# Cryptography and Network Security
# Interactive Proof

Xiang-Yang Li

# Interactive Proof

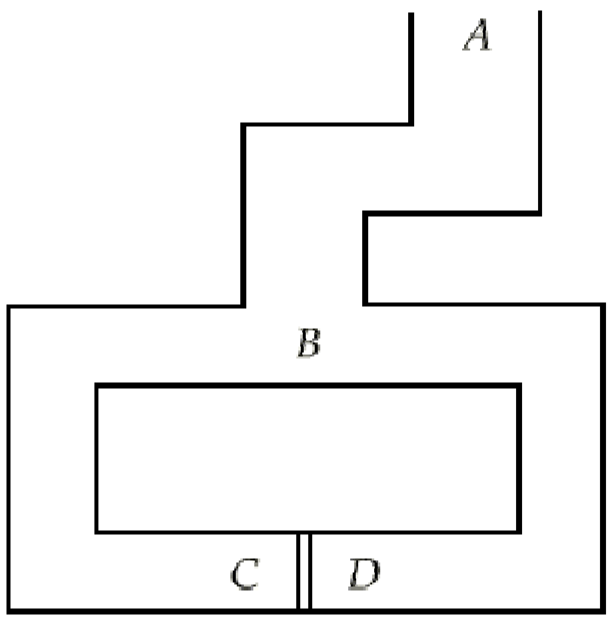
- *Interactive proof* is a protocol between two parties in which one party, called the *prover*, tries to prove a certain fact to the other party, called the *verifier*
- Often takes the form of a challenge-response protocol

# Desired Properties

- Desired properties of interactive proofs
  - *Completeness*: The verifier always accepts the proof if the prover knows the fact and both the prover and the verifier follow the protocol.
  - *Soundness*: Verifier always rejects the proof if prover doesnot know the fact, and verifier follows protocol.
  - *Zero knowledge*: The verifier learns nothing about the fact being proved (except that it is correct) from the prover that he could not already learn without the prover. In a zero-knowledge proof, the verifier cannot even later prove the fact to anyone else.

# An example

- Ali Baba's Cave



# Cont.

- Alice wants to prove to Bob that
  - she knows the secret words to open the portal at CD
  - but does not wish to reveal the secret to Bob.
  - In this scenario, Alice's commitment is to go to C or D.

# Proof Protocol

- A typical round in the proof proceeds as follows:
  - Bob goes to A, waits there while Alice goes to C or D.
  - Bob then asks Alice to appear from either the right side or the left side of the tunnel.
  - If Alice does not know the secret words
    - there is only a 50 percent chance that she will come out from the right tunnel.
  - Bob will repeat this round as many times as he desires until he is certain that Alice knows the secret words.
  - No matter how many times that the proof repeats, Bob does not learn the secret words.

# Graph Isomorphism

- Problem Instance
  - Two graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$
- Question
  - Is there a bijection f from $V_1$ to $V_2$, so (u,v)? $E_1$ implies that (f(u),f(v))? $E_2$
  - If such bijection exists, then graphs $G_1$ and $G_2$ are said to be isomorphic
  - If such bijection does not exist, then graphs $G_1$ and $G_2$ are said to be non-isomorphic

# Graph Non-isomorphism

- Input: graphs $G_1$ and $G_2$ over {1,2,….n}
- Prover want to prove
  - $G_1$ and $G_2$ are not isomophic
- Assumption
  - Prover has unbounded computational power
  - Verifier has limited computational power

# Proof Protocol

- Protocol (repeated for n rounds)
  - Verifier
    - Randomly chooses i=1 or 2
    - Selects a random permutation f and compute H to be the image of $G_i$ under f, sends H to prover
  - Prover
    - Determines the value j such that $G_j$ is isomorphic to H
    - Sends j to verifier
  - Verifier checks if j=i
  - If equal for n rounds, then accepts the proof

# Correctness and Soundness

- Correctness
  - If $G_1$ and $G_2$ are not isomorphic, then for any round, there is only one graph of $G_1$, $G_2$ that could produce H under a permutation f
  - So if the verifier knows non-isomorphism, then each round a correct j will be computed
- Soundness
  - If the verifier does not know ($G_1$ and $G_2$ are isomorphic), then each round two answers possible, and it has half chance to get the correct i chosen by the prover.

# Graph Isomorphism

- Input: graphs $G_1$ and $G_2$ over $\{1,2,\ldots n\}$
- Prover want to prove
  - $G_1$ and $G_2$ are isomophic
- Assumption
  - Prover has unbounded computational power
  - Verifier has limited computational power

# Proof Protocol

- Protocol (repeated for n rounds)
  - Prover
    - Selects a random permutation f and compute H to be the image of $G_1$ under f, sends H to prover
  - Verifier
    - Randomly chooses i=1 or 2, sends it to prover
  - Prover
    - Computes the permutation g such that H is the image of $G_j$ under g, and sends g to verifier
  - Verifier
    - checks if H is the image of $G_j$ under g
  - If yes for n rounds, then accepts the proof

# Correctness and Soundness

- Correctness
  - If $G_1$ and $G_2$ are isomorphic, and the verifier knows how to find the permutation between $G_1$ and $G_2$, then each round a correct g will be computed
- Soundness
  - If the verifier does not know ($G_1$ and $G_2$ are non-isomorphic or the permutation between $G_1$ and $G_2$), then each round prover can deceive the verifier is to guess the value i chosen by the verifier

# Perfect Zero-Knowledge

- The graph isomorphism proof is ZKP
  - All information seen by the verifier is the same as generated by a random simulator
  - Define transcript of the proof as
    - $t=(G_1,G_2,(H_1,i,g_1),(H_2,i,g_2),\ldots.(H_n,i,g_n))$
  - Anyone can generate the transcript without knowing which permutation carries $G_1$ to $G_2$
  - Hence the verifier gains nothing by knowing the transcript (I.e., the proof history)

# ZKP for Verifier

- Perfect Zero-knowledge for verifier
  - Suppose we have a poly-time interactive proof system and a poly-time simulator S. Let T be all yes-instance transcripts and let F be all transcripts generated by S. For any transcript t if
    - Pr(t occurs in T)=Pr(t occurs in F)
  - We say the interactive proof system are perfect zero-knowledge for the verifier

# Isomorphism Proof: ZKP-verifier

- Graph isomorphism is a perfect zero-knowledge for verifier
  - A triple (H,i,g). There are 2n! valid triples.
  - All triples (H,i,g) occurs equiprobable in some transcript
    - Here, assume that both the verifier and the prover are honest
    - Both of them randomly chooses parameters that supposed to be chosen randomly

# Cheating Verifier

- What happened if verifier does not follow the protocol (does not choose i randomly)
  - Transcript produced by ZKP is not same as that produced by the random simulator anymore
  - The verifier may gain some information due to this imbalance
  - But, there is another expected poly-time simulator to generate the same transcript
  - Hence, the verifier still gains nothing

# Perfect Zero-Knowledge

- Definition
  - Suppose we have a poly-time interactive proof system, a poly-time algorithm V to generate random numbers by verifier, and a poly-time simulator S. Let T be all yes-instance transcripts (depending on V) and let F be all transcripts generated by S and V. For any transcript t if
    - Pr(t occurs in T)=Pr(t occurs in F)
  - We say the interactive proof system are perfect zero-knowledge

# Forging Simulator

- Initial transcript $t=(G_1,G_2)$, repeat n rounds
  - Let old-state=state(V), repeat follows
    - Chooses $i_j$ from {1,2} randomly
    - Chooses $g_j$ to be a random permutation over {1,...n}
    - Compute $H_j$ to be the image of $G_i$ under g
    - Call V with input $H_j$, obtaining a challenge $i_j$'
    - If $i_j=i_j$', then concatenate ($H_j$, $i_j$, $g_j$) onto the end of t
    - Else reset V by state(V)=old-state
  - Until $i_j=i_j$'

# Perfect Zero-knowledge

- The graph isomorphism is perfect ZKP
  - The expected running time of simulator is 2n
  - For the $k^{th}$ round of the interactive proof system
    - Let $p_k$ be the probability that verifier chooses i=1
    - Then (H,1,g) occurs in actual transcript with $p_k/n!$, (H,2,g) occurs in actual transcript with $(1-p_k)/n!$
    - For simulator, when it terminates the simulation for the $k^{th}$ round, same probability distribution for (H,1,g) and (H,2,g)
    - Therefore, all transcripts by simulator or actual has the same probability distribution

# Quadratic Residue

- Question
  - Given integer n=pq, here p, q are primes.
  - Prover wants to prove
    - Integer x is a quadratic residue mod n
    - In other words, knows u so $x=u^2$ mod n
  - Quadratic residue is hard to solve if do not knowing the factoring of n

# Proof Protocol

- Repeat the following for $\log_2 n$ times
  - Prover
    - Chooses random v less than n and computes $y=v^2$ mod n. Sends y to verifier
  - Verifier
    - Chooses a random I from {0,1}, sends it to prover
  - Prover
    - Computes $z=u^2v$ mod n, sends z to verifier
  - Verifier
    - Checks if $z^2=x^iy$ mod n
  - Accepts the proof if equation holds all $\log_2 n$ rounds

# Bit Commitments

- Bit commitment
  - Sometimes, it is desirable to give someone a piece of information, but not commit to it until a later date. It may be desirable for the piece of information to be held secret for a certain period of time.
  - Example: stock up and down

# Properties

- Bit commitment scheme
  - The sender encrypts the b in some way
  - The encrypted form of b is called blob
  - Scheme f: (X,b)✍Y
- Properties
  - Concealing: verifier cannot detect b from f(x,b)
  - Binding: sender can open the blob by revealing x
  - Hence, the sender must use random x to mask b

# Methods

- One can choose any encryption method E
  - Function $f((x_0,k),b)=E_k((x_0,b))$
    - Need supply decryption k to reveal b
    - Assume the decryption method D is known
- Choose any integer n=pq, p and q are large primes
  - Function $f(x,b)=m^b x^2 \bmod n$
    - Goldwasser-Micali Scheme
    - Here n=pq, m is not quadratic residue, m,n public
    - $mx_1^2 \bmod n$ ? $x_2^2 \bmod n$
    - So sender can not change mind after commitment

# Coin Flip

- Even protocols
  - Alice has a coin flip result i or j
  - Bob wants to guess the result
  - Alice has a message M that is commitment
  - If bob guesses correct, Bob should have M received
  - Alice starts with 2 pairs of public keys (Ei,Di) and (Ej,Dj)
  - Bob starts with a symmetric encryption S and a key k

# Protocol

- Procedure
  - Alice sends Ei, Ej to Bob
  - Bob guess h and sends y=Eh(k) to Alice
  - Alice computes p=Dj(y) and sends the encryption z of M by p using S to Bob
  - Bob decrypts the encryption z using S and key k
  - If the guess is correct, then Bob gets the commitment

# Oblivious Transfer

- What is oblivious transfer
  - Alice wants to send Bob a secret in such a way that Bob will know whether he gets it, but Alice won't. Another version is where Alice has several secrets and transfers one of them to Bob in such a way that Bob knows what he got, but Alice doesn't. This kind of transfer is said to be oblivious (to Alice).

# Transfer Factoring

- By means of RSA, oblivious transfer of any secret amounts to oblivious transfer of the factorization of n=pq
  - Bob chooses *x* and sends $x^2$ mod n to Alice
  - Alice (who knows *p*,*q*) computes the square roots x,-x,y,-y of $x^2$ mod n and sends one of them to Bob. Note that Alice does not know *x*.
  - If Bob gets one of y or -y, he can factor *n*. This means that with probability 1/2, Bob gets the secret. Alice doesn't know whether Bob got one of y or -y because she doesn't know *x*.

# Factoring

- If one knows x and y such that
  - 1) $x^2=y^2$ mod n
  - 2) 0<x,y<n, x? y and x+y? 0 mod n
  - Number n is the production of two primes
- Then n can be factored
  - First gcd(x+y,n) is a factor of n
  - And gcd(x-y,n) is a factor of n

# Quadratic Solution

- Given n=p, and a is a quadratic residue
  - Then there is two positive integers x less than n
  - Such that $x^2$=a mod n
- Given n=pq, and a is a quadratic residue
  - Then there is four positive integers x less than n
  - Such that $x^2$=a mod n

# Oblivious Transfer of Message

- Alice has a message M, bob wants to get M through oblivious transfer
  - Alice does not know if Bob get M or not
  - Bob knows if he gets it or not
  - Bob gets M with probability ½
  - Coin flipping can be used to achieve this

# New Protocol

- ElGamal based protocol

# Contract Signing

- It requires two things
  - Commitment: after certain point, both parties are bound by the contract, until then, neither is
  - Unforgeability: it must be possible for either party to prove the signature of the other party
- With Pen and Paper
  - Two party together, face to face
  - Sign simultaneously (or one character by one)

# Remote Contract Signing

- Simple one
  - Alice generate a signature, divided into SL, SR
  - Alice randomly select two keys KL, KR
  - Encrypt the signatures SL, SR
  - Transfer encrypted SL,SR to Bob
  - Obliviously transfer KL, KR to bob
    - Bob gets one, but Alice does not know which one
  - Bob decrypts the encrypted SL or SR
    - Verify the decrypted signature, if invalid, stop
  - Alice sends the ith bits of keys KL and KR to Bob
    - Here i=1 to the length of the keys

# Cont.

- The protocol will be conducted by Bob also
  - What is the chance of Alice to cheat successfully?
    - Alice can guess which key will be transferred obliviously --- (1/2 chance)
    - Then send wrong signature for the other half or send the wrong key of the other half
    - Bob can not detect it if Alice can guess which key Bob got
  - How about Alice stop prematurely?
    - One bit advance over Bob
- Enhanced protocol
  - Use many pair of keys and signatures instead of one