MC525: Cryptography
#12: Zero-Knowledge Proofs

Sang-Hyun Yoon

Zero-Knowledge Proof Systems: Informal Definition

Very informally, a zero-knowledge proof system is an interactive protocol between two parties, a prover and a verifier, in which:

- Both parties have in input a proposition (that is true/false).
  - e.g. a graph G and "G is 3-colorable", or

    $N, r > 0$ and "there is an integer x s.t. $x^2 \bmod N = r$".

1. If the proposition is true, then then prover can prove to the verifier that the proposition is true (**completeness**)
2. If the proposition is false, then then prover cannot cheat the verifier that the proposition is true (**soundness**)
3. without revealing any additional information beyond the truth of the proposition (**zero-knowledge**)
   - i.e. verifier alone cannot still prove the proposition

Outline

(Algorithmic) Problems vs. Propositions vs. Languages

Algorithmic problems (decision version)

- $f : X \to \{\mathtt{T}, \mathtt{F}\}$
  - e.g. $f_{GI}(G_1, G_2) = \begin{cases} \mathtt{T} & \text{if graphs } G_1 \text{ and } G_2 \text{ are isomorphic} \\ \mathtt{F} & \text{otherwise} \end{cases}$
  - c.f. certificate, witness, proof

Propositions

- input instances of the decision-version of the ATP problem
  - $f(\text{proposition}) =$
    $$\begin{cases} \mathtt{T} & \text{if provable from ZFC (i.e. true in all models of ZFC)} \\ \mathtt{F} & \text{if disprovable from ZFC (i.e. false in all models of ZFC)} \\ ? & \text{if independent from ZFC} \end{cases}$$

Languages (history of computation DFA/NFA/PDA/TM..)

- $L = f^{-1}(\mathtt{T})$ encoded with 0/1s (i.e. $L \subseteq \{0, 1\}^*$)
  - e.g. $L_{GI} = \{(G_1, G_2) \,|\, \text{graphs } G_1 \text{ and } G_2 \text{ are isomorphic}\}$

(Algorithmic) Problems vs. Languages

Let $f : X \to \{\mathtt{T}, \mathtt{F}\}$ be an algorithmic problem where X is infinite.

- 모든 $x \in X$의 크기는 유한해야 하므로 X는 countable set
- 따라서, 임의의 bijection $\phi : X \to \{0,1\}^*$을 이용하여 f를 binary encoding 할 수 있다
- $L = \phi(f^{-1}(T)) \subseteq \{0,1\}^*$ 로 두면

$$x \in L \iff f\left(\phi^{-1}(x)\right) = \mathtt{T}$$
$$x \notin L \iff f\left(\phi^{-1}(x)\right) = \mathtt{F}$$

- 임의의 bijection $\phi, \phi' \in X \to \{0,1\}^*$에 대해 ϕ, ϕ'가 poly-time computable이면 $\phi(x) \mapsto \phi'(x)$, $\phi'(x) \mapsto \phi(x)$ mapping도
- 즉, 어떤 encoding을 사용해도 계산복잡도 측면에서 무관 (i.e. $L = \phi\left(f^{-1}(T)\right)$와 $L' = \phi'\left(f^{-1}(T)\right)$는 isomorphic)
- $L \in 2^{\{0,1\}^*}$므로 $f \in 2^{\{0,1\}^*}$로 취급할 수 있고, problem과 (isomorphic) language간은 섞어서 사용

Interactive Turing Machines

Definition (Interactive Turing Machine (ITM))

$M(x, m)$ 형태의 TM (보조입력 z가 추가될 수도 있음)

- x는 아래에서 common input, m은 상대 TM의 output
- Internal state variable도 가질 수 있음 (즉, pure function 아님)
- may be randomized (with random number generator for coin-toss)

Definition (Interactive Computation of two ITMs)

Given two ITMs P, V and common input x, the result of the interactive computation, written $\langle P, V\rangle(x)$, is the return value of

$m_v := \epsilon$
`while` $m_v \notin \{\text{"T"}, \text{"F"}\}$ `# "accept"/"reject"`
  $m_p := P(x, m_v)$
  $m_v := V(x, m_p)$
`return` m_v

If P, V are randomized, then $\langle P, V\rangle(x)$ is a random variable

Interactive Proof Systems

Fix a language $L \in 2^{\{0,1\}^*}$

Definition (Interactive Proof System)

An interactive proof system for L is a pair of two ITMs (P, V) s.t.

- $\forall x \in L,\ \Pr\left[\langle P, V\rangle(x) = \text{"T"}\right] = 1$ (completeness)
  - P, V가 randomized인 것을 허용하여서 확률적으로 정의
  - $\ldots = 1$을 $\ldots = 1 - \epsilon(|x|)$로 relax할 수도 있음
- $\forall P^* \forall x \notin L,\ \Pr\left[\langle P^*, V\rangle(x) = \text{"F"}\right] \geq 1 - \epsilon(|x|)$ (soundness)
  - where $0 \leq \epsilon(|x|) < 1/p(|x|)$ for every polynomial function $p(\cdot)$
  - 위와 달리 모든 가능한 (prover) P^*에 대해 성립해야 함
  - (Fake prover도 $\epsilon(|x|)$만큼은 속일 수 있어서) $1 - \epsilon(|x|)$로
- V is a (probabilistic) polynomial-time TM
  - whereas no time-bound placed on P (may be exp-time TM)

Such P is called a prover, and V a verifier

Interactive Proof $\wedge$ Zero-Knowledge $=$ Zero-Knowledge Proof

Interactive Proof Systems: Trivial Cases (1/2)

Every language $L \in \mathcal{P}$ has an interactive proof system

- Let $\mathcal{A}_L$ be any poly-t. algorithm for (problem $\equiv$ language) L
- Let $V(x, m) = \mathcal{A}_L(x)$ (just ignore m)
- Then, for any ITM P, (P, V) is an interactive proof system
  - completeness, soundness, deterministic poly-time

In the same say, every language in $\mathcal{BPP}$ also has an interactive proof system

Interactive Proof Systems: Trivial Cases (2/2)

Every language in $L \in \mathcal{NP}$ has an interactive proof

- Let C_L be any poly-time certifier for L
  - i.e. for each $x \in L$, there is (certifier/proof) y s.t. $C_L(x, y) = \mathtt{T}$
  - i.e. for each $x \in L$, there is no y s.t. $C_L(x, y) = \mathtt{T}$
- Let $P(x, m) = \begin{cases} (\text{proof})\ y \ \text{ s.t. } C_L(x, y) = \mathtt{T} & \text{if } x \in L \\ \mathtt{None} & \text{otherwise} \end{cases}$
  - recall: prover P는 exp-time에 수행되는 것이 허용됨
- Let $V(x, m) = C_L(x, m)$ (i.e. just check if indeed a proof!)
- Then, (P, V) is an interactive proof system
  - complete: $x \in L$면 $V(x, P(x, \cdot)) = C_L(x, \text{proof of } x) = T$
  - sound: $x \notin L$면 $V(x, P^*(x, \cdot)) = F$ for all P^* (x has no proof)
  - poly-time verifier: $V = C_L$ is a poly-time TM

The above (P, V) is not a zero-knowledge proof system (stay tuned)

The Class $\mathcal{IP}$

Definition ($\mathcal{IP}$)

- $\mathcal{IP} \triangleq \{L \in 2^{\{0,1\}^*} \mid L \text{ has an interactive proof system}\}$
  - not the languages with zero-knowledge proof systems

- $\mathcal{BPP} \cup \mathcal{NP} \subseteq \mathcal{IP}$
  - Remind: it is not known whether or not $\mathcal{BPP} \subseteq \mathcal{NP}$

Theorem (Shamir, 1992)

$$\mathcal{IP} = \mathcal{PSPACE}$$

Outline

Recall: Isomorphism of Graphs

Definition (Graph isomorphism)

Two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are said to be isomorphic, written $G_1 \approx G_2$, if

- $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(a, b) \in E_1 \Leftrightarrow (\pi(a), \pi(b)) \in E_2$
  - such π is called an isomorphism

- Given a graph $G = (V, E)$ and a bijection $\pi : V \to V'$, $\pi(G)$ represents a graph $G' = \Big(V',\ \{(\pi(u), \pi(v)) \,\big|\, (u, v) \in E\}\Big)$

Computational complexity on graph isomorphism:

- $\textsc{Graph-Isomorphism} \in \mathcal{NP}$
- $\textsc{Graph-Non-Isomorphism} \in \text{co-}\mathcal{NP}$
- not known: $\textsc{Graph-Isomorphism} \in \mathcal{NP}$-hard or not
- not known: $\textsc{Graph-Isomorphism} \in \text{co-}\mathcal{NP}$

  $\textsc{Graph-Non-Isomorphism} \in \mathcal{NP}$

An Interactive Proof Systems for GNI (1/4)

Example

- Peggy knows an experimental procedure to distinguish between Korean/imported beef. (e.g. DNA test)
- Peggy wishes to prove to Victor that she knows the experimental procedure
  - so that she sells the technology to Victor at a high rate.
- But Peggy wants not to reveal any information about the experimental procedure
  - apart from the fact that she knows it.
- A zero-knowledge proof for Peggy: blind test (> 100 times)

An Interactive Proof Systems for GNI (2/4)

The Graph Non-Isomorphism (GNI) Problem

$L_{GNI} = \{(G_1, G_2) \mid G_1 \text{ and } G_2 \text{ are not isomorphic}\}$

Prover $P(x, m)$

```
P(x, m)
    (G1, G2) := x        # decode
    H := m               # π(Gi)

    # graph isomorphism은
    # exp-time에 계산가능
    if (H ≈ G1)
        return 1
    else
        return 2
```

Verifier $V(x, m)$

```
round := 0          # internal state variable
i := None           # internal state variable

V(x, m)
    (G1(V1, E1), G2(V2, E2)) := x
    if m ≠ ε                     # ≥ second round
        j := m                   # decode
        if (i ≠ j)
            return F             # 바로 reject
        elif (round = |V1|)
            return T             # accept
    i := random({1, 2})
    π := random({bijec. ∈ V1 → V2})
    round := round + 1
    return π(Gi)
```

from the viewpoint of interactive computation?

An Interactive Proof Systems for GNI (3/4)

From the Viewpoint of Interactive Computation

- Common input: undirected graphs $G_1 = ([n], E_1)$, $G_2 = ([n], E_2)$
- Repeat the following steps n times:
  1. Verifier chooses a random $i \in \{1, 2\}$ and a random permutation $\pi \in \mathcal{S}_n$, and sends $H = \pi(G_i)$ to prover
  2. Prover computes $j \in \{1, 2\}$ s.t. $G_j \approx H$, and sends j to verifier
  3. Verifier checks to see if $i = j$
- Verifier accepts prover's proof if $i = j$ in each of the n rounds.

Indeed an interactive proof system?

- completeness: $\langle P, V\rangle(G_1, G_2) = \mathtt{T}$ for all $G_1 \not\approx G_2$?
- soundness: $\forall P^*$, $\Pr\left[\langle P^*, V\rangle(G_1, G_2) = F\right] = 1 - \epsilon(|x|)$ for all $G_1 \approx G_2$?
- poly-time verifier: obvious
  - prover의 time-bound는 전혀 제한하지 않았음을 상기

An Interactive Proof Systems for GNI (4/4)

Completeness: $\forall G_1 \not\approx G_2$, $\langle P, V\rangle(G_1, G_2) = \mathtt{T}$

1. Exactly one of G_1, G_2 is isomorphic to $H = \pi(G_i)$, and the other not is not isomorphic to H
2. Prove can find G_j that is isomorphic to H (in exp-time), and send to verifier the right answer (s.t. $j = i$)

Soundness: $\forall P^*, \forall G_1 \approx G_2$, $\Pr\left[\langle P^*, V\rangle(G_1, G_2) = \mathtt{F}\right] = 1 - \epsilon(|x|)$

1. Let π^* be an isomorphism s.t. $\pi^*(G_1) = G_2$, and π be a random permutation selected by verifier
2. The probability distributions of π and $\pi \circ \pi^*$ are the same
3. The pdf of $\pi(G_1)$ and $\pi(G_2) = (\pi \circ \pi^*)(G_2)$ are the same
4. Thus, no prover can do better than make a guess $j = 1$ or 2, and so the probability of guessing all n choice $\leq 2^{-n}$

Poly-time verifier: obvious

An Interactive Proof Systems for GI (1/2)

The Graph Isomorphism (GI) Problem

$L_{GI} = \{(G_1, G_2) \mid G_1 \text{ and } G_2 \text{ are isomorphic}\}$

From the Viewpoint of Interactive Computation

- Common input: undirected graphs $G_1 = ([n], E_1)$, $G_2 = ([n], E_2)$
- Additional input to prover: isomorphism π^* s.t. $\pi^*(G_2) = G_1$
  - prover will convince verifier existence of π^* (w/o revealing π^*)
- Repeat the following steps n times:
  1. Prover chooses a random permutation $\pi \in \mathcal{S}_n$, and sends $H = \pi(G_1)$ to prover
  2. Verifier sends a random $i \in \{1, 2\}$ to prover
  3. Prover sends to verifier $\sigma = \begin{cases} \pi & \text{if } i = 1 \\ \pi \circ \pi^* & \text{if } i = 2 \end{cases}$
  4. Verifier checks if $\sigma(G_i) = H$
- Verifier accepts if $\sigma(G_i) = H$ in each of the n rounds.

An Interactive Proof Systems for GI (2/2)

Completeness: $\forall G_1 \approx G_2,\ \langle P, V\rangle(G_1, G_2) = \mathtt{T}$

1. If $i = 1$, then $\sigma(G_i) = \pi(G_1) = H$
2. If $i = 2$, then $\sigma(G_i) = \pi(\pi^*(G_2)) = \pi(G_1) = H$
    - 참고: π의 역할은 π^*를 uniform 확률분포속에 숨겨주기

Soundness: $\forall P^*, \forall G_1 \not\approx G_2,\ \Pr\left[\langle P^*, V\rangle(G_1, G_2) = \mathtt{F}\right] = 1 - \epsilon(|x|)$
(What happens if an (invalid) prover P^* tries to cheat verifier?)

1. No prover P^* can send H that is isomorphic to both G_1, G_2
2. The probability that verifier picks i s.t. $G_i \not\approx H$ is $\geq 2^{-1}$
3. If $G_i \not\approx H$, then there is no $\sigma \in \mathcal{S}_n$ s.t. $\sigma(G_i) = H$. Thus, P^* can cheat verifier with probability $\leq 2^{-1}$
4. The probability that P^* can cheat verifier n rounds $\leq 2^{-n}$

IPS for GNI와 달리 prover도 poly-time algorithm임에 주목

Zero-Knowledge?

Does the interactive proof systems (IPSs)for GNI/GI reveal any "knowledge" about the proofs beyond mere existence?

IPS for GNI:

- Verifier가 prover로부터 얻을 수 있는 "knowledge"는 이미 verifier 자신이 알고 있는 것이 전부
- Non-isomorphism의 proof에 대한 어떤 정보도 얻을 수 없음

IPS for GI:

- All that verifier sees is a random isomorphic copy H of G_1, G_2 and a permutation σ s.t. $\sigma(G_1) = H$ or $\sigma(G_2) = H$
- Verifier가 이 정보로부터 isomorphism π^*에 대한 조금의 "knowledge"를 얻을 수 있을까?

이제 zero-knowledge를 엄밀하게 정의해보자!

Outline

Perfect Zero-Knowledge Proof Systems

Informally, an interactive proof system (P, V) for a language L is said to be zero-knowledge if

- whatever can be obtained from P (in poly-time) on $x \in L$ can also be computed by V alone (in poly-time)

p.p.t = probabilistic polynomial-time

Definition (Perfect Zero-Knowledge)

Let (P, V) be an IPS for some language L. We say that (P, V), or actually P, is perfect zero-knowledge if

- $\forall$ p.p.t. ITM V^*, $\exists$ p.p.t. TM M^*, $\forall x \in L$, random variable $\langle P, V^* \rangle(x)$ and $M^*(x)$ are equally distributed

Such M^* is called a (perfect) simulator for (P, V^*)

주의: cheating verifier를 고려하려고 V 대신에 $\forall V^*$로 정의

Simulator for IPS for GI

The following sequence of date, called transcript, fully captures verifier's view of interactive computation:

- $T = ((G_1, G_2), (H_1, i_1, \sigma_1), (H_2, i_2, \sigma_2), \cdots, (H_n, i_n, \sigma_n))$

Any (fake) verifier V^* can simulate transcripts by itself!

```
T := (G_1, G_2)
for (j := 1 to n)
    Choose i_j ∈ {1,2} at random
    Choose σ_j ∈ S_n at random
    Compute H_j ∈ σ_j(G_{i_j}) at random
    T := append(T, (H_j, i_j, σ_j))
```

Informally speaking,

- 이런 방식으로 흉내낸 transcript의 마지막 계산 값을 $M^*(G_1, G_2)$로 simulator를 정의해버리면
- 랜덤변수 $\langle P, V\rangle(G_1, G_2)$와 $M^*(G_1, G_2)$의 확률분포는 동일!

Computational Zero-Knowledge Proof Systems

- PKZ의 정의는 지나치게 강하다고 볼 수도 있음
- 조건을 약화시키면 더욱 많은 language에 대해 더욱 효율적인 ZKP를 구성할 수 있음
- 랜덤변수 $\langle P, V^* \rangle(x)$와 $M^*(x)$의 확률분포가 똑같을 필요까지는 없고, computationally indistinguishable이면 충분

Definition (Computational Zero-Knowledge)

Let (P, V) be an IPS for some language L. We say that (P, V), or actually P, is computational zero-knowledge if

- $\forall$ p.p.t. ITM V^*, $\exists$ p.p.t. TM M^*,

  the following ensembles are computationally indistinguishable:

  $\{\langle P, V^* \rangle(x)\}_{x \in L}$ and $\{M^*(x)\}_{x \in L}$

Summary: Cheating Relations

Zero-knowledge proof의 정의의 어떤 요소에서 cheating이 prevent되나?

- Completeness: 딱히 cheater가 존재하지 않음
- Soundness: prover가 cheater가 될 수 있음. prover가 $x \notin L$의 proof가 존재한다고 속이는 것을 막기 위한 정의를 위해 all potential (fake) "prover" P^*로 정의
- Zero-knowledge: verifier가 cheater가 될 수 있음. verifier가 $x \in L$에 대한 prover의 (zero-knowledge) proof에서 정보를 crack하는 것을 막기 위한 정의를 위해 all ponential (fake) "verifier" V^*로 정의

Outline

Recall: Secure Coin flipping

Alice and Bob

- talk about going for dinner over telephone and
- want to decide who will pay for it.

The event must happen in the following order:

1. Alice flips a coin.
2. The coin lands. Alice notify Bob.
3. Bob informs Alice of his guess (head/tail).
4. Alice tells Bob whether the guess is right or not.

An analog protocol to prevent Alice from cheating Bob

- Step 2: Alice (1) take a photo of the coin landed, (2) put the photo in a safe, (3) locks the safe, (4) send the safe to Bob.
- Step 4: Alice send the key to unlock the safe.

Recall: Coin flipping using a public-key cryptosystem

An analog protocol to prevent Alice from cheating Bob

- Step 2: Alice (1) take a photo of the coin landed, (2) put the photo in a safe, (3) locks the safe, (4) send the safe to Bob.
- Step 4: Alice send the key to unlock the safe.

A digital protocol

- photo $\Rightarrow$ 0 or 1 (with random garbage padded)
- safe $\Rightarrow$ any one-way trapdoor function (encryption func)
- key of the safe $\Rightarrow$ trapdoor info (secret key)

1. Alice randomly chooses 0 or 1.
   - coin flipping not needed!
2. Alice sends its encryption (and the encryption function.)
3. Bob informs Alice of his guess (0 or 1).
4. Alice sends Bob the secret key to invert the encryption func.

Commitment Schemes: Informal Definition

Very informally, a commitment scheme is a two-phase interactive protocol bet'n two parties, a sender and a receiver, in which:

The first phase (commit phase):

1. Sender picks a random key k
2. Sender computes an encryption $y = e_k(m)$ of a msg m
3. Sender sends y (a "commitment" to m) to receiver

The second phase (reveal phase):

1. Sender sends the key k (along with $d_k(\cdot)$) to receiver
2. Receiver opens the "commitment" y to find out m

A commitment scheme must satisfy two security requirements:

- Hiding: no p.p.t. receiver can cheat (no info. about m from y)
- Binding: no p.p.t sender can cheat (no k' s.t. $e_{k'}(m') = e_k(m)$)

Commitment Schemes: Informal Definition

A commitment scheme must satisfy two security requirements:

- Hiding: no p.p.t. receiver can cheat
  - no information about m can be computed from y in poly-time
- Binding: no p.p.t sender can cheat
  - no k', m' s.t. $e_{k'}(m') = e_k(m)$ can be computed in poly-time
  - i.e.

p.p.t. sender/receiver로 제한한 이유: exp-time을 허용하면 두 조건 모두 깨지므로

- Public-key encryption function의 존재성(현재로서는 $P \neq NP$ 보다 강한 명제)을 가정한다면 위 조건들을 모두 만족하는 commitment scheme을 쉽게 만들수 있음
- $m \in \{0, 1\}$인 경우만 고려해도 충분 (bit를 이어붙이면 됨)

Commitment Schemes

Definition (Commitment Scheme (somewhat simplified))

A p.p.t. TM C is called a commitment scheme if there exists some polynomial $p(\cdot)$ s.t.

- hiding: $\forall n \in \mathbb{N},\ \forall v_0, v_1 \in \{0,1\}^n$,

  the following ensembles are computationally indistinguishable:

  $\left\{C(v_0, r)\right\}_{r \in \{0,1\}^{p(n)}}$ and $\left\{C(v_1, r)\right\}_{r \in \{0,1\}^{p(n)}}$
- binding: $\forall n \in \mathbb{N},\ \forall v_0, v_1 \in \{0,1\}^n,\ \forall r_0, r_1 \in \{0,1\}^{p(n)}$,

  $C(v_0, r_0) \neq C(v_1, r_1)$

Theorem

If one-way permutations exist, then commitment schemes exist

Outline

Zero-Knowledge Proof System for 3-Colorability (1/2)

The Graph 3-Colorability (G3C) Problem $\in \mathcal{NPH}$

$$L_{G3C} = \{G=(V,E) \mid \exists\, \phi : V \to [3],\ \forall (u,v) \in E,\ \phi(u) \neq \phi(v)\}$$

From the Viewpoint of Interactive Computation

- Common input: undirected graph $G=(V,E)$
- Additional input to prover: valid 3-coloring $\phi : V \to [3]$ of G
  - prover will convince verifier existence of ϕ (w/o revealing ϕ)
- Repeat the following steps $|V||E|$ times:
  1. Prover chooses a random permutation $\pi \in \mathcal{S}_3$, and sends $\left(C_v(\pi(\phi(v)))\right)_{v \in V}$ to sender (each C_v is a commitment)
  2. Verifier sends a random $(u,v) \in E$ to prover
  3. Prover sends keys to open commitments $C_u(\cdot)$ and $C_v(\cdot)$
  4. Verifier opens commitments $a_u = \pi(\phi(u))$, $a_v = \pi(\phi(v))$
  5. Verifier checks if $a_u \neq a_v$
- Verifier accepts if $a_u \neq a_v$ in each of $|V||E|$ rounds

Zero-Knowledge Proof System for 3-Colorability (2/2)

Completeness: $\forall G \in L_{G3C}$, $\langle P, V\rangle(G) = \mathtt{T}$

1. For any $(u, v) \in E$, $\pi(\phi(u)) \neq \pi(\phi(v))$ (since $\phi(u) \neq \phi(v)$)

Soundness: $\forall P^*$, $\forall G \notin L_{G3C}$, $\Pr\left[\langle P^*, V\rangle(G) = \mathtt{F}\right] = 1 - \epsilon(|x|)$

1. For each $\phi^* : V \to [3]$, there is $(u, v) \in E$ s.t. $\phi^*(u) = \phi^*(v)$
2. By the binding property of the commitment scheme, a cheating prover is caught with probability $\geq 1/|E|$
3. The probability that a cheating prover successfully cheats in all $|V||E|$ rounds is $\leq (1 - 1/|E|)^{|V||E|} \leq e^{-|V|}$

(Computational) Zero-knowledge: (see next slide for more..)

- The hiding property of the commitment scheme guarantees that, in each iteration, everything except 2 random colors is hidden

Simulator for ZKP for 3-Colorability

Any (fake) verifier V^* can simulate transcripts by itself!

1. Choose $(u', v') \in E$ at random
2. Choose $a'_u, a'_v \in [3]$ s.t. $a'_u \neq a'_v$ at random
3. Let $a'_w = 1$ for all $w \in V \setminus \{u', v'\}$
4. Commit to a'_u for each $u \in V$ and feed the commitments to V^* (just as honest prover)
   - while also providing it truly random bits as its random coins
5. Let (u, v) denote the answer from V^*
6. If $(u, v) = (u', v')$, then reveal the two colors, and output the view of V^*
7. Otherwise, restart the process from the first step, but at most $|V||E|$ times.
8. If, after $|V||E|$ repetitions the simulation has not been successful, output $\mathtt{F}$

(Computational) ZKP for $\mathcal{NP}$

Theorem

If one-way permutations exist, then every $L \in \mathcal{NP}$ has a (computational) ZKP.

1. Fix a language $L \in \mathcal{NP}$. Note that $L_{G3C} \in \mathcal{NPC}$
2. By Cook-Levin theorem, there is a deterministic poly-time algorithm (i.e. reduction) $R : \{0,1\}^* \to \{0,1\}^*$ s.t.
   $$x \in L \iff R(x) \in L_{G3C}$$
3. Furthermore, for each $x \in L$ and its certificate z, reduction R also implicitly computes the certificate z' of $R(x) \in L_{G3C}$
   - certificate도 계산하도록 augment된 reduction을 R^c로 두자
4. L의 ZKP를 L_{G3C}의 ZKP를 이용하여 구성하면 된다:

$z :=$ `certificate of` $X \in L$
$P_L(x, m)$
  $(G, z') := R^w(x, z)$ `# reduction`
  $P_{L_{G3C}}(G, m, z')$

$V(x, m)$
  $G := R(x)$ `# reduction`
  $V_{L_{G3C}}(G, m)$

Remark

- 임의의 $L \in \mathcal{NP}$에 대한 ZKP를 구성하기 위해 하필 $\mathrm{G3C}$를 이용한 이유는 이것이 NP-complete이기 때문
  - GI처럼 NP-easy만 증명된 language의 경우 $L \in \mathcal{NP}$에서 L_{GI}로의 poly-time reduction이 존재함이 보장되지 않음
- $L \in \mathcal{NP}$에 대한 "practical" ZKP를 구성하려고 할 경우에는 G3C로 reduction시켜서 만든 generic ZKP를 사용하면 곤란
  - poly-time reduction $L \mapsto \mathrm{SAT} \mapsto \mathrm{G3C}$ 과정에서 instance/certificate의 크기가 매우 커지므로
- $\mathrm{G3C}$에 대한 ZKP는 perfect는 아니고 computational ZK만 보장되므로 $L \in \mathcal{NP}$에 대한 computational ZKP의 존재성까지만 보장됨

Complexity Issues

$$\mathcal{BPP} \subseteq \mathcal{PZK} \subseteq \mathcal{CZK} \subseteq \mathcal{IP} = \mathcal{PSPACE}$$

- $\mathcal{PZK}$: the set of languages with perfect ZKP
- $\mathcal{CZK}$: the set of languages with computational ZKP
- If one-way functions exists, then $\mathcal{CZK} = \mathcal{IP}$ $(= \mathcal{PSPACE})$
- It is widely believed that $\mathcal{BPP} \subsetneq \mathcal{PZK} \subsetneq \mathcal{CZK}$

Poly-time prover

- Theorem: Every $L \in \mathcal{NP}$ has a (computational) ZKP where prover can be implemented in poly-time given a certificate
- G3C, GI가 (certificate가 주어진 상황에서) poly-time prover를 가짐을 상기
  - $\mathcal{NP}$에 속하는지 여부가 밝혀지지 않은 GNI는 exp-time prover를 소개했었음
- Prover가 poly-time에 작동하는 것은 identification scheme, multi-party secure computation 등의 application에서 필요

Outline

Zero-Knowledge Proofs of Knowledge

- Let $L \in \mathcal{NP}$, and C_L be a poly-time certifier for L.
- z is a proof of (the proposition) "$x \in L$" if $C_L(x, z) = \mathtt{T}$
- Zero-knowledge proof of (the proposition) "$x \in L$": proof of the "mere" existence of such z (w/o revealing any info about z)
  - prover는 z의 존재성만 보이면 되므로 z의 실체는 몰라도 됨
- Zero-knowledge proof of knowledge of "$x \in L$": proof of actually knowing such z (w/o revealing any info about z)
  - prover는 z의 존재성을 넘어서서 z의 실체를 알아야 함
  - c.f. non-constructive proof vs. constructive proof
- Zero-knowledge proof of knowledge는 ZKP보다 강한 정의로 어떤 application에서는 이것이 필요한 경우가 있다

Proofs of Knowledge

For simplicity, we cosider only $L \in \mathcal{NP}$ (with poly-time certifier C_L)

Definition (Proof of Knowledge)

A ZKP (P, V) for L is called a proof of knowledge with knowledge error bound ϵ and extractor slowdown *es* if

- there is TM K (called knowledge extractor) s.t. $\forall P^*$, $\forall x \in L$,

$$\Pr\left[\langle P^*, V\rangle(x) = \text{"T"}\right] \geq \epsilon + \delta \implies$$

$$K(P^*, x) \text{ computes } z \text{ s.t. } C_L(x, z) = \mathrm{T}$$

$$\text{in average time of} \leq es \cdot |x|^{O(1)} \cdot \delta^{-1}$$

- ZKP for GI is a knowledge of proof with knowledge error $1/2$
- ZKP for G3C is a knowledge of proof with k. error $1 - 1/|E|$
- 여러번 돌려서 knowledge error를 exponentially 줄일 수 있음

Theorem

Any $L \in \mathcal{NP}$ has a ZKP for proofs of knowledge (assuming OWF..)

Outline

Identification Scheme

- 사용자(prover)가 서버(verifier)로의 접속을 위해 신원을 확인(identification)시키는 절차
  - 사용자의 passwd를 암호화해서 보내면 될 것 같은데..
- RSA와 같은 public-key encryption scheme에 기반한 digital signature를 사용하면 어떤 방식의 cracking이 가능?
  - 암호화된 passwd를 그대로 Eve가 가로채서 사용자 행세..
  - 또는, 서버에 passwd 관련된 정보가 남은 상태에서, Eve가 서버를 턴다면..
  - 알고보니 서버 관리자가 Eve라면..
- 사용자가 $L \in \mathcal{NPC}$와 certificate(proof) z를 구성할 수 있는 $x \in L$를 선택하여 z를 암호로 삼고, 서버에 (L, x)를 넘겨주고 서버와의 ZKP of proof of knowledge를 돌리면 서버에는 암호 z에 대한 어떤 정보도 남지 않음!
  - 임의의 x에 대한 z는 구성하기 힘들지만, (x, z)를 한번에 구성할 수 있는 방법은 많음

Recall: Secure Multi-Party Computation for Dating for Shy People

- encryption: $f(x) = x^e \bmod n$ / dec.: $f^{-1}(y) = y^d \bmod n$

Dating protocol based on RSA (for avoiding Alice's shame)

1. Alice creates e, d, n and publicize the public key (i.e. $f(x)$).
2. Alice sends Bob $\big(f(x),\ f(y)\big)$ where x, y are:
   - $x =$ "0"+random, $y =$ "0"+random if not interested in Bob
   - $x =$ "0"+random, $y =$ "1"+random if interested in Bob
   - Both $f(x)$ and $f(y)$ look completely random to Bob.
3. Bob picks a random $r \in \mathbb{Z}_n$. Then, he sends to Alice z:
   - $z = f(x) \cdot f(r) \bmod n = f(xr)$ if not interested in Alice
   - $z = f(y) \cdot f(r) \bmod n = f(yr)$ if interested in Alice
4. Alice computes/sends $f^{-1}(z) = xr$ or $yr \bmod n$ back to Bob.
   - Either way, $f^{-1}(z)$ looks completely random (by r) to Alice
5. Bob computes $w = f^{-1}(z) \cdot r^{-1} \bmod n = x$ or y.
   - Bob not interested: $w = x$ (Alice's interest not revealed)
   - Bob interested: $w = y$ (Alice's interest revealed)

Multi-Party Secure Computation with ZKP

- Alice/Bob이 앞의 protocol을 충실히 따른다고 가정할 때 원하는 함수를 정확히, 정보유출없이 계산할 수 있음
  - $f : \{\text{"interested"}, \text{"uninterested"}\}^2 \rightarrow \{\text{"date"}, \text{"rupture"}\}$;
    $f(x_1, x_2) = \text{"date"} \quad \text{iff} \quad x_1 = x_2 = \text{"interested"}$

- 이런 상황을 "honest but curious" player만 참가한다고 하는데, "malicious" player가 참가할 경우에는 위 함수가 제대로 계산되지 않음 (즉, cheating이 가능할 수도 있음)

- Malicious cheating을 막기 위해서는 앞 페이지의 protocol의 각 step마다 다음 명제에 대한 ZKP를 함께 보내면 됨:
  "내가 보내는 msg가 protocol상에 정의된 것과 같음"
  - 적절한 $L \in \mathcal{NP}$가 존재하여 위 명제를 $x \in L$로 표현가능
  - $L \in \mathcal{NP}$에 대한 (computational) ZKP는 항상 존재!