), but it is in $\mathcal{IP}$

IP for L_{GNI}

Let $x = (G_1, G_2)$ be the common input

$\mathcal{V}$: Pick $i \xleftarrow{R} \{1, 2\}$, randomly permute vertices of G_i and send to $\mathcal{P}$

$\mathcal{P}$: Find $b \in \{1, 2\}$ s.t. $G_i \cong G_b$ and send b (note that $\mathcal{P}$ is unbounded)

$\mathcal{V}$: Accept if $b = i$

- If $G_1 \ncong G_2$, any permutation of G_i uniquely determines i
- If $G_1 \cong G_2$, distribution of G_i independent of i

Zero-Knowledge Proofs

Zero-Knowledge Proofs

- Does proving the validity of an assertion always require giving away extra knowledge?
  - No, captured by zero-knowledge
- No adversary can gain anything from a prover (beyond being convinced of the validity of an assertion)
- How to model this requirement?
  - All an adversarial verifier can learn from interacting with the prover can be learned based on the assertion itself
  - Transcripts of real interactions not distinguishable from "simulated" interactions (on only public input)

Zero-Knowledge Proofs

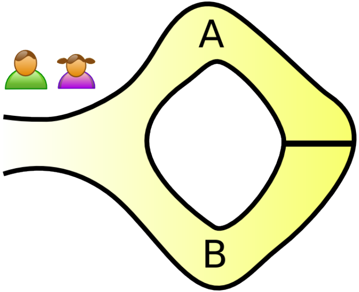
- Does proving the validity of an assertion always require giving away extra knowledge?
  - No, captured by zero-knowledge
- No adversary can gain anything from a prover (beyond being convinced of the validity of an assertion)
- How to model this requirement?
  - All an adversarial verifier can learn from interacting with the prover can be learned based on the assertion itself
  - Transcripts of real interactions not distinguishable from "simulated" interactions (on only public input)

Zero-Knowledge Proofs

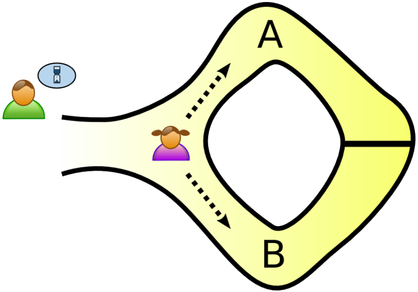
- Does proving the validity of an assertion always require giving away extra knowledge?
  - No, captured by zero-knowledge
- No adversary can gain anything from a prover (beyond being convinced of the validity of an assertion)
- How to model this requirement?
  - All an adversarial verifier can learn from interacting with the prover can be learned based on the assertion itself
  - Transcripts of real interactions not distinguishable from "simulated" interactions (on only public input)

Story of Ali Baba

- Alice knows a secret word to open a magic door in a cave
- Alices wants to convince Bob that she knows the secret
- But Alice does not want to reveal the secret word, nor for anyone else to find out about her skills (paparazzi)



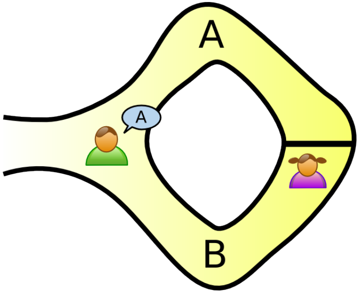
Story of Ali Baba



http://en.wikipedia.org/

1. Alice randomly chooses path A or B, while Bob waits outside.
2. Bob chooses an exit path.
3. Alice reliably appears at the exit Bob names.
4. An observer Otto won’t be convinced – prior agreement?

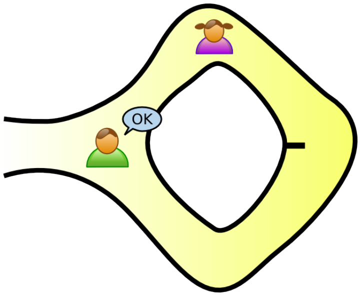
Story of Ali Baba



http://en.wikipedia.org/

1. Alice randomly chooses path A or B, while Bob waits outside.
2. Bob chooses an exit path.
3. Alice reliably appears at the exit Bob names.
4. An observer Otto won't be convinced – prior agreement?

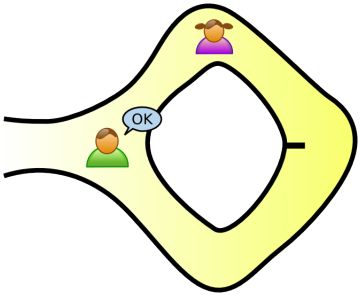
Story of Ali Baba



http://en.wikipedia.org/

1. Alice randomly chooses path A or B, while Bob waits outside.
2. Bob chooses an exit path.
3. Alice reliably appears at the exit Bob names.
4. An observer Otto won't be convinced – prior agreement?

Story of Ali Baba



http://en.wikipedia.org/

1. Alice randomly chooses path A or B, while Bob waits outside.
2. Bob chooses an exit path.
3. Alice reliably appears at the exit Bob names.
4. An observer Otto won’t be convinced – prior agreement?

Zero-Knowledge Proofs: Formalization

Perfect Zero-Knowledge

An IPS for a language L is said to provide perfect zero-knowledge, if for every PPT $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S}$ s.t.

$$(\mathcal{P}, \mathcal{V}^*)(x) \equiv \mathcal{S}(x), \text{ for every } x \in S$$

- Statistical ZK: distributions are statistically close
- Computational ZK: distributions cannot be told apart by efficient distinguishers - computationally indistinguishable

Zero-Knowledge Proofs: Formalization

Perfect Zero-Knowledge

An IPS for a language L is said to provide perfect zero-knowledge, if for every PPT $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S}$ s.t.

$$(\mathcal{P}, \mathcal{V}^*)(x) \equiv \mathcal{S}(x), \text{ for every } x \in S$$

- Statistical ZK: distributions are statistically close
- Computational ZK: distributions cannot be told apart by efficient distinguishers - computationally indistinguishable

Zero-Knowledge Proofs: Formalization

- ZK
  - We do not know how $\mathcal{V}^*$ exactly behaves
  - $\mathcal{S}$ needs to exist for arbitrary $\mathcal{V}^*$
  - So, we consider black-box access to $\mathcal{V}^*$ in the simulation
- Honest-verifier ZK
  - We assume $\mathcal{V}^*$ behaves honestly
  - Consequently, $\mathcal{S}$ ignores $\mathcal{V}^*$ in the simulation

Example: ZK Proof for GI

ZKP for GI

Let the common input be a pair of graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and let φ be an arbitrary isomorphism between them

$\mathcal{P}$: Choose random permutation π and send $G' = (V_2, E)$ with $E = \{(\pi(u), \pi(v)) | \{u, v\} \in E_2\}$ to $\mathcal{V}$ (if $G_1 \cong G_2$ this graph is isomorphic to both)

$\mathcal{V}$: Choose $b \xleftarrow{R} \{1, 2\}$ and ask $\mathcal{P}$ to reveal an isomorphism between G' and G_b

$\mathcal{P}$: If $b = 2$ send $\psi \leftarrow \pi$, otherwise send $\psi \leftarrow \pi \circ \varphi$ to $\mathcal{V}$

$\mathcal{V}$: If received ψ is isomorphism between G' and G_b output `accept` and `reject` otherwise

- Honest prover always succeeds; cheating prover will succeed with prob. $1/2$ (correctly guess the bit of $\mathcal{V}$)

Example: ZK Proof for GI

ZKP for GI

Let the common input be a pair of graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and let φ be an arbitrary isomorphism between them

$\mathcal{P}$: Choose random permutation π and send $G' = (V_2, E)$ with $E = \{(\pi(u), \pi(v)) | \{u, v\} \in E_2\}$ to $\mathcal{V}$ (if $G_1 \cong G_2$ this graph is isomorphic to both)

$\mathcal{V}$: Choose $b \xleftarrow{R} \{1, 2\}$ and ask $\mathcal{P}$ to reveal an isomorphism between G' and G_b

$\mathcal{P}$: If $b = 2$ send $\psi \leftarrow \pi$, otherwise send $\psi \leftarrow \pi \circ \varphi$ to $\mathcal{V}$

$\mathcal{V}$: If received ψ is isomorphism between G' and G_b output `accept` and `reject` otherwise

- Honest prover always succeeds; cheating prover will succeed with prob. $1/2$ (correctly guess the bit of $\mathcal{V}$)

Example: ZK Proof for GI

- The GI protocol is honest-verifier ZK
  - $\mathcal{S}$ chooses b and ψ uniformly at random and outputs (G', b, ψ) with G' being ψ applied to G_b
- The GI protocol is perfect ZK
  - Let b^* be the random choice of $\mathcal{V}^*$
  - $\mathcal{S}$ selects $b \xleftarrow{R} \{1, 2\}$ (hoping $\mathcal{V}^*$ selects $b^* = b$)
  - $\mathcal{S}$ constructs G' by permuting G_b under random ψ
  - If $b^* \neq b$, $\mathcal{S}$ restarts, otherwise output (G', b, ψ)
  - Output of $\mathcal{S}$ is perfectly indistinguishable from real (note b^* is independent of b) and we expect a valid transcript every two runs (poly time $\mathcal{S}$)

Example: ZK Proof for GI

- The GI protocol is honest-verifier ZK
  - $\mathcal{S}$ chooses b and ψ uniformly at random and outputs (G', b, ψ) with G' being ψ applied to G_b
- The GI protocol is perfect ZK
  - Let b^* be the random choice of $\mathcal{V}^*$
  - $\mathcal{S}$ selects $b \xleftarrow{R} \{1, 2\}$ (hoping $\mathcal{V}^*$ selects $b^* = b$)
  - $\mathcal{S}$ constructs G' by permuting G_b under random ψ
  - If $b^* \neq b$, $\mathcal{S}$ restarts, otherwise output (G', b, ψ)
  - Output of $\mathcal{S}$ is perfectly indistinguishable from real (note b^* is independent of b) and we expect a valid transcript every two runs (poly time $\mathcal{S}$)

Zero-Knowledge for $\mathcal{NP}$

- ZK proofs exist for all $L \in \mathcal{NP}$
- Recall $\mathcal{NP}$-completeness
  - A problem is $\mathcal{NP}$ complete if it is in $\mathcal{NP}$
  - and every problem in $\mathcal{NP}$ is poly time reducible to it
- ZK proof for $\mathcal{NP}$-complete language $\mathbf{L}$ (e.g., graph 3-coloring)
  - Reduce L to $\mathbf{L}$ (and the witness)
  - Run ZK proof for $\mathbf{L}$

Proofs of Knowledge

- ZKPs only interested in the validity of the assertion itself
- Proofs of knowledge (PoKs) capture IPs where $\mathcal{P}$ asserts knowledge of some object (e.g., a particular isomorphism)
- What does it mean for a machine M to know something?
  - There exists an efficient machine $\mathcal{E}$, which, given black-box access to M can extract M's "knowledge" (a string)
- PoK: Whenever there is a $\mathcal{P}^*$ that convinces $\mathcal{V}$ to know something, we can extract this string from $\mathcal{P}^*$
- Stronger notion of soundness

Proofs of Knowledge

- ZKPs only interested in the validity of the assertion itself
- Proofs of knowledge (PoKs) capture IPs where $\mathcal{P}$ asserts knowledge of some object (e.g., a particular isomorphism)
- What does it mean for a machine M to know something?
  - There exists an efficient machine $\mathcal{E}$, which, given black-box access to M can extract M's "knowledge" (a string)
- PoK: Whenever there is a $\mathcal{P}^*$ that convinces $\mathcal{V}$ to know something, we can extract this string from $\mathcal{P}^*$
- Stronger notion of soundness

Proofs of Knowledge: Formalization

- Consider an $\mathcal{NP}$ relation $R = \{(x, w) | W(x, w) = \texttt{accept}\}$ where W is a PT algorithm deciding membership in R
- We can write $L_R = \{x | \exists w \text{ s.t. } (x, w) \in R\}$

Proof of Knowledge (PoK)

Let $(\mathcal{P}, \mathcal{V})$ be an IPS for L_R. Then, $(\mathcal{P}, \mathcal{V})$ is a PoK with knowledge error ρ if there exists a PPT knowledge extractor $\mathcal{E}$ such that for any $x \in L_R$ and any PPT $\mathcal{P}^*$ with $\delta = \Pr[(\mathcal{P}^*, \mathcal{V})(x) \texttt{ accepts}] > \rho$, we have that

$$\Pr[w \leftarrow \mathcal{E}^{P^*}(x) : R(x, w) = \texttt{accept}] \geq \text{poly}(\delta - \rho)$$

Proofs of Knowledge: Formalization

- Consider an $\mathcal{NP}$ relation $R = \{(x, w) | \ W(x, w) = \texttt{accept}\}$ where W is a PT algorithm deciding membership in R
- We can write $L_R = \{x | \ \exists w \text{ s.t. } (x, w) \in R\}$

Proof of Knowledge (PoK)

Let $(\mathcal{P}, \mathcal{V})$ be an IPS for L_R. Then, $(\mathcal{P}, \mathcal{V})$ is a PoK with knowledge error ρ if there exists a PPT knowledge extractor $\mathcal{E}$ such that for any $x \in L_R$ and any PPT $\mathcal{P}^*$ with $\delta = \Pr[(\mathcal{P}^*, \mathcal{V})(x) \ \texttt{accepts}] > \rho$, we have that

$$\Pr[w \leftarrow \mathcal{E}^{P^*}(x) : R(x, w) = \texttt{accept}] \geq \text{poly}(\delta - \rho)$$

Some Notes

- Non-interactive ZK (Single message)
  - In the common reference string model
  - General constructions very inefficient
- Witness indistinguishability (Relaxation of ZK)
  - For $\mathcal{NP}$ relation R no $\mathcal{V}^*$ can distinguish if $\mathcal{P}$ uses witness w_1 or w_2 to x with $(x, w_i) \in R$ for $i \in \{1, 2\}$
- Public coin (e.g., GI) vs. private coin (e.g., GNI - our version is not ZK - but a slightly modified one)
- What we have seen so far is mainly of theoretical interest
- Will see (NI)-ZKPoKs that are useful and efficient

Some Notes

- Non-interactive ZK (Single message)
  - In the common reference string model
  - General constructions very inefficient
- Witness indistinguishability (Relaxation of ZK)
  - For $\mathcal{NP}$ relation R no $\mathcal{V}^*$ can distinguish if $\mathcal{P}$ uses witness w_1 or w_2 to x with $(x, w_i) \in R$ for $i \in \{1, 2\}$
- Public coin (e.g., GI) vs. private coin (e.g., GNI - our version is not ZK - but a slightly modified one)
- What we have seen so far is mainly of theoretical interest
- Will see (NI)-ZKPoKs that are useful and efficient

Some Notes

- Non-interactive ZK (Single message)
  - In the common reference string model
  - General constructions very inefficient
- Witness indistinguishability (Relaxation of ZK)
  - For $\mathcal{NP}$ relation R no $\mathcal{V}^*$ can distinguish if $\mathcal{P}$ uses witness w_1 or w_2 to x with $(x, w_i) \in R$ for $i \in \{1, 2\}$
- Public coin (e.g., GI) vs. private coin (e.g., GNI - our version is not ZK - but a slightly modified one)
- What we have seen so far is mainly of theoretical interest
- Will see (NI)-ZKPoKs that are useful and efficient

Some Notes

- Non-interactive ZK (Single message)
  - In the common reference string model
  - General constructions very inefficient
- Witness indistinguishability (Relaxation of ZK)
  - For $\mathcal{NP}$ relation R no $\mathcal{V}^*$ can distinguish if $\mathcal{P}$ uses witness w_1 or w_2 to x with $(x, w_i) \in R$ for $i \in \{1, 2\}$
- Public coin (e.g., GI) vs. private coin (e.g., GNI - our version is not ZK - but a slightly modified one)
- What we have seen so far is mainly of theoretical interest
- Will see (NI)-ZKPoKs that are useful and efficient

Some Notes

- Non-interactive ZK (Single message)
  - In the common reference string model
  - General constructions very inefficient
- Witness indistinguishability (Relaxation of ZK)
  - For $\mathcal{NP}$ relation R no $\mathcal{V}^*$ can distinguish if $\mathcal{P}$ uses witness w_1 or w_2 to x with $(x, w_i) \in R$ for $i \in \{1, 2\}$
- Public coin (e.g., GI) vs. private coin (e.g., GNI - our version is not ZK - but a slightly modified one)
- What we have seen so far is mainly of theoretical interest
- Will see (NI)-ZKPoKs that are useful and efficient

Questions?

Further Reading I

[1] Mihir Bellare and Oded Goldreich.

On defining proofs of knowledge.

In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 390–420, 1992.

[2] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie.

Efficient zero-knowledge proofs of knowledge without intractability assumptions.

In *Public Key Cryptography, Third International Workshop on Practice and Theory in Public Key Cryptography, PKC 2000, Melbourne, Victoria, Australia, January 18-20, 2000, Proceedings*, pages 354–373, 2000.

[3] Ivan Damgard.

On Σ-protocols.

http://cs.au.dk/~ivan/Sigma.pdf.

[4] Juan A. Garay, Philip D. MacKenzie, and Ke Yang.

Strengthening zero-knowledge protocols using signatures.

J. Cryptology, 19(2):169–209, 2006.

Further Reading II

[5] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi.

Zkboo: Faster zero-knowledge for boolean circuits.

In *USENIX Security*, 2016.

[6] Oded Goldreich.

Computational Complexity - A Conceptual Perspective.

Cambridge University Press, 2008.

[7] Jens Groth and Amit Sahai.

Efficient non-interactive proof systems for bilinear groups.

Cryptology ePrint Archive, Report 2007/155, 2007.

http://eprint.iacr.org/2007/155.

[8] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai.

Zero-knowledge from secure multiparty computation.

In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30, 2007.