Introduction to Modern Cryptography
Recitation 10

Orit Moskovich
Tel Aviv University
January 4, 2017

Interactive Proof System (IP)

Definition. An **Interactive Proof System** for a language L is a two-party game between a PPT verifier V and a prover P that interact on a common input x in a way satisfying the following properties:

- **Completeness**: For all $x \in L$, the prover P convinces the verifier with probability at least $1 - \varepsilon$

$$\Pr_V[(P, V)(x) = 1] \geq 1 - \varepsilon$$

- **Soundness**: For all $x \notin L$, any prover strategy P^* convinces the verifier with probability at most ε

$$\Pr_V[(P^*, V)(x) = 1] \leq \varepsilon$$

Interactive Proof System (IP)

- Any $L \in NP$ has an IP system
  - Why?
  - Let R_L be the witness relation for L
  - $L = \{x \mid \exists y.\, (x, y) \in R_L\}$ and R_L is recognizable in poly time

- The protocol on common input x, and P's private witness y
- $P \rightarrow V$: send y to V
- V verifies using R_L

  - Completeness?
  - Soundness?
  - Zero knowledge?

- GNI

Zero Knowledge (ZK)

- What does V learn from the proof?
- What if V cheats?
- ZK:
  - The prover P convinces the verifier V that a statement is true,
  - Yet, V will not learn anything as a result of this process

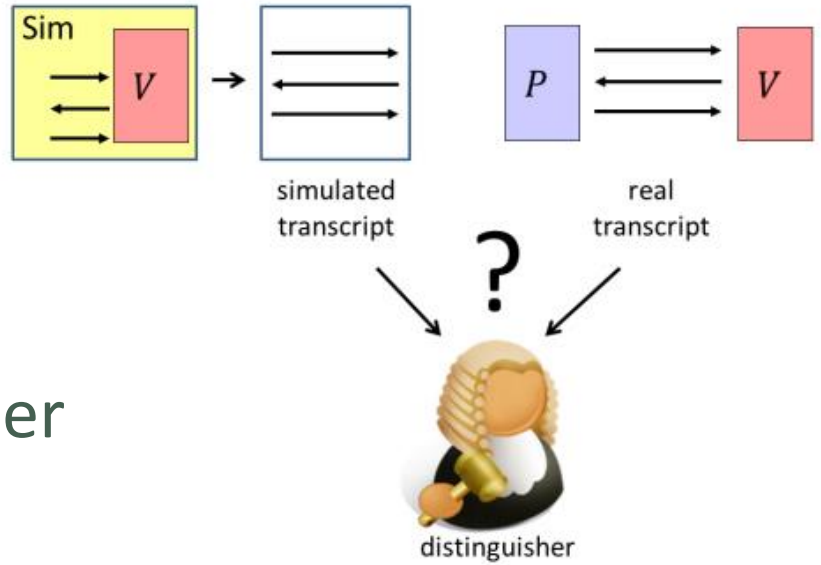
- We'll say that an IP sysyem (P, V) is ZK if whatever can be efficiently computed after interacting with P on input x
  can also be efficiently computed from x (without any interaction)

Zero Knowledge (ZK)

Definition. An Interactive Proof System for a language L is (computational/statistical) **zero knowledge (ZK/CZK/SZK)** if for any PPT verifier V^*, there exist an *expected* PPT simulator S such that for every $x \in L$:

$$S(x) \approx_{c/s,\varepsilon} View_{V^*}(P, V^*)(x)$$

- The view of V^* on x:
  - All messages sent from P to V^*
  - All random bits used by V^* during the execution of the protocol on x
- Can consider only the case of an honest verifier



3-Col

A protocol for proving 3-Col $(P(\varphi), V)(G)$:
Common input: $G = (V, E)$
Private input of P: $\varphi: V \to [3]$ such that $\forall (u, v) \in V . \varphi(u) \neq \varphi(v)$

1. $P \to V$: Chooses a random permutation $\sigma: [3] \to [3]$ and sends to V a commitment $c_v = COM\left(\sigma\left(\varphi(v)\right)\right)$ for every $v \in V$
   Let $\varphi'(v) = \sigma\left(\varphi(v)\right)$
2. $V \to P$: Picks an edge (u, v) and send u, v to P
3. $P \to V$: Reveals $\varphi'(u)$ and $\varphi'(v)$
4. V accepts iff $\varphi'(u) \neq \varphi'(v)$

3-Col

1. $P \to V$: Chooses a random permutation σ: $[3] \to [3]$ and sends to V a commitment $c_v = COM\left(\sigma\left(\varphi(v)\right)\right)$ for every $v \in V$
   Let $\varphi'(v) = \sigma\left(\varphi(v)\right)$
2. $V \to P$: Picks an edge (u, v) and send u, v to P
3. $P \to V$: Reveals $\varphi'(u)$ and $\varphi'(v)$
4. V accepts iff $\varphi'(u) \neq \varphi'(v)$

- Completeness:
  - Perfect
  - If φ is a valid 3-coloring then $\varphi'(u) \neq \varphi'(v)$ for any edge (u, v)
  - V accepts

3-Col

1. $P \to V$: Chooses a random permutation σ: $[3] \to [3]$ and sends to V a commitment $c_v = COM\left(\sigma\left(\varphi(v)\right)\right)$ for every $v \in V$
   Let $\varphi'(v) = \sigma\left(\varphi(v)\right)$
2. $V \to P$: Picks an edge (u, v) and send u, v to P
3. $P \to V$: Reveals $\varphi'(u)$ and $\varphi'(v)$
4. V accepts iff $\varphi'(u) \neq \varphi'(v)$

- Soundness:
  - If $G \notin 3 - Col$, for every coloring φ' for which P^* commits, there exists an edge (u, v) such that $\varphi'(u) = \varphi'(v)$.
  - Any prover P^* fails to convince V with probability at least $\frac{1}{|E|}$
  - V accepts the false statement w.p $\leq 1 - \frac{1}{|E|}$

3-Col

1. $P \to V$: Chooses a random permutation σ: $[3] \to [3]$ and sends to V a commitment $c_v = COM\left(\sigma\left(\varphi(v)\right)\right)$ for every $v \in V$
   Let $\varphi'(v) = \sigma\left(\varphi(v)\right)$
2. $V \to P$: Picks an edge (u, v) and send u, v to P
3. $P \to V$: Reveals $\varphi'(u)$ and $\varphi'(v)$
4. V accepts iff $\varphi'(u) \neq \varphi'(v)$

- CZK for honest verifier, a PPT simulator $S(G)$:

1. Choose a random edge $(\tilde{u}, \tilde{v}) \in E$
2. Choose randomly two colors $\varphi'(\tilde{u}) \neq \varphi'(\tilde{v})$ and complete the rest of the coloring randomly
3. $M_1(\mathrm{P} \to V)$: A commitment $c_v = COM(\varphi'(v))$ for every $v \in V$
4. $M_2(V \to P)$: $(\tilde{u}, \tilde{v})$
5. $M_3(P \to V)$: Reveal $\varphi'(\tilde{u})$ and $\varphi'(\tilde{v})$

3-Col

1. $P \to V$: Chooses a random permutation σ: $[3] \to [3]$ and sends to V a commitment $c_v = COM\left(\sigma\left(\varphi(v)\right)\right)$ for every $v \in V$
   Let $\varphi'(v) = \sigma\left(\varphi(v)\right)$
2. $V \to P$: Picks an edge (u, v) and send u, v to P
3. $P \to V$: Reveals $\varphi'(u)$ and $\varphi'(v)$
4. V accepts iff $\varphi'(u) \neq \varphi'(v)$

?

- CZK for honest verifier, a PPT simulator $S(G)$:

1. Choose a random edge $(\tilde{u}, \tilde{v}) \in E$
2. Choose randomly two colors $\varphi'(\tilde{u}) \neq \varphi'(\tilde{v})$ and complete the rest of the coloring randomly
3. $M_1(\mathrm{P} \to V)$: A commitment $c_v = COM(\varphi'(v))$ for every $v \in V$
4. $M_2(V \to P)$: $(\tilde{u}, \tilde{v})$
5. $M_3(P \to V)$: Reveal $\varphi'(\tilde{u})$ and $\varphi'(\tilde{v})$

3-Col

1. $P \to V$: Chooses a random permutation σ: $[3] \to [3]$ and sends to V a commitment $c_v = COM\left(\sigma\left(\varphi(v)\right)\right)$ for every $v \in V$
   Let $\varphi'(v) = \sigma\left(\varphi(v)\right)$
2. $V \to P$: Picks an edge (u, v) and send u, v to P
3. $P \to V$: Reveals $\varphi'(u)$ and $\varphi'(v)$
4. V accepts iff $\varphi'(u) \neq \varphi'(v)$

malicious

- CZK for ~~honest~~ verifier, an expected PPT simulator $S(G)$:

1. Choose a random edge $(\tilde{u}, \tilde{v}) \in E$
2. Choose randomly two colors $\varphi'(\tilde{u}) \neq \varphi'(\tilde{v})$ and complete the rest of the coloring randomly
3. $M_1(\mathrm{P} \to V)$: A commitment $c_v = COM(\varphi'(v))$ for every $v \in V$
4. $M_2(V \to P)$: Query V^* for an edge (u^*, v^*)
5. $M_3(P \to V)$: If $(\tilde{u}, \tilde{v}) = (u^*, v^*)$, reveal $\varphi'(u)$ and $\varphi'(v)$, otherwise, rewind

3-Col

1. $P \to V$: Chooses a random permutation σ: $[3] \to [3]$ and sends to V a commitment $c_v = COM\left(\sigma\left(\varphi(v)\right)\right)$ for every $v \in V$
   Let $\varphi'(v) = \sigma\left(\varphi(v)\right)$
2. $V \to P$: Picks an edge (u, v) and send u, v to P
3. $P \to V$: Reveals $\varphi'(u)$ and $\varphi'(v)$
4. V accepts iff $\varphi'(u) \neq \varphi'(v)$

malicious

- CZK for ~~honest~~ verifier, an ? expected PPT simulator $S(G)$:

1. Choose a random edge $(\tilde{u}, \tilde{v}) \in E$
2. Choose randomly two colors $\varphi'(\tilde{u}) \neq \varphi'(\tilde{v})$ and complete the rest of the coloring randomly
3. $M_1(\mathrm{P} \to V)$: A commitment $c_v = COM(\varphi'(v))$ for every $v \in V$
4. $M_2(V \to P)$: Query V^* for an edge (u^*, v^*)
5. $M_3(P \to V)$: If $(\tilde{u}, \tilde{v}) = (u^*, v^*)$, reveal $\varphi'(u)$ and $\varphi'(v)$, otherwise, rewind

Hamilton Cycle

A protocol for proving G has a Hamiltonian cycle $(P(c), V)(G)$:

Common input: $G = (V, E), V = [1, \dots, n]$

Private input of P: A Hamiltonian cycle c in G

1. $P \to V$: Chooses a random permutation on the vertices $\sigma: [n] \to [n]$ and computes $G_\sigma = \sigma(G)$ and send to V a commitment $COM(\sigma)$ and a commitment of the adjacency matrix G_σ
2. $V \to P$: Chooses a random $b \in \{0,1\}$ and sends it to P
3. $P \to V$: If $b = 0$ ➔ Reveals σ and G_σ to V
   If $b = 1$ ➔ P uses $\sigma(c)$ to show V a Hamiltonian cycle in G_σ
4. V verifies the last message according to b:
   If $b = 0$ ➔ V checks that the information sent by P matches the commitment, and verifies $G_\sigma = \sigma(G)$
   If $b = 1$ ➔ V checks if the cycle P sent is actually a Hamiltonian cycle

Hamilton Cycle

- Completeness:
  - Perfect
  - If c is a valid Hamiltonian cycle then there exists a Hamiltonian cycle in $\sigma(G)$
  - V accepts

- Soundness:
  - If $G \notin HamCycle$, for every permutation σ for which P^* commits, there is no Hamiltonian cycle ➔ Any prover P^* fails to convince V with probability $\frac{1}{2}$ (when V asks $b = 1$)
  - If P^* cheats in the first commitment and commits to a random graph with Hamiltonian cycle, he will be caught w.p $\frac{1}{2}$ (when V asks for the isomorphism – $b = 0$)

Hamilton Cycle

- CZK for honest verifier, a PPT simulator $S(G)$:

1. Choose a random bit $\tilde{b} \in \{0,1\}$ and a random permutation on the vertices $\sigma: [n] \to [n]$
2. If $\tilde{b} = 0$, commit to G_σ
   If $\tilde{b} = 1$, commit to a random graph that contains a HamCycle (under the permutation σ)
3. $M_1(\mathrm{P} \to V)$: The commitment chosen in step 2
4. $M_2(V \to P)$: $\tilde{b}$
5. $M_3(P \to V)$: Reveal the commitment ($\tilde{b} = 0$) or the cycle ($\tilde{b} = 1$)

Hamilton Cycle

?

- CZK for honest verifier, a PPT simulator $S(G)$:

1. Choose a random bit $\tilde{b} \in \{0,1\}$ and a random permutation on the vertices $\sigma: [n] \to [n]$
2. If $\tilde{b} = 0$, commit to G_σ
   If $\tilde{b} = 1$, commit to a random graph that contains a HamCycle (under the permutation σ)
3. $M_1(\mathrm{P} \to V)$: The commitment chosen in step 2
4. $M_2(V \to P)$: $\tilde{b}$
5. $M_3(P \to V)$: Reveal the commitment ($\tilde{b} = 0$) or the cycle ($\tilde{b} = 1$)

Hamilton Cycle

malicious

- CZK for ~~honest~~ verifier, an expected PPT simulator $S(G)$:

1. Choose a random bit $\tilde{b} \in \{0,1\}$ and a random permutation on the vertices $\sigma: [n] \to [n]$
2. If $\tilde{b} = 0$, commit to G_σ
   If $\tilde{b} = 1$, commit to a random graph that contains a HamCycle (under the permutation σ)
3. $M_1(\mathrm{P} \to V)$: The commitment chosen in step 2
4. $M_2(V \to P)$: Query V^* for a bit b^*
5. $M_3(P \to V)$: If $\tilde{b} = b^*$, Reveal the commitment ($\tilde{b} = 0$) or the cycle ($\tilde{b} = 1$) otherwise, rewind

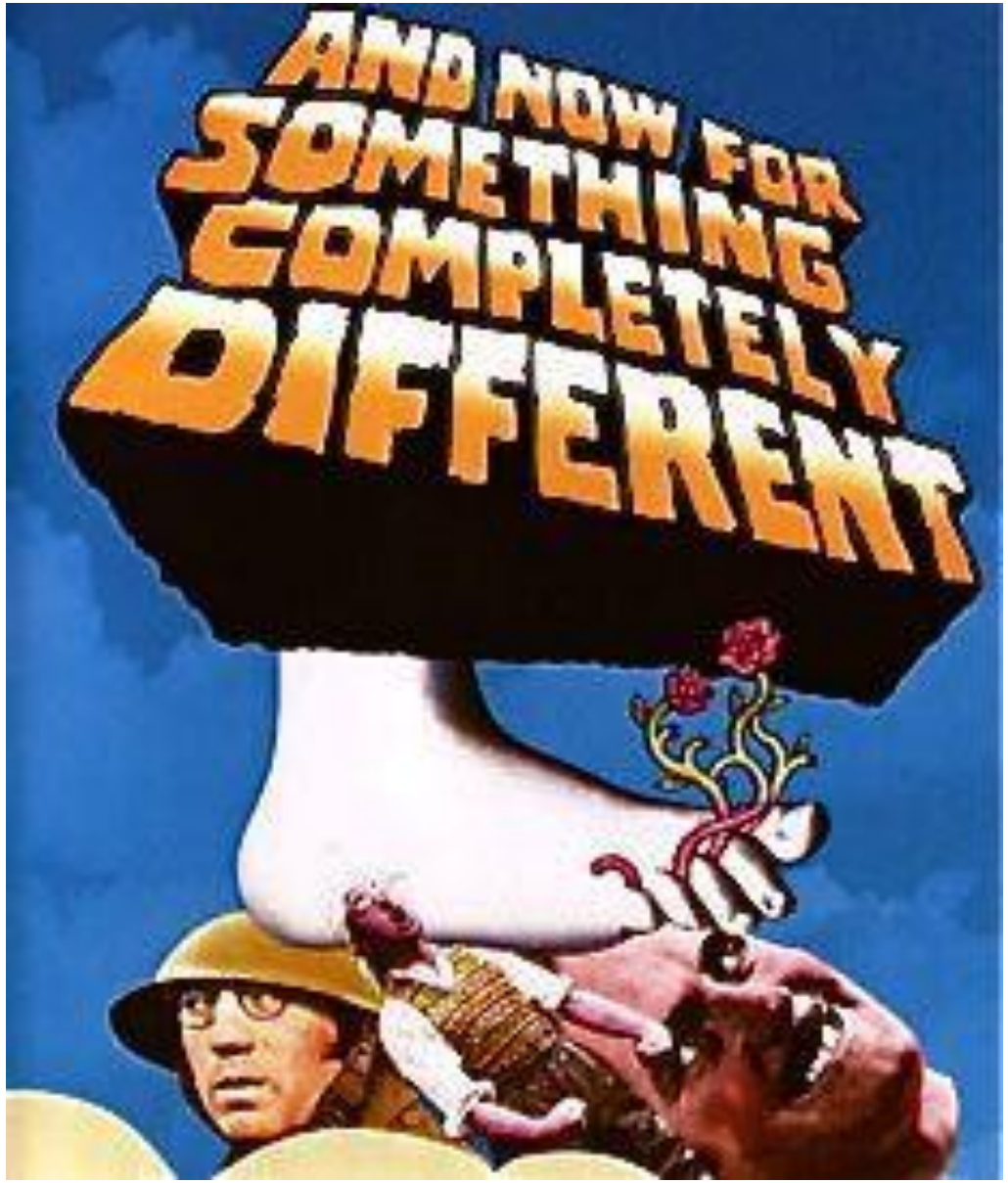
Hamilton Cycle

malicious

- CZK for ~~honest~~ verifier, an expected PPT simulator $S(G)$:

?

1. Choose a random bit $\tilde{b} \in \{0,1\}$ and a random permutation on the vertices $\sigma: [n] \to [n]$
2. If $\tilde{b} = 0$, commit to G_σ
   If $\tilde{b} = 1$, commit to a random graph that contains a HamCycle (under the permutation σ)
3. $M_1(\mathrm{P} \to V)$: The commitment chosen in step 2
4. $M_2(V \to P)$: Query V^* for a bit b^*
5. $M_3(P \to V)$: If $\tilde{b} = b^*$, Reveal the commitment ($\tilde{b} = 0$) or the cycle ($\tilde{b} = 1$) otherwise, rewind


AND NOW FOR
SOMETHING
COMPLETELY
DIFFERENT

Partially Homomorphic Encryption

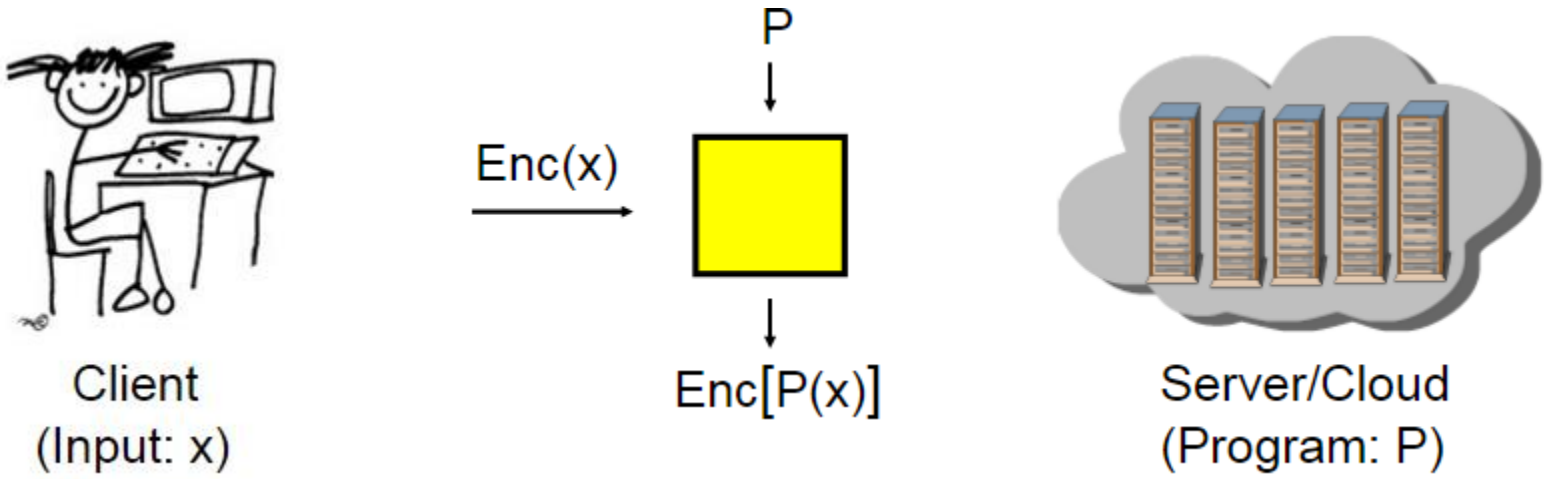
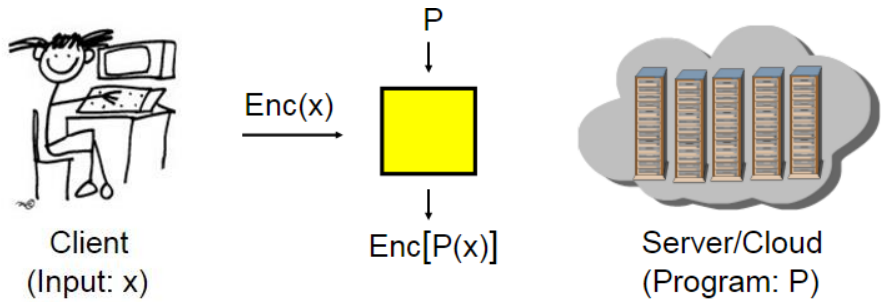


Image from: http://slideplayer.com/slide/236532/

Partially Homomorphic Encryption



Definition. A PKE scheme (Gen, Enc, Dec) is (partially) **homomorphic** if for all pk, sk and for all m_1, c_1, m_2, c_2:

$m_1 = Dec_{sk}(c_1)$ and $m_2 = Dec_{sk}(c_2)$ ➔

$$Dec_{sk}(c_1 \tilde{\odot} c_2) = m_1 \odot m_2$$

El Gamal Encryption Scheme- The Short Version

El Gamal PKE Scheme. Let G be a cyclic group of order q and a generator $g \in G$.

1. Key generation algorithm Gen chooses a random $x \in \mathbb{Z}_q$.
   $pk = (G, q, g, g^x), sk = (G, q, g, x)$
2. To encrypt a message $m \in G$ using $pk = (G, q, g, h)$:
   - Choose a random $y \leftarrow \mathbb{Z}_q$
   - Compute g^y
   - Send $Enc_{pk}(m) = (g^y, h^y \cdot m)$
3. To decrypt a message (c_1, c_2) using $sk = (G, q, g, x)$:
   - Compute $Dec_{sk}\big((c_1, c_2)\big) = c_2 \cdot (c_1^x)^{-1}$

Partially Homomorphic Encryption

- El Gamal PKE scheme:
- $pk = (G, q, g, g^x = h)$
- $sk = x$

- $Enc_{pk}(m_1) = (g^y, h^y \cdot m_1) = c_1$
- $Enc_{pk}(m_2) = \left(g^{y'}, h^{y'} \cdot m_2\right) = c_2$

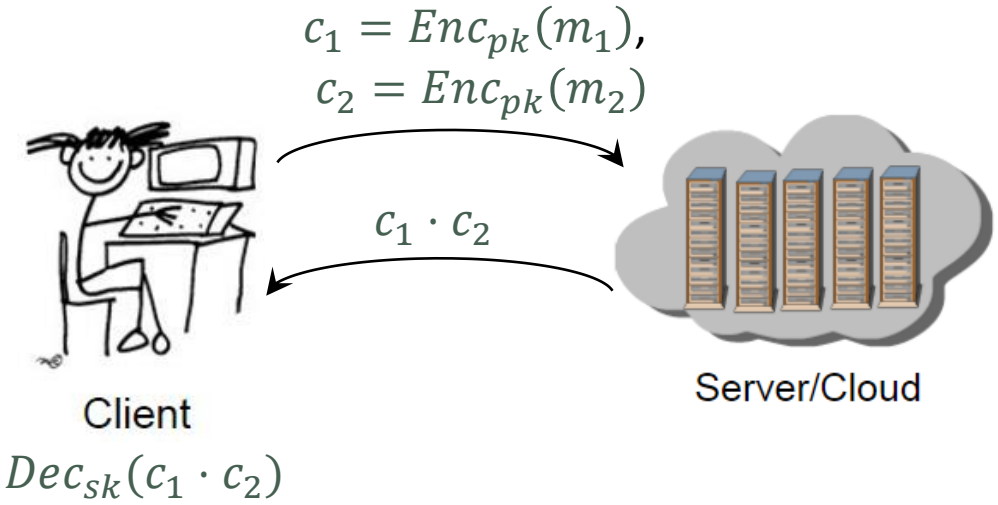
- ➔ $c_1 \cdot c_2 = \left(g^{y+y'}, h^{y+y'} \cdot (m_1 m_2)\right)$
- ➔ $Dec_{sk}(c_1 \cdot c_2) = m_1 m_2$

Partially Homomorphic Encryption

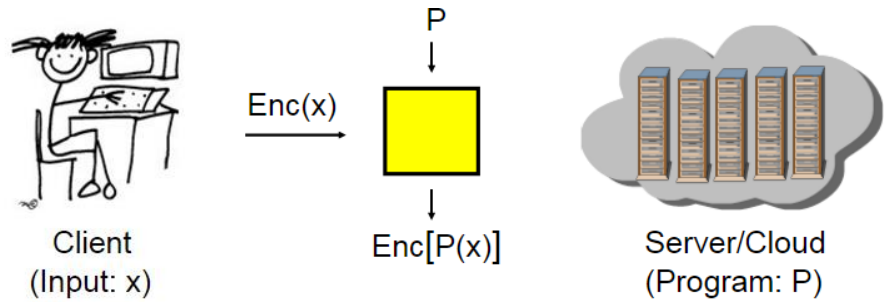
- El Gamal PKE scheme:
- $pk = (G, q, g, g^x = h)$
- $sk = x$

- $Enc_{pk}(m_1) = (g^y, h^y \cdot m_1) = c_1$
- $Enc_{pk}(m_2) = (g^{y'}, h^{y'} \cdot m_2) = c_2$

- ➔ $c_1 \cdot c_2 = \left(g^{y+y'}, h^{y+y'} \cdot (m_1 m_2)\right)$
- ➔ $Dec_{sk}(c_1 \cdot c_2) = m_1 m_2$



Fully Homomorphic Encryption



Definition. An encryption scheme $(Gen, Enc, Dec, \boldsymbol{Eval})$ is (fully) **homomorphic (FHE)** if:

- Gen, Enc, Dec as usual (assume private key for now)
- $Eval$: A PPT algorithm that given $F: \{0,1\}^n \to \{0,1\}^{n'}$ and an encryption $c = Enc_{sk}(m)$, F is homomorphically computed

$$\tilde{c} = Eval(F, c) \sim Enc_{sk}\big(F(m)\big)$$

Fully Homomorphic Encryption

- Properties:
  - Semantic security $Enc_{sk}(m_0) \approx_{c,\varepsilon} Enc_{sk}(m_1)$
  - $Eval$ has to be probabilistic

- We'll assume:
  - F is efficiently computable
  - $\tilde{c}$ is taken from $\{Enc_{sk}(F(m))\}$
    (the result of $Eval$ is indistinguishable from regular encryption)
  - $|\tilde{c}|$ depends on $|F(m)|$ (not on the running time of F)
  - Decryption time is not dependent on the computation time of F

ZK from FHE

- $L \in NP$ if
- There exists a function $F_L(x, w)$ such that
- Completeness: $\forall x \in L.\ \exists w.\ F_L(x, w) = 1$
- Soundness: $\forall x \notin L.\ \forall w.\ F_L(x, w) \neq 1$
- F_L is poly time in $|x|$

ZK for any $L \in NP$

- Let $L \in NP$
- **First attempt:**

ZK for any $L \in NP$

A protocol for proving $x \in L$ $(P(w), V)(x)$:

Common input: x

Private input of P: A witness w such that $F_L(x, w) = 1$

1. $P \to V$: Send to V an encryption of the witness $c = Enc_{sk}(w)$
2. $V \to P$: Compute $\hat{c} = Eval(F_L(x,\cdot), c) \sim Enc_{sk}\left(F_L(x, w)\right)$ and send it to P
3. $P \to V$: Decrypt $b = Dec_{sk}(\hat{c})$ and send the result to V
4. V accepts iff $b = 1$

ZK for any $L \in NP$

- Let $L \in NP$
- **First attempt:**
  - Not sound – a cheating P^* can simply send $b = 1$ on step 3

ZK for any $L \in NP$

- Let $L \in NP$
- **Second attempt:**

ZK for any $L \in NP$

A protocol for proving $x \in L$ $(P(w), V)(x)$:

Common input: x

Private input of P: A witness w such that $F_L(x, w) = 1$

1. $P \to V$: Send to V an encryption of the witness $c = Enc_{sk}(w)$
2. $V \to P$: Flip a coin $b \in \{0,1\}$

   If $b' = 0$ ➔ Compute $\hat{c} = Eval\left(F_\phi(x,\cdot), c\right) \sim Enc_{sk}(0)$ and send it to P ($F_\phi \equiv 0$)

   If $b' = 1$ ➔ Compute $\hat{c} = Eval(F_L(x,\cdot), c) \sim Enc_{sk}\left(F_L(x, w)\right)$ and send it to P
3. $P \to V$: Decrypt $b = Dec_{sk}(\hat{c})$ and send the result to V
4. V accepts iff $b' = b$

ZK for any $L \in NP$

- Let $L \in NP$
- **Second attempt:**
  - Not ZK – a cheating V^* can run $Eval$ on any function and learn information about w (e.g., w's first bit)

ZK for any $L \in NP$

- Let $L \in NP$
- **Third attempt:**

ZK for any $L \in NP$

1. $P \rightarrow V$: Send to V an encryption of the witness $c = Enc_{sk}(w)$
2. $V \rightarrow P$: Flip a coin $b \in \{0,1\}$

   If $b' = 0$ ➔ Compute $\hat{c} = Eval\left(F_\phi(x,\cdot), c\right) \sim Enc_{sk}(0)$ and send it to P ($F_\phi \equiv 0$)

   If $b' = 1$ ➔ Compute $\hat{c} = Eval(F_L(x,\cdot), c) \sim Enc_{sk}\left(F_L(x,w)\right)$ and send it to P
3. $P \rightarrow V$: Decrypts $b = Dec_{sk}(\hat{c})$ and send a commitment $COM(b)$ to V
4. $V \rightarrow P$: Sends all its randomness
5. $P \rightarrow V$: Verifies that $\hat{c}$ is a valid homomorphic computation of F_L or F_ϕ on c. If yes ➔ Reveals b

   If no ➔ Stops
6. V accepts iff $b' = b$