One-Way Functions

A function f is a **one-way function** if the following hold.[a]

1. f is one-to-one.
2. For all $x \in \Sigma^*$, $|\, x\,|^{1/k} \leq |f(x)| \leq |\, x\,|^k$ for some $k > 0$.
   - f is said to be **honest**.
3. f can be computed in polynomial time.
4. f^{-1} cannot be computed in polynomial time.
   - Exhaustive search works, but it must be slow.

[a] Diffie & Hellman (1976); Boppana & Lagarias (1986); Grollmann & Selman (1988); Ko (1985); Ko, Long, & Du (1986); Watanabe (1985); Young (1983).

Existence of One-Way Functions (OWFs)

- Even if P $\neq$ NP, there is no guarantee that one-way functions exist.
- No functions have been proved to be one-way.
- Is breaking glass a one-way function?

Candidates of One-Way Functions

- Modular exponentiation $f(x) = g^x \bmod p$, where g is a primitive root of p.
  - **Discrete logarithm** is hard.[a]
- The RSA[b] function $f(x) = x^e \bmod pq$ for an odd e relatively prime to $\phi(pq)$.
  - Breaking the RSA function is hard.

---

[a]Conjectured to be 2^{n^ϵ} for some $\epsilon > 0$ in both the worst-case sense and average sense. Doable in time $n^{O(\log n)}$ for finite fields of small characteristic (Barbulescu, et al., 2013). It is in NP in some sense (Grollmann & Selman, 1988).

[b]Rivest, Shamir, & Adleman (1978).

Candidates of One-Way Functions (concluded)

- Modular squaring $f(x) = x^2 \bmod pq$.
  - Determining if a number with a Jacobi symbol 1 is a quadratic residue is hard—the **quadratic residuacity assumption** (**QRA**).[a]
  - Breaking it is as hard as factorization when $p \equiv q \equiv 3 \bmod 4$.[b]

---

[a] Due to Gauss.

[b] Rabin (1979).

The Secret-Key Agreement Problem

- Exchanging messages securely using a private-key cryptosystem requires Alice and Bob have the *same* key.[a]
  - An example is the r in the one-time pad.[b]
- How can they agree on the same secret key when the channel is insecure?
- This is called the **secret-key agreement problem**.
- It was solved by Diffie and Hellman (1976) using one-way functions.

---

[a]See p. 662.
[b]See p. 661.

The Diffie-Hellman Secret-Key Agreement Protocol

1: Alice and Bob agree on a large prime p and a primitive root g of p; $\{p$ and g are public.$\}$

2: Alice chooses a large number a at random;

3: Alice computes $\alpha = g^a \bmod p$;

4: Bob chooses a large number b at random;

5: Bob computes $\beta = g^b \bmod p$;

6: Alice sends α to Bob, and Bob sends β to Alice;

7: Alice computes her key $\beta^a \bmod p$;

8: Bob computes his key $\alpha^b \bmod p$;

Analysis

- The keys computed by Alice and Bob are identical as

$$\beta^a = g^{ba} = g^{ab} = \alpha^b \bmod p.$$

- To compute the common key from p, g, α, β is known as the **Diffie-Hellman problem**.
- It is conjectured to be hard.[a]
- If discrete logarithm is easy, then one can solve the Diffie-Hellman problem.
  - Because a and b can then be obtained by Eve.
- But the other direction is still open.

[a] This is the **computational Diffie-Hellman assumption** (CDH).

The RSA Function

- Let p, q be two distinct primes.
- The RSA function is $x^e \bmod pq$ for an odd e relatively prime to $\phi(pq)$.
  - By Lemma 59 (p. 501),

$$\phi(pq) = pq\left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right) = pq - p - q + 1. \quad (16)$$

- As $\gcd(e, \phi(pq)) = 1$, there is a d such that

$$ed \equiv 1 \bmod \phi(pq),$$

which can be found by the Euclidean algorithm.[a]

[a] One can think of d as e^{-1}.

A Public-Key Cryptosystem Based on RSA

- Bob generates p and q.
- Bob publishes pq and the encryption key e, a number relatively prime to $\phi(pq)$.
  - The encryption function is
    $$y = x^e \bmod pq.$$
  - Bob calculates $\phi(pq)$ by Eq. (16) (p. 676).
  - Bob then calculates d such that $ed = 1 + k\phi(pq)$ for some $k \in \mathbb{Z}$.

A Public-Key Cryptosystem Based on RSA (continued)

- The decryption function is

$$y^d \bmod pq.$$

- It works because

$$y^d = x^{ed} = x^{1+k\phi(pq)} = x \bmod pq$$

by the Fermat-Euler theorem when $\gcd(x, pq) = 1$ (p. 506).

A Public-Key Cryptosystem Based on RSA (continued)

- What if x is not relatively prime to pq?[a]
- As $\phi(pq) = (p-1)(q-1)$,

$$ed = 1 + k(p-1)(q-1).$$

- Say $x \equiv 0 \bmod p$.
- Then

$$y^d \equiv x^{ed} \equiv 0 \equiv x \bmod p.$$

[a]Of course, one would be unlucky here.

A Public-Key Cryptosystem Based on RSA (continued)

- Either $x \not\equiv 0 \bmod q$ or $x \equiv 0 \bmod q$.
- If $x \not\equiv 0 \bmod q$, then

$$\begin{aligned} y^d &\equiv x^{ed} \equiv x^{ed-1} x \equiv x^{k(p-1)(q-1)} x \equiv \left(x^{q-1}\right)^{k(p-1)} x \\ &\equiv x \bmod q. \end{aligned}$$

  by Fermat's "little" theorem (p. 504).

- If $x \equiv 0 \bmod q$, then

$$y^d \equiv x^{ed} \equiv 0 \equiv x \bmod q.$$

A Public-Key Cryptosystem Based on RSA (concluded)

- By the Chinese remainder theorem (p. 503),

$$y^d \equiv x^{ed} \equiv 0 \equiv x \bmod pq,$$

even when x is not relatively prime to p.

- When x is not relatively prime to q, the same conclusion holds.

The "Security" of the RSA Function

- Factoring pq or calculating d from (e, pq) seems hard.
- Breaking the last bit of RSA is as hard as breaking the RSA.[a]
- Recommended RSA key sizes:[b]
  - 1024 bits up to 2010.
  - 2048 bits up to 2030.
  - 3072 bits up to 2031 and beyond.

---

[a] Alexi, Chor, Goldreich, & Schnorr (1988).

[b] RSA (2003). RSA was acquired by EMC in 2006 for 2.1 billion US dollars.

The "Security" of the RSA Function (continued)

- Recall that problem A is "harder than" problem B if solving A results in solving B.
  - Factorization is "harder than" breaking the RSA.
  - It is not hard to show that calculating Euler's phi function[a] is "harder than" breaking the RSA.
  - Factorization is "harder than" calculating Euler's phi function (see Lemma 59 on p. 501).
  - So factorization is harder than calculating Euler's phi function, which is harder than breaking the RSA.

[a] When the input is not factorized!

The "Security" of the RSA Function (concluded)

- Factorization cannot be NP-hard unless NP = coNP.[a]
- So breaking the RSA is unlikely to imply P = NP.
- But numbers can be factorized efficiently by quantum computers.[b]
- RSA was alleged to have received 10 million US dollars from the government to promote unsecure p and q.[c]

---

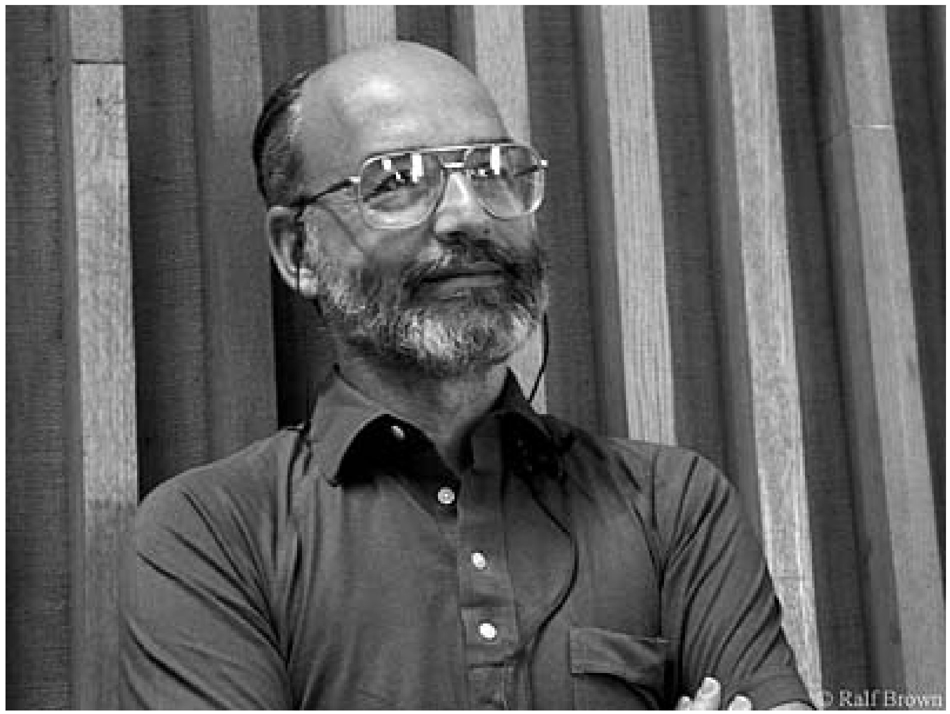
[a] Brassard (1979).
[b] Shor (1994).
[c] Menn (2013).

Adi Shamir, Ron Rivest, and Leonard Adleman

Ron Rivest[a] (1947–)

[a]Turing Award (2002).

Adi Shamir[a] (1952–)



[a]Turing Award (2002).

A Parallel History

- Diffie and Hellman's solution to the secret-key agreement problem led to public-key cryptography.
- In 1973, the RSA public-key cryptosystem was invented in Britain before the Diffie-Hellman secret-key agreement scheme.[a]

---

[a] Ellis, Cocks, and Williamson of the Communications Electronics Security Group of the British Government Communications Head Quarters (GCHQ).

Is a forged signature the same sort of thing
as a genuine signature,
or is it a different sort of thing?
— Gilbert Ryle (1900–1976),
The Concept of Mind (1949)

"Katherine, I gave him the code.
He verified the code."
"But did you verify him?"
— *The Numbers Station* (2013)

Digital Signatures[a]

- Alice wants to send Bob a *signed* document x.
- The signature must unmistakably identifies the sender.
- Both Alice and Bob have public and private keys

$$e_{\text{Alice}}, e_{\text{Bob}}, d_{\text{Alice}}, d_{\text{Bob}}.$$

- Every cryptosystem guarantees $D(d, E(e, x)) = x$.
- Assume the cryptosystem also satisfies the commutative property

$$E(e, D(d, x)) = D(d, E(e, x)). \tag{17}$$

  – E.g., the RSA system satisfies it as $(x^d)^e = (x^e)^d$.

---

[a] Diffie & Hellman (1976).

Digital Signatures Based on Public-Key Systems

- Alice signs x as
$$(x, D(d_{\text{Alice}}, x)).$$
- Bob receives (x, y) and verifies the signature by checking
$$E(e_{\text{Alice}}, y) = E(e_{\text{Alice}}, D(d_{\text{Alice}}, x)) = x$$
based on Eq. (17).
- The claim of authenticity is founded on the difficulty of inverting E_{Alice} without knowing the key d_{Alice}.

Blind Signatures[a]

- There are applications where the document author (Alice) and the signer (Bob) are *different* parties.
- Sender privacy: We do not want Bob to see the document.
  - Anonymous electronic voting systems, digital cash schemes, anonymous payments, etc.
- Idea: The document is **blinded** by Alice before it is signed by Bob.
- The resulting blind signature can be publicly verified against the original, unblinded document x as before.

[a]Chaum (1983).

Blind Signatures Based on RSA

Blinding by Alice:

1: Pick $r \in Z_n^*$ randomly;

2: Send

$$x' = xr^e \bmod n$$

to Bob using his public encryption key e; $\{x$ is blinded by $r^e.\}$

- Note that $r \rightarrow r^e \bmod n$ is a one-to-one correspondence.
- Hence $r^e \bmod n$ is a random number, too.
- As a result, x' is random and leaks no information, even if x has any structure.

Blind Signatures Based on RSA (continued)

Signing by Bob with his private decryption key d:

1: Send the blinded signature

$$s' = (x')^d \bmod n$$

to Alice;

Blind Signatures Based on RSA (continued)

The RSA signature of Alice:

1: Alice obtains the signature $s = s' r^{-1} \bmod n$;

- This works because

  $$s \equiv s' r^{-1} \equiv (x')^d r^{-1} \equiv (x r^e)^d r^{-1} \equiv x^d r^{ed-1} \equiv x^d \bmod n$$

  by the properties of the RSA function.

- Note that only Alice knows r.

Blind Signatures Based on RSA (concluded)

- Anyone can verify the document was signed by Bob by checking with Bob's encryption key e the following:

$$s^e \equiv x \bmod n.$$

  - This works because

$$s^e \equiv \left(x^d\right)^e \equiv x \bmod n.$$

- But Bob does not know s is related to x' (thus Alice).

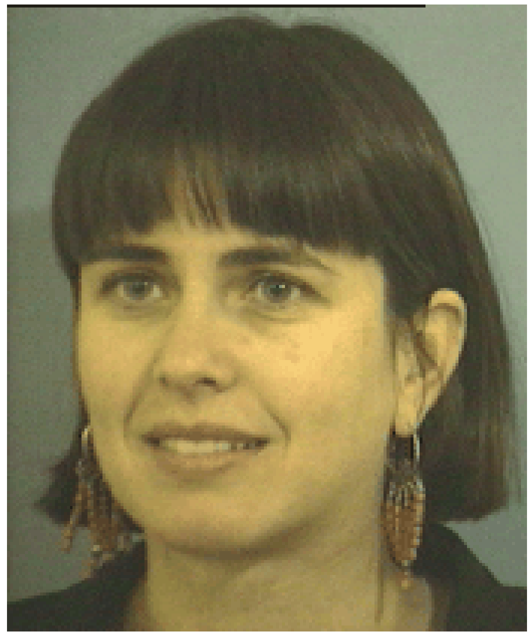
Probabilistic Encryption[a]

- A deterministic cryptosystem can be broken if the plaintext has a distribution that favors the "easy" cases.
- The ability to forge signatures on even a vanishingly small fraction of strings of some length is a security weakness if those strings were the probable ones!
- A scheme may also "leak" *partial* information.
  – Parity of the plaintext, e.g.
- The first solution to the problems of skewed distribution and partial information was based on the QRA.

---

[a]Goldwasser & Micali (1982). This paper "laid the framework for modern cryptography" (2013).

Shafi Goldwasser[a] (1958–)



[a]Turing Award (2013).

Silvio Micali[a] (1954–)

[a]Turing Award (2013).

Goldwasser and Micali

A Useful Lemma

Lemma 82 *Let $n = pq$ be a product of two distinct primes. Then a number $y \in Z_n^*$ is a quadratic residue modulo n if and only if $(y \,|\, p) = (y \,|\, q) = 1$.*

- The "only if" part:
  - Let x be a solution to $x^2 = y \bmod pq$.
  - Then $x^2 = y \bmod p$ and $x^2 = y \bmod q$ also hold.
  - Hence y is a quadratic modulo p and a quadratic residue modulo q.

The Proof (concluded)

- The "if" part:
  - Let $a_1^2 = y \bmod p$ and $a_2^2 = y \bmod q$.
  - Solve

$$\begin{aligned} x &= a_1 \bmod p, \\ x &= a_2 \bmod q, \end{aligned}$$

    for x with the Chinese remainder theorem (p. 503).
  - As $x^2 = y \bmod p$, $x^2 = y \bmod q$, and $\gcd(p, q) = 1$, we must have $x^2 = y \bmod pq$.

The Jacobi Symbol and Quadratic Residuacity Test

- The Legendre symbol can be used as a test for quadratic residuacity by Lemma 69 (p. 573).
- Lemma 82 (p. 701) says this is *not* the case with the Jacobi symbol in general.
- Suppose $n = pq$ is a product of two distinct primes.
- A number $y \in Z_n^*$ with Jacobi symbol $(y \,|\, pq) = 1$ is a quadratic *nonresidue* modulo n when

$$(y \,|\, p) = (y \,|\, q) = -1,$$

because $(y \,|\, pq) = (y \,|\, p)(y \,|\, q)$.

The Setup

- Bob publishes $n = pq$, a product of two distinct primes, and a quadratic nonresidue y with Jacobi symbol 1.
- Bob keeps secret the factorization of n.
- Alice wants to send bit string $b_1 b_2 \cdots b_k$ to Bob.
- Alice encrypts the bits by choosing a random quadratic residue modulo n if b_i is 1 and a random quadratic nonresidue (with Jacobi symbol 1) otherwise.
- So a sequence of residues and nonresidues are sent.
- Knowing the factorization of n, Bob can efficiently test quadratic residuacity and thus read the message.

The Protocol for Alice

for $i = 1, 2, \ldots, k$ **do**

  Pick $r \in Z_n^*$ randomly;

  if $b_i = 1$ **then**

    Send $r^2 \bmod n$; {Jacobi symbol is 1.}

  else

    Send $r^2 y \bmod n$; {Jacobi symbol is still 1.}

  end if

end for

The Protocol for Bob

for $i = 1, 2, \ldots, k$ **do**
    Receive r;
    if $(r \mid p) = 1$ and $(r \mid q) = 1$ **then**
        $b_i := 1;$
    else
        $b_i := 0;$
    end if
end for

Semantic Security

- This encryption scheme is probabilistic.
- There are a large number of different encryptions of a given message.
- One is chosen at random by the sender to represent the message.
  - Encryption is a *one-to-many* mapping.
- This scheme is both polynomially secure and **semantically secure**.

What then do you call proof?
— Henry James (1843–1916),
The Wings of the Dove (1902)

Leibniz knew what a proof is.
Descartes did not.
— Ian Hacking (1973)

What Is a Proof?

- A proof convinces a party of a certain claim.
  - "$x^n + y^n \neq z^n$ for all $x, y, z \in \mathbb{Z}^+$ and $n > 2$."
  - "Graph G is Hamiltonian."
  - "$x^p = x \bmod p$ for prime p and $p \not| x$."
- In mathematics, a proof is a fixed sequence of theorems.
  - Think of it as a written examination.
- We will extend a proof to cover a proof *process* by which the validity of the assertion is established.
  - Recall a job interview or an oral examination.

Prover and Verifier

- There are two parties to a proof.
  - The **prover** (**Peggy**).
  - The **verifier** (**Victor**).
- Given an assertion, the prover's goal is to convince the verifier of its validity (**completeness**).
- The verifier's objective is to accept only correct assertions (**soundness**).
- The verifier usually has an easier job than the prover.
- The setup is similar to the Turing test.[a]

[a] Turing (1950).

Interactive Proof Systems

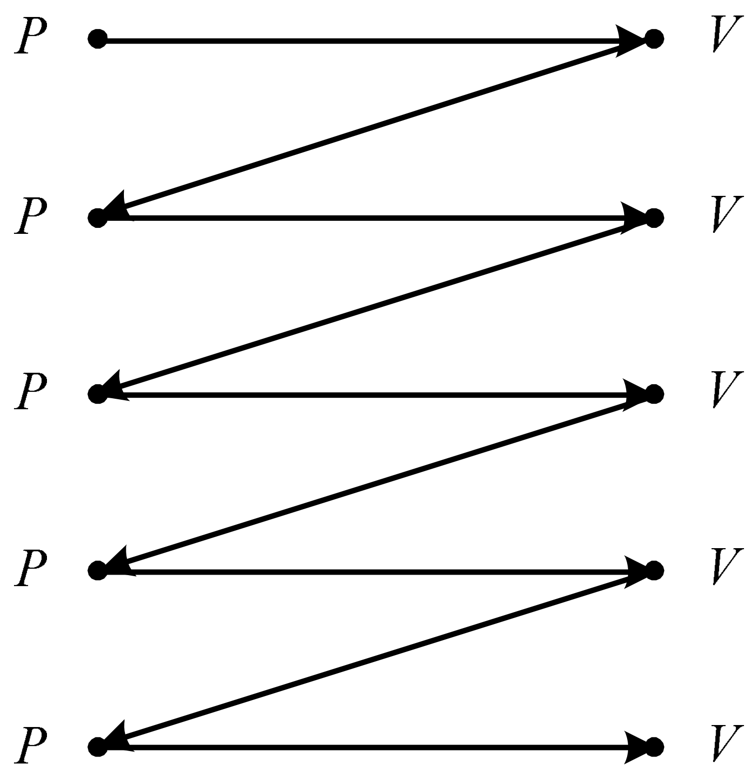
- An **interactive proof** for a language L is a sequence of questions and answers between the two parties.
- At the end of the interaction, the verifier decides whether the claim is true or false.
- The verifier must be a probabilistic polynomial-time algorithm.
- The prover runs an exponential-time algorithm.[a]
  - If the prover is not more powerful than the verifier, no interaction is needed!

[a] See the problem to Note 12.3.7 on p. 296 and Proposition 19.1 on p. 475, both of the textbook, about alternative complexity assumptions without affecting the definition. Contributed by Mr. Young-San Lin (B97902055) and Mr. Chao-Fu Yang (B97902052) on December 18, 2012.

Interactive Proof Systems (concluded)

- The system decides L if the following two conditions hold for any common input x.
  - If $x \in L$, then the probability that x is accepted by the verifier is at least $1 - 2^{-|x|}$.
  - If $x \notin L$, then the probability that x is accepted by the verifier with *any* prover replacing the original prover is at most $2^{-|x|}$.
- Neither the number of rounds nor the lengths of the messages can be more than a polynomial of $|x|$.

An Interactive Proof



IP ("Interactive Polynomial Time")[a]

- **IP** is the class of all languages decided by an interactive proof system.
- When $x \in L$, the completeness condition can be modified to require that the verifier accept with certainty without affecting IP.[b]
- Similar things cannot be said of the soundness condition when $x \notin L$.
- Verifier's coin flips can be public (called **Arthur-Merlin games**).[c]

---

[a] Goldwasser, Micali, & Rackoff (1985).
[b] Goldreich, Mansour, & Sipser (1987).
[c] Goldwasser & Sipser (1989).

The Relations of IP with Other Classes

- NP $\subseteq$ IP.
  - IP becomes NP when the verifier is deterministic and there is only one round of interaction.[a]
- BPP $\subseteq$ IP.
  - IP becomes BPP when the verifier ignores the prover's messages.
- IP = PSPACE.[b]

---

[a] Recall Proposition 41 on p. 344.
[b] Shamir (1990).

Graph Isomorphism

- $V_1 = V_2 = \{ 1, 2, \ldots, n \}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there exists a permutation π on $\{ 1, 2, \ldots, n \}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \cong G_2$.
- No known polynomial-time algorithms.[a]
- The problem is in NP (hence IP).
- It is not likely to be NP-complete.[b]

---

[a]The recent bound of Babai (2015) is $2^{O(\log^c n)}$ for some constant c.
[b]Schöning (1987).

GRAPH NONISOMORPHISM

- $V_1 = V_2 = \{ 1, 2, \ldots, n \}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **nonisomorphic** if there exist *no* permutations π on $\{ 1, 2, \ldots, n \}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \not\cong G_2$.
- Again, no known polynomial-time algorithms.
  - It is in coNP, but how about NP or BPP?
  - It is not likely to be coNP-complete.[a]
- Surprisingly, GRAPH NONISOMORPHISM $\in$ IP.[b]

---

[a] Schöning (1987).
[b] Goldreich, Micali, & Wigderson (1986).

A 2-Round Algorithm

Victor selects a random $i \in \{1, 2\}$;
Victor selects a random permutation π on $\{1, 2, \ldots, n\}$;
Victor applies π on graph G_i to obtain graph H;
Victor sends (G_1, H) to Peggy;
if $G_1 \cong H$ **then**
    Peggy sends $j = 1$ to Victor;
else
    Peggy sends $j = 2$ to Victor;
end if
if $j = i$ **then**
    Victor accepts; $\{G_1 \not\cong G_2.\}$
else
    Victor rejects; $\{G_1 \cong G_2.\}$
end if

Analysis

- Victor runs in probabilistic polynomial time.
- Suppose $G_1 \not\cong G_2$.
  - Peggy is able to tell which G_i is isomorphic to H, so $j = i$.
  - So Victor always accepts.
- Suppose $G_1 \cong G_2$.
  - No matter which i is picked by Victor, Peggy or any prover sees 2 *identical* copies.
  - Peggy or any prover with exponential power has only probability one half of guessing i correctly.
  - So Victor erroneously accepts with probability 1/2.
- Repeat the algorithm to obtain the desired probabilities.

Knowledge in Proofs

- Suppose I know a satisfying assignment to a satisfiable boolean expression.
- I can convince Alice of this by giving her the assignment.
- But then I give her more knowledge than is necessary.
  - Alice can claim that she found the assignment!
  - Login authentication faces essentially the same issue.
  - See `www.wired.com/wired/archive/1.05/atm_pr.html` for a famous ATM fraud in the U.S.

Knowledge in Proofs (concluded)

- Suppose I always give Alice random bits.
- Alice extracts no knowledge from me by any measure, but I prove nothing.
- Question 1: Can we design a protocol to convince Alice (the knowledge) of a secret without revealing anything extra?
- Question 2: How to define this idea rigorously?

Zero Knowledge Proofs[a]

An interactive proof protocol (P, V) for language L has the **perfect zero-knowledge** property if:

- For every verifier V', there is an algorithm M with expected polynomial running time.
- M on any input $x \in L$ generates the same probability distribution as the one that can be observed on the communication channel of (P, V') on input x.

[a] Goldwasser, Micali, & Rackoff (1985).

Comments

- Zero knowledge is a property of the prover.
  - It is the robustness of the prover against attempts of the verifier to extract knowledge via interaction.
  - The verifier may deviate arbitrarily (but in polynomial time) from the predetermined program.
  - A verifier cannot use the transcript of the interaction to convince a third-party of the validity of the claim.
  - The proof is hence not transferable.

Comments (continued)

- Whatever a verifier can "learn" from the specified prover P via the communication channel could as well be computed from the verifier alone.
- The verifier does not learn anything except "$x \in L$."
- Zero-knowledge proofs yield no knowledge in the sense that they can be constructed by the verifier who believes the statement, and yet these proofs do convince him.

Comments (continued)

- The "paradox" is resolved by noting that it is *not* the transcript of the conversation that convinces the verifier.
- But the fact that this conversation was held "on line."
- *Computational* zero-knowledge proofs are based on complexity assumptions.
  - M only needs to generate a distribution that is computationally indistinguishable from the verifier's view of the interaction.

Comments (concluded)

- If one-way functions exist, then zero-knowledge proofs exist for every problem in NP.[a]
- If one-way functions exist, then zero-knowledge proofs exist for every problem in PSPACE.[b]
- The verifier can be restricted to the honest one (i.e., it follows the protocol).[c]
- The coins can be public.[d]
- The digital money Zcash (2016) is based on zero-knowledge proofs.

---

[a] Goldreich, Micali, & Wigderson (1986).
[b] Ostrovsky & Wigderson (1993).
[c] Vadhan (2006).
[d] Vadhan (2006).

Quadratic Residuacity (QR)

- Let n be a product of two distinct primes.
- Assume extracting the square root of a quadratic residue modulo n is hard without knowing the factors.
- QR asks if $x \in Z_n^*$ is a quadratic residues modulo n.

A Useful Corollary of Lemma 82 (p. 701)

Corollary 83 *Let $n = pq$ be a product of two distinct primes. (1) If x and y are both quadratic residues modulo n, then $xy \in Z_n^*$ is a quadratic residue modulo n. (2) If x is a quadratic residue modulo n and y is a quadratic nonresidue modulo n, then $xy \in Z_n^*$ is a quadratic nonresidue modulo n.*

- Suppose x and y are both quadratic residues modulo n.
- Let $x \equiv a^2 \bmod n$ and $y \equiv b^2 \bmod n$.
- Now xy is a quadratic residue as $xy \equiv (ab)^2 \bmod n$.

The Proof (concluded)

- Suppose x is a quadratic residue modulo n and y is a quadratic nonresidue modulo n.
- By Lemma 82 (p. 701), $(x \,|\, p) = (x \,|\, q) = 1$ but, say, $(y \,|\, p) = -1$.
- Now xy is a quadratic nonresidue as $(xy \,|\, p) = -1$, again by Lemma 82 (p. 701).

Zero-Knowledge Proof of QR[a]

Below is a zero-knowledge proof for $x \in Z_n^*$ being a quadratic residue.

```
1: for m = 1, 2, ..., log_2 n do
2:     Peggy chooses a random v ∈ Z_n^* and sends
       y = v^2 mod n to Victor;
3:     Victor chooses a random bit i and sends it to Peggy;
4:     Peggy sends z = u^i v mod n, where u is a square root
       of x; {So u^2 ≡ x mod n.}
5:     Victor checks if z^2 ≡ x^i y mod n;
6: end for
7: Victor accepts x if Line 5 is confirmed every time;
```

[a]Goldwasser, Micali, & Rackoff (1985).

Analysis

- Suppose x is a quadratic residue.
  - Then x's square root u can be computed by Peggy.
  - Peggy can answer all challenges.
  - Now,
  $$z^2 \equiv \left(u^i\right)^2 v^2 \equiv \left(u^2\right)^i v^2 \equiv x^i y \bmod n.$$
  - So Victor will accept x.

Analysis (continued)

- Suppose x is a quadratic nonresidue.
  - Corollary 83 (p. 728) says if a is a quadratic residue, then xa is a quadratic nonresidue.
  - As y is a quadratic residue, $x^i y$ can be a quadratic residue (see Line 5) only when $i = 0$.
  - Peggy can answer only one of the two possible challenges, when $i = 0$.[a]
  - So Peggy will be caught in any given round with probability one half.

---

[a]Line 5 ($z^2 \equiv x^i y \bmod n$) cannot equate a quadratic residue z^2 with a quadratic nonresidue $x^i y$ when $i = 1$.

Analysis (continued)

- How about the claim of zero knowledge?
- The transcript between Peggy and Victor when x is a quadratic residue can be generated *without* Peggy!
- Here is how.
- Suppose x is a quadratic residue.[a]
- In each round of interaction with Peggy, the transcript is a triplet (y, i, z).
- We present an efficient Bob that generates (y, i, z) with the same probability *without* accessing Peggy's power.

---

[a]There is no zero-knowledge requirement when $x \notin L$.

Analysis (concluded)

Bob chooses a random $z \in Z_n^*$;

Bob chooses a random bit i;

Bob calculates $y = z^2 x^{-i} \bmod n$;[a]

Bob writes (y, i, z) into the transcript;

---

[a]Recall Line 5 on p. 730: Victor checks if $z^2 \equiv x^i y \bmod n$.

Comments

- Assume x is a quadratic residue.
- For (y, i, z), y is a random quadratic residue, i is a random bit, and z is a random number.
- Bob cheats because (y, i, z) is *not* generated in the same order as in the original transcript.
  - Bob picks Peggy's answer z first.
  - Bob then picks Victor's challenge i.
  - Bob finally patches the transcript.

Comments (concluded)

- So it is not the transcript that convinces Victor, but that *conversation with Peggy is held "on line."*
- The same holds even if the transcript was generated by a cheating Victor's interaction with (honest) Peggy.
- But we skip the details.[a]
- What if Victor always chooses $i = 1$ in the protocol, the harder case?[b]

---

[a] Or apply Vadhan (2006).

[b] Contributed by Mr. Chih-Duo Hong (`R95922079`) on December 13, 2006, Mr. Chin-Luei Chang (`D95922007`) on June 16, 2008, and Mr. Han-Ting Chen (`R10922073`) on December 30, 2021.

Zero-Knowledge Proof of 3 Colorability[a]

for $i = 1, 2, \ldots, |E|^2$ **do**
  Peggy chooses a random permutation π of the 3-coloring ϕ;
  Peggy samples encryption schemes randomly, commits[b] them, and sends $\pi(\phi(1)), \pi(\phi(2)), \ldots, \pi(\phi(|V|))$ *encrypted* to Victor;
  Victor chooses at random an edge $e \in E$ and sends it to Peggy for the coloring of the endpoints of e;
  if $e = (u, v) \in E$ **then**
    Peggy reveals the colors $\pi(\phi(u))$ and $\pi(\phi(v))$ and "proves" that they correspond to their encryptions;
  else
    Peggy stops;
  end if

---

[a] Goldreich, Micali, & Wigderson (1986).

[b] Contributed by Mr. Ren-Shuo Liu (`D98922016`) on December 22, 2009.

10: **if** the "proof" provided in Line 6 is not valid **then**

11: Victor rejects and stops;

12: **end if**

13: **if** $\pi(\phi(u)) = \pi(\phi(v))$ or $\pi(\phi(u)), \pi(\phi(v)) \notin \{1, 2, 3\}$ **then**

14: Victor rejects and stops;

15: **end if**

16: **end for**

17: Victor accepts;

Analysis

- If the graph is 3-colorable and both Peggy and Victor follow the protocol, then Victor always accepts.
- Suppose the graph is not 3-colorable and Victor follows the protocol.
- Let e be an edge that is *not* colored legally.
- Victor will pick it with probability $1/m$ per round, where $m = |E|$.
- Then however Peggy plays, Victor will reject with probability at least $1/m$ per round.

Analysis (concluded)

- So Victor will accept with probability at most

$$\left(1 - m^{-1}\right)^{m^2} \leq e^{-m}.$$

- Thus the protocol is a valid IP protocol.
- This protocol yields no knowledge to Victor as all he gets is a bunch of random pairs.
- The proof that the protocol is zero-knowledge to *any* verifier is intricate.[a]

---

[a]Or simply cite Vadhan (2006).

Comments

- Each $\pi(\phi(i))$ is encrypted by a different cryptosystem in Line 3.[a]
  - Otherwise, the coloring will be revealed in Line 6.
- Each edge e must be picked randomly.[b]
  - Otherwise, Peggy will know Victor's game plan and plot accordingly.

[a]Contributed by Ms. Yui-Huei Chang (`R96922060`) on May 22, 2008

[b]Contributed by Mr. Chang-Rong Hung (`R96922028`) on May 22, 2008

Approximability

All science is dominated by
the idea of approximation.
— Bertrand Russell (1872–1970)

Just because the problem is NP-complete
does not mean that
you should not try to solve it.
— Stephen Cook (2002)

Tackling Intractable Problems

- Many important problems are NP-complete or worse.
- **Heuristics** have been developed to attack them.
- They are **approximation algorithms**.
- How good are the approximations?
  - We are looking for theoretically *guaranteed* bounds, not "empirical" bounds.
- Are there NP problems that cannot be approximated *well* (assuming NP $\neq$ P)?
- Are there NP problems that cannot be approximated *at all* (assuming NP $\neq$ P)?

Some Definitions

- Given an **optimization problem**, each problem instance x has a set of **feasible solutions** $F(x)$.
- Each feasible solution $s \in F(x)$ has a cost $c(s) \in \mathbb{Z}^+$.
  - Here, cost refers to the quality of the feasible solution, not the time required to obtain it.
  - It is our **objective function**: total distance, number of satisfied clauses, cut size, etc.

Some Definitions (concluded)

- The **optimum cost** is

$$\text{OPT}(x) = \min_{s \in F(x)} c(s)$$

  for a minimization problem.

- It is

$$\text{OPT}(x) = \max_{s \in F(x)} c(s)$$

  for a maximization problem.

Approximation Algorithms

- Let (polynomial-time) algorithm M on x returns a feasible solution.
- M is an ϵ**-approximation algorithm**, where $\epsilon \geq 0$, if for all x,
$$\frac{|\, c(M(x)) - \text{OPT}(x) \,|}{\max(\text{OPT}(x), c(M(x)))} \leq \epsilon.$$
  - For a minimization problem,
$$\frac{c(M(x)) - \min_{s \in F(x)} c(s)}{c(M(x))} \leq \epsilon.$$
  - For a maximization problem,
$$\frac{\max_{s \in F(x)} c(s) - c(M(x))}{\max_{s \in F(x)} c(s)} \leq \epsilon. \tag{18}$$

Lower and Upper Bounds

- For a minimization problem,

$$\min_{s \in F(x)} c(s) \leq c(M(x)) \leq \frac{\min_{s \in F(x)} c(s)}{1 - \epsilon}.$$

- For a maximization problem,

$$(1 - \epsilon) \times \max_{s \in F(x)} c(s) \leq c(M(x)) \leq \max_{s \in F(x)} c(s). \tag{19}$$

Lower and Upper Bounds (concluded)

- ϵ ranges between 0 (best) and 1 (worst).
- For minimization problems, an ϵ-approximation algorithm returns solutions within

$$\left[\text{OPT}, \frac{\text{OPT}}{1-\epsilon} \right].$$

- For maximization problems, an ϵ-approximation algorithm returns solutions within

$$[\,(1-\epsilon) \times \text{OPT}, \text{OPT}\,].$$

Approximation Thresholds

- For each NP-complete optimization problem, we shall be interested in determining the *smallest* ϵ for which there is a polynomial-time ϵ-approximation algorithm.
- But sometimes ϵ has no minimum value.
- The **approximation threshold** is the greatest lower bound of all $\epsilon \geq 0$ such that there is a polynomial-time ϵ-approximation algorithm.
- By a standard theorem in real analysis, such a threshold exists.[a]

---

[a]Bauldry (2009).

Approximation Thresholds (concluded)

- The approximation threshold of an optimization problem is anywhere between 0 (approximation to any desired degree) and 1 (no approximation is possible).
- If P = NP, then all optimization problems *in NP* have an approximation threshold of 0.
- So assume P $\neq$ NP for the rest of the discussion.

Approximation Ratio

- ϵ-approximation algorithms can also be measured via the **approximation ratio**:[a]

$$\frac{c(M(x))}{\text{OPT}(x)}.$$

- For a minimization problem, the approximation ratio is

$$1 \leq \frac{c(M(x))}{\min_{s \in F(x)} c(s)} \leq \frac{1}{1-\epsilon}. \tag{20}$$

[a]Williamson & Shmoys (2011).

Approximation Ratio (concluded)

- For a maximization problem, the approximation ratio is[a]

$$1 - \epsilon \leq \frac{c(M(x))}{\max_{s \in F(x)} c(s)} \leq 1. \tag{21}$$

- Suppose there is an approximation algorithm that achieves an approximation ratio of θ.
  - For a minimization problem, it implies a $(1 - \theta^{-1})$-approximation algorithm by Eq. (20).
  - For a maximization problem, it implies a $(1 - \theta)$-approximation algorithm by Eq. (21).

---

[a] Some define the ratio as $1 \leq \frac{\max_{s \in F(x)} c(s)}{c(M(x))} \leq \frac{1}{1-\epsilon}$, symmetrical to inequalities (20).