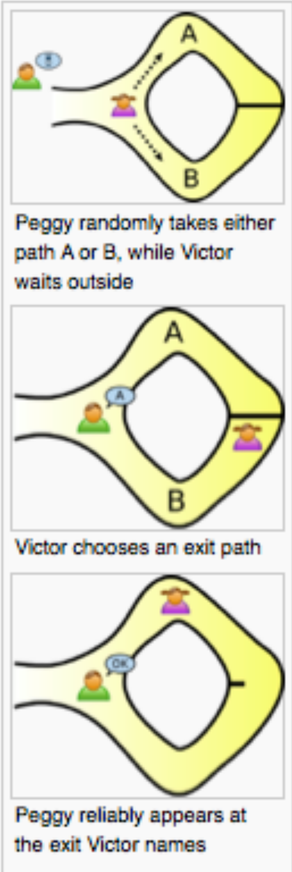
Interactive proof and zero knowledge protocols

- Zero-knowledge: definition
- Probabilistic complexity classes and Interactive proofs
  - Graph isomorphism and PCP
- Some zero knowledge protocols:
  - Feige-Fiat-Shamir authentication protocol
  - Extension to signature
  - Guillou-Quisquater authentication and signature

- Computational Complexity: A Modern Approach. Sanjeev Arora and Boaz Barak http://www.cs.princeton.edu/theory/complexity/
- Handbook of Applied Cryptography [Menzenes, van Oorschot, Vanstone]
- Applied Cryptography [Schneier]
- Contemporary cryptography [Opplinger]

Example [wikipedia]

- Ali Baba (Peggy) knows the secret
  - "iftaH ya simsim" («Open Sesame»)
  - "Close, Simsim" («Close Sesame»).
- Bob (Victor) and Ali Baba design a protocol to prove that Ali Baba has the secret without revealing it
  - Ali Baba is *the prover*
  - Bob is *the verifier*
  - Ali Baba leaks no information



Peggy randomly takes either path A or B, while Victor waits outside

Victor chooses an exit path

Peggy reliably appears at the exit Victor names

Proof and Interactive proof

- Importance of « proof » in crypto: eg. identity proof=authentication
- Two parts in a proof:
  - Prover: knows the proof (-> the secret) [or is intended to know]
  - Verifier: verifies the proof is correct (-> authentication)
- Correctness of a proof system/verifier:
  - **Soundness**: ***<u>every invalid proof</u>*** **is rejected** by the verifier
  - **Completeness**: ***every valid proof*** **is accepted** by the verifier
- Interactive proof system
  - Protocol (questions/answers) between the verifier and the prover
  - Verifier: **probabilistic** algorithm, **polynomially bounded**
  - Soundness: every invalid proof is rejected with probability (> 1/2)
  - Competeness: every valid proof is accepted with probability (>1/2)

Interactive protocol :Example

- Example: interactive authentication based on quadratic residue
- See exercise (question 3.b)
  - Completeness : Alice, who gets the secret (square root) is accepted
  - But not Soundness : Eve, who doesn't know the secret may cheat
- Fiat-Shamir's protocol (question 3.c)
  - Soundness : Eve, who doesn't know the secret, is rejected.(if we assume n factorization unknown)

Does x belongs to L ?

- Verifier
  - An element x
  - Ask questions to prover
  - Gets anwer:
  - Completeness: Is convinced that x in L, if so
  - Soundess: reject « x in L » if not so
- Zero-knowledge:
  - Intuitively: at the end, verifier is convinced that x in L (if so), but *learns nothing else*.

Example of interactive computation

- Graph isomorphism:
  - Input: G=(V,E) and G'=(V',E')
  - Output: YES iff G == G' (i.e. a permutation of V ->V' makes E=E')
- NP-complete, not known to be in co-NP
- Assume an NP Oracle for Graph isomorphism => then a probabilistic verifier can compute Graph isomorphism in polynomial time.
  - Protocol and error probability analysis.
- Theorem [Goldreich&al] :
  - NP included in IP.
  - any language in NP possesses a zero-knowledge protocol.

Interactive Algorithm Graph Isomorhism

AlgoGraphIso($G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$) {
  If ($\#V_1$!= $\#V_2$) or ($\#E_1$!= $\#E_2$)
    return "NO : G_1 not isomorphic to G2";
  n := $\#V_1$;
  For (i=1 .. k) {
    P := randompermutation([1, …, n]) ;
    b := random({1,2}) ;
    G' := $P(G_b)$;
    (i, P_i) := Call **OracleWhichIsIso**(G_1, G_2, G') ;
    If ($G_i \neq P_i$ (G')) FAILURE("Oracle is not reliable") ;
    If (b ≠ i) **return "YES : G_1 is isomorphic to G_2"** ;
  }
  return "NO : G_1 not isomorphic to G_2";
}

OracleWhichIsIso(G_1, G_2, G') {
  // precondition: G' is isomorphic to
  // G_1 or G_2 or both.
  // Output: i into {1,2} and a permutation
  // P_i such that G_i = P(G')
  … ;
  Return (i, P_i) ;
}

Theorem: Assuming OracleWhichIsIso of polynomial time,
AlgoGraphIso(G_1, G_2) proves in polynomial time $k.n^{O(1)}$ that :
- either G_1 is isomorphic to G_2 (no error)
- or G_1 is not isomorphic with error probability $\leq 2^{-k}$.

Thus, it is a MonteCarlo (randomized) algorithm for GRAPH ISOMORPHISM

Analysis of error probability

Prob(Output of AlgoGraphIso(G_1, G_2)) / *Truth: $G_1 = G_2$??*	*"YES : G_1 is isomorphic to G_2"*	*"NO: G_1 not isomorphic to G_2"*
Case $G_1 = G_2$ (completeness)	Prob = $1 - 2^{-k}$	Prob = 2^{-k}
No: Case $G_1 \neq G_2$ (soundness)	Impossible (Prob = 0)	Always (Prob = 1)

-When the algorithm output YES : G_1 is isomorphic to G_2 then $G_1 = G_2$
=> no error on this output.

-When the algorithm output "NO: G_1 not isomorphic to G_2" then we may have an error (iff $G_1 = G_2$), but with a probability $\leq 2^{-k}$

One-sided error => Monte Carlo algorithm for Graph-Isomorphism

Complexity classes

Decision problems (1 output bit: YES/ NO)

Deterministic polynomial time:

- P : both Yes/No sides
- NP : certification for the Yes side
- co-NP: certification for the No side

Randomized polynomial time:

- BPP: Atlantic City: prob(error) < 1/2
- RPP: Monte Carlo: prob(error YES side)=0 ; prob(error NO side)< 1/2
- ZPP: Las Vegas: prob(failure)<1/2 but prob(error)=0

IP Interactive proof

- Verifier: randomized polynomial time
- Prover: interactive (dynamic), unbound power
  - F(x) = YES => it exists a correct prover Π such that Prob[Verifier (Π, x) accepts] = 1;
  - F(x) = NO => for all prover Π: Prob[Verifier (Π, x) accepts] < 1/2.
- Theorem: IP = PSPACE

PCP: Probabilistiic Checkable Proofs (static proof)

- PCP(r, q) : the verifier uses random bits and reads q bits of the proof only.
- Theorem: NP=PCP(log n, O(1))

Summary

- Interactive proof : generalization of a mathematical proof in which prover and polynomial-time probabilistic verifier interact:
  - Completeness and soundness
- Input: x, proof of property L(x)
  Correct proof: x is accepted iff L(x) is true.
  - Completeness : any x: L(x)=true is accepted (with prob≥2/3).
  - Soundess : any y: L(y)=false is rejected (with prob≥2/3).
- Power of interactive proof w.r.t. « static » proof
  - IP = PSACE

Zero knowledge

- How to prove zero knowledge: by proving the verifier could have produced the transcript of the protocol in (expected) polynomial time with no help of the prover.
- **Def:** a sound and correct interactive protocol is zero-knowledge if there exists a non-interactive randomized polynomial time algorithm (named « **simulator** ») which, for any input x accepted by the verifier (using interaction with the prover) can produce transcripts indistinguishable from those resulting from interaction with the real prover.
- **Consequence:** releases no information to an observer.

Graph [non]-isomorphism and zero knowledge

- In a zero-knowledge protocol, the verifier learns that G_1 is isomorphic to G_2 but nothing else.
- Previous protocol (slide 7) not known to be zero-knowledge:
  - Prover sends the permutation P_i such that $G_1 = P_i(G_2)$: so the verifier learns not only G_1 isomorphic to G_2 but P_i too.
  - We do not know, given two isomorphic graph, wether there exists a (randomized) polynomial time algorithm that returns a permutation that proves isomorphism.

A zero-knowledge interactive proof for Graph Isomorhism

Verifier

input: ($G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$)

Accepts prover if convinced that G1 is isomorphic to G2

2. Receives H;

Chooses b=random(1,2) and sends b to the prover

4. receives P'' and checks H = P''(G_b)

Proover

gets G_1, G_2

private secret perm. P_s: $G_2=P_s(G_1)$;

1. Chooses a random perm. P' and sends to verifier H=P'(G_2)

3. Receives b;

if b=1 sends P''=P'oP_s to the verifier

else b=2: sends P''=P' to the verifier

Theorem: This is a zero-knowledge, sound and complete, polynomial time interactive proof that the two graphs G_1 and G_2 are isomorph.

Zero-knowledge interactive proof for Graph Isomorhism

- Completeness
- Soundness
- Zero-knowledge
- Polynomial time

Zero-knowledge interactive proof for Graph Isomorhism

- Completeness
  - if $G_1=G_2$, verifier accepts with probability 1.
- Soundness
  - if $G_1 \neq G_2$, verifier rejects with probability ≥ ½
- Zero-knowledge
  - Simulation algorithm:
    1. Choose first b=rand(1,2) and π random permutation (like P');
    2. Compute $H = \pi(G_b)$;
    3. Output transcript [H, b, π] ;
  - The transcript [H, b, π] is distributed uniformly, exactly as the transcript [H, b, P'] in the interactive protocol.
- Polynomial time

Another simulation algorithm

- Without changing the verifier, by just modifying the prover:

  Do {

  1. b' = random(1,2) and π=random(permutation);
     Compute $H=\pi(G_{b'})$ and send H to verifier;

  3. receive b ;

  } while (b ≠ b') ;

  Output transcript [H, b, π]

- Polynomial time:
  - Expectation time = $\text{Time}_{\text{Loop\_body}} \cdot \sum_{k \geq 0} 2^k \leq 2.\text{Time}_{\text{Loop\_body}}$

Exercise

- Provide an interactive polynomial time protocol to prove a verifier that has an integer N that you know the factorization N=P.Q without revealing it.
  - Application:
    - a sensitive building, authorized people know 2 secret primes P and Q (and N=PQ)
    - The guard knows only N

Quadratic residue authentication: is this version **perfectly** zero-knowledge?

- A **trusted part T** provides a Blum integer n=p.q; n is public.

- **Alice (Prover) builds her secret and public keys:**
  - For i=1, …, k: chooses at random s_i coprime to n
  - Compute $v_i := (s_i^2)$ *mod* n. [NB v_i ranges over all square coprime to n]
    v_i = ***quadratic residue*** that admits s_i = ***modular square root***
  - Secret key: s_1 , …, s_k
  - Public key: v_1 , …, v_k and identity photo, … registered by T

- **Bob (Verifier)** authenticates Alice: **Zero-knowledge protocol in 3 messages** :
  1. Alice chooses a random r<n; she sends $y=r^2$ *mod* n to Bob.
  2. Bob sends k random bits: b_1 , …, b_k
  3. Alice computes $z := rs_1^{b_1}. \dots . s_k^{b_k}$ mod *n* and sends *z* to Bob.
     Bob authenticates iff $z^2 = y.v_1^{b_1}. \dots .v_k^{b_k}$ mod n.

- **Simulation algorithm** : *is the protocol perfectly zeo-knowledge?*
  1. Choose k random bits b_1 , …, b_k and a random z<n;
     compute $w= v_1^{b_1}. \dots .v_k^{b_k}$ mod n and $y=z^2 .w^{-1}$ mod n ;
  2. Transcript is [y ; b_1 , …, b_k ; z]

Feige-Fiat-Shamir
zero-knowledge authentication protocol

- A **trusted part T** computes a Blum integer n=p.q; n is public.

- **Alice (Prover) builds her secret and public keys:**
  - For i=1, …, k: chooses at random s_i coprime to n
  - Compute $v_i := (s_i^2)$ *mod* n. [NB v_i ranges over all square coprime to n]
    v_i = ***quadratic residue*** that admits s_i = ***modular square root***
  - Secret key: s_1 , …, s_k
  - Public key: v_1 , …, v_k and identity photo, … registered by T

- **Bob (Verifier)** authenticates Alice: **Zero-knowledge protocol in 3 messages** :
  1. Alice chooses a random r<n and a sign u=±1; she sends $y=u.r^2$ *mod* n to Bob.
  2. Bob sends k random bits: b_1 , …, b_k
  3. Alice computes $z := r.\ s_1^{b_1}.\ \ldots\ .\ s_k^{b_k}$ mod *n* and sends *z* to Bob.
     Bob authenticates iff $z^2 = +/-\ y.v_1^{b_1}.\ \ldots\ .v_k^{b_k}$ mod n.

- Remark: possible variant: Alice chooses its own modulus n

Feige-Fiat-Shamir

Truth: X=Alice or anyone else? \ *Prob(Output of authentication)*	***YES: "Authentication of Alice OK"***	***NO: "Authentication of Alice KO »***
Case X = Alice (completeness)	Always	Impossible
Case X ≠ Alice (soundness)	Prob = 2^{-k}	Prob = 1 - 2^{-k}

- **Completeness**
  - Alice is allways authenticated (error prob=0)

- **Soundness**
  - Probability for Eve to impersonate Alice = 2^{-k}. If t rounds are performed: 2^{-kt}

- **Zero-knowledge**
  - A simulation algorithm exists that provides a transcript which is indistinguishable with the trace of interaction with correct prover.

From zero-knowledge authentication to zero knowledge signature

- Only one communication: the message+signature
  - The prover uses a CSPRNG (e.g. a secure hash function) to generate directly the random bits of the challenge
  - The bits are transmitted to the verifier, who verifies the signature.

- Example: Fiat-Shamir signature
  - Alice builds her secret key (s_1 , …, s_k) and public key (v_1 , …, v_k) as before.
  - Let M be a message Alice wants to sign.
  - Signature by Alice
    1. For i=1, …, t: chooses randomly r_i and computes w_i s.t. $w_i := r_i^2$ *mod* n.
    2. Computes h = H(M || w_1 ||…|| w_t) this gives k.t bits b_{ik}, that appear as random (similarly to the ones generated by Bob in step 2 of Feige-Fiat-Shamir)
    3. Alice computes $z_i := r_i . s_1^{b_{i1}} . \ldots . s_k^{b_{ik}}$ *mod* n (for i = 1 .. t) ;
       She sends the message M and its signature: $\sigma = (z_1 \ldots z_t, b_{11} .. b_{tk})$ to Dan
  - Verification of signature σ by Dan:
    1. Dan computes $y_i := z_i^2 .(v_1^{b_{i1}} . \ldots . v_k^{b_{ik}})^{-1}$ *mod* n for i=1..t
       A correct signature gives $y_i = w_i$
    2. Computes H(M, || y_1 ||…|| y_t) and
       he verifies that he obtains the bits b_{ik} in Alice's signature

Zero-knowledge vs other asymetric protocols

- No degradation with usage.
- No need of encryption algorithm.
- Efficiency: often higher communication/computation overheads in zero-knowledge protocols than public-key protocols.
- For both , provable security relies on conjectures
  (eg: intractability of quadratic residuosity)

Exercise

- Guillou-Quisquater zero-knowledge authentication and signature protocol.

Feige-Fiat-Shamir zero-knowledge authentication protocol

- A **trusted part T** (or Alice) computes a Blum integer n=p.q; n is public.
- **Alice (Prover) builds her secret and public keys:**
  - For i=1, …, k: chooses at random s_i coprime to n and n random bits d_i
  - Compute $v_i := (s_i^2)$ *mod* n. [NB v_i ranges over all square coprime to n]
    $(-1)^{d_i} v_i$ = ***quadratic residue*** that admits s_i = ***modular square root***
  - Secret key: $s_1, \ldots, s_k$. (Note that $v_i . s_i^2 = (-1)^{d_i}$ = 1 or -1 mod n)
  - Public key: $v_1, \ldots, v_k$ and identity photo, … registered by T

- **Bob (Verifier)** authenticates Alice: **Zero-knowledge protocol in 3 msgs** :
  1. Alice chooses a random value r < n. She sends $y := r^2$ *mod* n to Bob.
  2. Bob sends k random bits: $b_1, \ldots, b_k$
  3. Alice computes $z := r . s_1^{b_1} . \ldots . s_k^{b_{k\,i}}$ mod *n* and sends *z* to Bob.
     Bob computes $w = z^2 . v_1^{b_1} . \ldots . v_k^{b_k}$ and authenticates iff y=w or y=-w mod n.

- Soundness and completeness, perfectly zero knowledge
  - Probability for Eve to impersonate Alice = 2^{-k}. If t rounds are performed: 2^{-kt}
  - Alice always authenticated (error prob=0)
  - Zero knowledge: transcript

IP and NP

Complexity classes

Decision problems (1 output bit: YES/ NO)

Deterministic polynomial time:

- P : both Yes/No sides
- NP : certification for the Yes side
- co-NP: certification for the No side

Randomized polynomial time:

- BPP: Atlantic City: prob(error) < 1/2
- RPP: Monte Carlo: prob(error YES side)=0 ; prob(error NO side)< 1/2
- ZPP: Las Vegas: prob(failure)<1/2 but prob(error)=0

IP Interactive proof

- Verifier: randomized polynomial time
- Prover: interactive (dynamic), unbound power
  - F(x) = YES => it exists a correct prover Π such that Prob[Verifier (Π, x) accepts] = 1;
  - F(x) = NO => for all prover Π: Prob[Verifier (Π, x) accepts] < 1/2.
- Theorem: IP = PSPACE (interaction with randomized algorithms helps!)

PCP: Probabilistiic Checkable Proofs (static proof)

- PCP(r, q) : the verifier uses random bits and reads q bits of the proof only.
- Theorem: NP=PCP(log n, O(1))

#3-SAT in IP

- Arithmetization in F_2: each clause c has a poly. Q(c)
  - Q(not(x)) = 1-x   Q(x and y) = x.y
  - Q(x or not(y) or z)=Q(not(not(x) and y and not(z))= 1–((1-x).y.(1-z))
- Let F = c_1 and … and c_m a 3-SAT CNF formula, and $g(X_1, …, X_n) = Q(c_1).Q(C_2). … .Q(c_m)$:   deg(g) ≤ 3m
  Then $\#F = \Sigma_{b_1=0,1} … \Sigma_{b_n=0,1}\, g(b_1, …, b_n)$
- Since #F ≤ 2^n, for p>2^n,   (#F=K) is equivalent to (#F=K mod p)
  – To limit to a polynomial number of operations, computation is performed mod a prime p in 2^n .. 2^{n+1} (provided by prover and checked by verifier)
- Let $h_n(X_n) = \Sigma_{b_1=0,1} … \Sigma_{b_{n-1}=0,1} g(b_1, b_2, …, b_{n-1}, X_n)$:
  h_n is an univariate polynomial (in X_n) of degree ≤ m

#3-SAT: interactive polynomial proof

Verifier

input: $F(X_1, …, X_n) = (c_1$ and … and $c_m)$
K an integer; let $g(x) = \Pi_{i=1,n} Pol(c_i)$
Accepts iff convinced that #F = K.
Preliminar receive p, check p is prime in {2^n, 2^{2n}}
Compute $g(X_1, …, X_n)= \Pi_{i=1,n} Pol(c_i)$ deg(g)≤3m
Check $K= \Sigma_{X1=0,1} … \Sigma_{Xn=0,1}\, g(X_1, …, X_n)$ [p] :

1. If n=1, if (g(0)+g(1) = K) accept ; else reject.
   If n≥2, ask $h_n(X)$ to P.

3. Receive s(X) of degree ≤m.
   Compute v=s(0)+s(1); if (v ≠ K) reject.
   Else choose r=random(0, … p-1); let K_n = s(r)
   and use the same protocol to check
   $K_n=\Sigma_{X1=0,1} … \Sigma_{Xn-1=0,1}\, g(X_1, …, X_{n-1}, r)$ [p]

Prover

Preliminar: sends p prime in {2^n, 2^{2n}}

2. Send s(X) ; [note that if P is not cheating, s(X) = $h_n(X)$]

Theorem: This is a sound and complete, polynomial time randomized interactive proof of #3-SAT.
Moreover, prob(V rejects | K ≠ #F) ≥ (1-m/p)^n ,
also prob(error) ≤ 1-(1-m/p)^n ≤ $mn2^{-n}$.

The End.

What have we learned?

- Perfect secrecy: the ciphertext has always the same distribution, it provides no information on the plaintext.
  - Eg: OTP
- Computational security :
  - Based on the assumption that a one-way function exists.
    - So that P ≠ NP

- One way-function and crypto hash functions
  - Compression + extension scheme (with padding)
  - Sponge construction
  - Encryption from one-way function with short keys (of length n^{-c}) to encrypt long messages (of length n)
  - One-way from block cipher

- Secure pseudo-random generator
  - Indistinguishability from true random (deskewing)
  - Left and right unpredicability

- Interactive zero knowledge protocol
  - Soundness + completness
  - Zero-knowledge: simulation that provides a transcript indistinguighable from the correct interaction!