BPP[a] (Bounded Probabilistic Polynomial)

- The class **BPP** contains all languages for which there is a precise polynomial-time NTM N such that:
  - If $x \in L$, then at least $3/4$ of the computation paths of N on x accept, and
  - If $x \notin L$, then at least $3/4$ of the computation paths of N on x reject.
- N accepts or rejects by a *clear* majority.

---

[a] Gill, 1977.

Magic $3/4$?

- The number $3/4$ bounds the probability of a right answer away from $1/2$.
- Any constant *strictly* between $1/2$ and 1 can be used without affecting the class BPP.
- In fact, any 0.5 plus inverse polynomial

$$0.5 + 1/p(n)$$

  between $1/2$ and 1 can be used.

The Majority Vote Algorithm

Suppose L is decided by N by majority $(1/2) + \epsilon$.

for $i = 1, 2, \ldots, 2k+1$ **do**
    Run N on input x;
end for
if "yes" is the majority answer **then**
    "yes";
else
    "no";
end if

Analysis

- The running time remains polynomial, being $2k + 1$ times N's running time.
- By Corollary 72 (p. 416), the probability of a false answer is at most $e^{-\epsilon^2 k}$.
- By taking $k = \lceil 2/\epsilon^2 \rceil$, the error probability is at most 1/4.
- As with the RP case, ϵ can be any inverse polynomial, because k remains polynomial in n.

Probability Amplification for BPP

- Let m be the number of random bits used by a BPP algorithm.
  - By definition, m is polynomial in n.
- With $k = \Theta(\log m)$ in the majority vote algorithm, we can lower the error probability to $\leq (3m)^{-1}$.

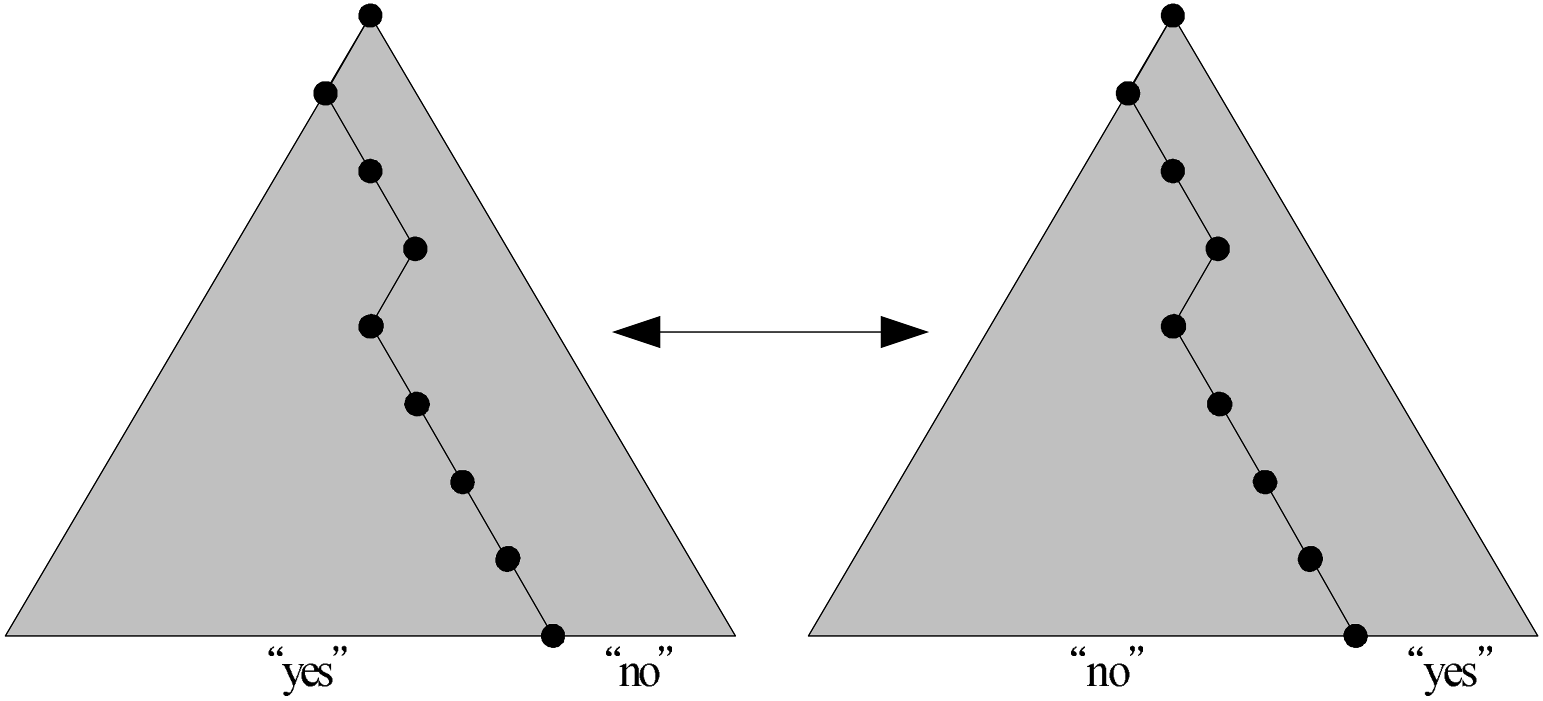
Aspects of BPP

- BPP is the most comprehensive yet plausible notion of efficient computation.
  - If a problem is in BPP, we take it to mean that the problem can be solved efficiently.
- $(\text{RP} \cup \text{coRP}) \subseteq (\text{NP} \cup \text{coNP})$.
- $(\text{RP} \cup \text{coRP}) \subseteq \text{BPP}$.
- Whether $\text{BPP} \subseteq (\text{NP} \cup \text{coNP})$ is unknown.
- But it is unlikely that $\text{NP} \subseteq \text{BPP}$ (p. 687).

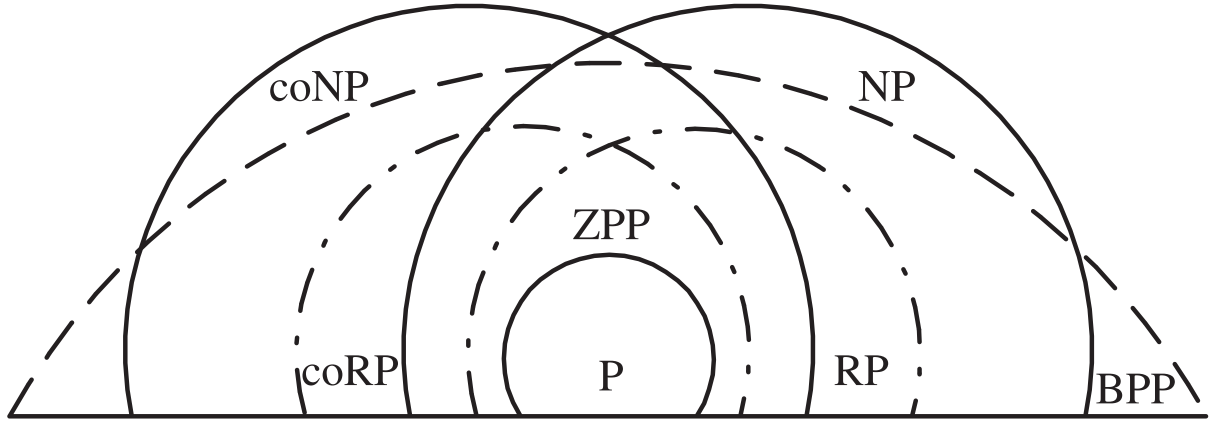
coBPP

- The definition of BPP is symmetric: acceptance by clear majority and rejection by clear majority.
- An algorithm for $L \in \text{BPP}$ becomes one for $\bar{L} \in \text{coBPP}$ by reversing the answer.
- Hence $\text{BPP} = \text{coBPP}$.
- This approach does not work for RP (it did not work for NP either).

BPP and coBPP



"The Good, the Bad, and the Ugly"



Circuit Complexity

- Circuit complexity is based on boolean circuits instead of Turing machines.
- A boolean circuit with n inputs computes a boolean function of n variables.
- By identify `true` with 1 and `false` with 0, a boolean circuit with n inputs accepts certain strings in $\{0, 1\}^n$.
- To relate circuits with arbitrary languages, we need one circuit for each possible input length n.

Formal Definitions

- The **size** of a circuit is the number of *gates* in it.
- A **family of circuits** is an infinite sequence $\mathcal{C} = (C_0, C_1, \ldots)$ of boolean circuits, where C_n has n boolean inputs.
- $L \subseteq \{0,1\}^*$ has **polynomial circuits** if there is a family of circuits $\mathcal{C}$ such that:
  - The size of C_n is at most $p(n)$ for some fixed polynomial p.
  - For input $x \in \{0,1\}^*$, $C_{|\,x\,|}$ outputs 1 if and only if $x \in L$.
    - C_n accepts $L \cap \{0,1\}^n$.

Exponential Circuits Contain All Languages

- Theorem 16 (p. 151) implies that there are languages that cannot be solved by circuits of size $2^n/(2n)$.
- But exponential circuits can solve all problems.

Proposition 73 *All decision problems (decidable or otherwise) can be solved by a circuit of size* 2^{n+2}.

- We will show that for any language $L \subseteq \{0,1\}^*$, $L \cap \{0,1\}^n$ can be decided by a circuit of size 2^{n+2}.

The Proof (concluded)

- Define boolean function $f : \{0,1\}^n \to \{0,1\}$, where

$$f(x_1 x_2 \cdots x_n) = \begin{cases} 1 & x_1 x_2 \cdots x_n \in L, \\ 0 & x_1 x_2 \cdots x_n \notin L. \end{cases}$$

- $f(x_1 x_2 \cdots x_n) = (x_1 \wedge f(1x_2 \cdots x_n)) \vee (\neg x_1 \wedge f(0x_2 \cdots x_n))$.
- The circuit size $s(n)$ for $f(x_1 x_2 \cdots x_n)$ hence satisfies

$$s(n) = 3 + 2s(n-1)$$

  with $s(1) = 1$.
- Solve it to obtain $s(n) = 2^{n+1} + 2^{n-1} - 4$.

The Circuit Complexity of P

Proposition 74 *All languages in P have polynomial circuits.*

- Let $L \in \mathrm{P}$ be decided by a TM in time $p(n)$.
- By Corollary 32 (p. 230), there is a circuit with $O(p(n)^2)$ gates that accepts $L \cap \{0,1\}^n$.
- The size of the circuit depends only on L and the length of the input.
- The size of the circuit is polynomial in n.

Languages That Polynomial Circuits Accept

- Do polynomial circuits accept only languages in P?
- There are *undecidable* languages that have polynomial circuits.
  - Let $L \subseteq \{0,1\}^*$ be an undecidable language.
  - Let $U = \{1^n : \text{the binary expansion of } n \text{ is in } L\}$.
  - U must be undecidable.
  - $U \cap \{1\}^n$ can be accepted by C_n that is trivially false if $1^n \notin U$ and trivially true if $1^n \in U$.
  - The family of circuits $(C_0, C_1, \ldots)$ is polynomial in size.

A Patch

- Despite their simplicity, the previous discussions imply the following:
  - Circuits are *not* a realistic model of computation.
  - Polynomial circuits are *not* a plausible notion of efficient computation.
- What gives?
- The *effective and efficient constructibility* of $C_0, C_1, \ldots$.

Uniformity

- A family $(C_0, C_1, \ldots)$ of circuits is **uniform** if there is a $\log n$-space bounded TM which on input 1^n outputs C_n.
  - Circuits now cannot accept undecidable languages (why?).
  - The circuit family on p. 431 is not constructible by a *single* Turing machine (algorithm).
- A language has **uniformly polynomial circuits** if there is a *uniform* family of polynomial circuits that decides it.

Uniformly Polynomial Circuits and P

Theorem 75 *$L \in P$ if and only if L has uniformly polynomial circuits.*

- One direction was proved in Proposition 74 (p. 430).
- Now suppose L has uniformly polynomial circuits.
- Decide $x \in L$ in polynomial time as follows:
  - Build $C_{|x|}$ in $\log |x|$ space, hence polynomial time.
  - Evaluate the circuit with input x in polynomial time.
- Therefore $L \in \mathrm{P}$.

Relation to P vs. NP

- Theorem 75 implies that P $\neq$ NP if and only if NP-complete problems have no *uniformly* polynomial circuits.
- A stronger conjecture: NP-complete problems have no polynomial circuits, *uniform or otherwise*.
- The above is currently the preferred approach to proving the P $\neq$ NP conjecture—without success so far.
  - Theorem 16 (p. 151) states that there are boolean functions requiring $2^n/(2n)$ gates to compute.
  - In fact, almost all boolean functions do.

BPP's Circuit Complexity

Theorem 76 (Adleman, 1978) *All languages in BPP have polynomial circuits.*

- Our proof will be *nonconstructive* in that only the existence of the desired circuits is shown.
  - Something exists if its probability of existence is nonzero.
- How to efficiently generate circuit C_n given 1^n is not known.
- In fact, if the construction of C_n is efficient, then $\text{P} = \text{BPP}$, a most unlikely result.

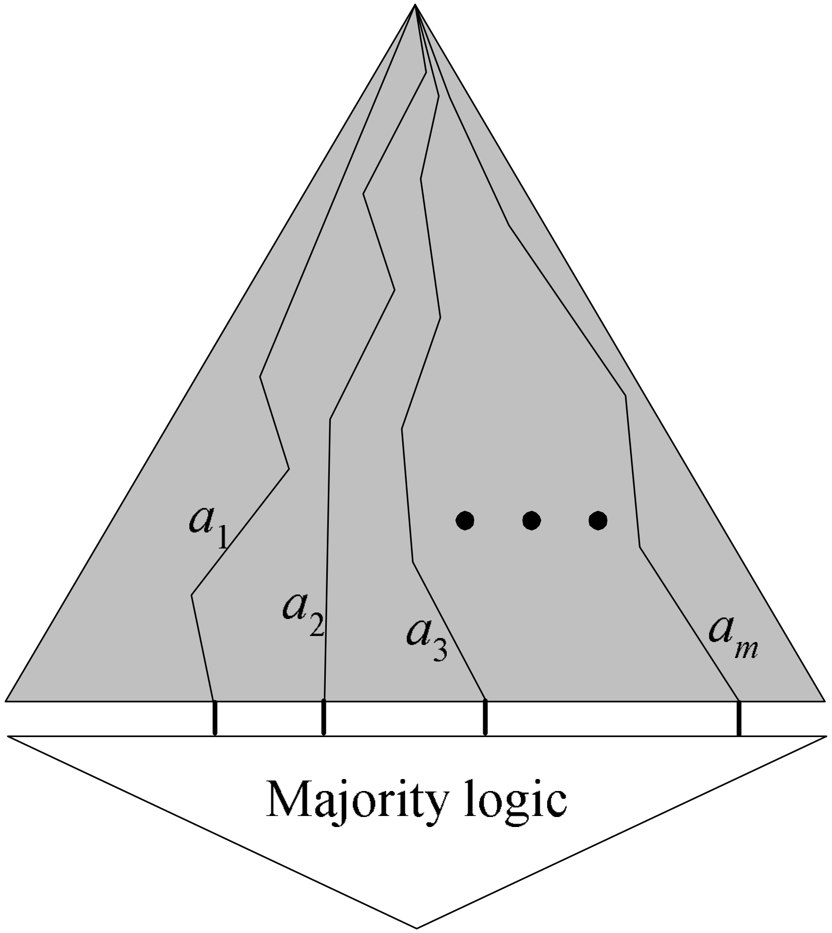
The Proof

- Let $L \in$ BPP be decided by a precise NTM N by clear majority.
- We shall prove that L has a polynomial family of circuits $C_0, C_1, \ldots$.
- Suppose N runs in time $p(n)$, where $p(n)$ is a polynomial.
- Let $A_n = \{a_1, a_2, \ldots, a_m\}$, where $a_i \in \{0,1\}^{p(n)}$ and $m = 12(n+1)$.
- Each $a_i \in A_n$ represents a sequence of nondeterministic choices—i.e., a computation path—for N.

The Proof (continued)

- Let x be an input with $|\,x\,| = n$.
- Circuit C_n simulates N on x with each sequence of choices in A_n and then takes the majority of the m outcomes.
- Because N with a_i is a polynomial-time TM, it can be simulated by polynomial circuits of size $O(p(n)^2)$.
  - See the proof of Proposition 74 (p. 430).
- The size of C_n is therefore $O([\,mp(n)\,]^2) = O(n^2 p(n)^2)$, a polynomial.
- We next prove the existence of A_n making C_n correct.

The Circuit



The Proof (continued)

- Call a_i **bad** if it leads N to a false positive or a false negative answer.
- Select A_n *uniformly randomly*.
- For each $x \in \{0,1\}^n$, at most $1/4$ of the computations of N are erroneous.
- Because the sequences in A_n are chosen randomly and independently, the expected number of bad a_i's is $m/4$.
- By the Chernoff bound (p. 412), the probability that the number of bad a_i's is $m/2$ or more is at most

$$e^{-m/12} < 2^{-(n+1)}.$$

The Proof (concluded)

- The error probability is $< 2^{-(n+1)}$ for each $x \in \{0,1\}^n$.
- The probability that there is an x such that A_n results in an incorrect answer is $< 2^n 2^{-(n+1)} = 2^{-1}$.
  - $\text{prob}[\, A \cup B \cup \cdots \,] \leq \text{prob}[\, A \,] + \text{prob}[\, B \,] + \cdots$.
- So with probability one half, a random A_n produces a correct C_n for *all* inputs of length n.
- Because this probability exceeds 0, an A_n that makes majority vote work for all inputs of length n exists.
- Hence a correct C_n exist.

Cryptography[a]

- **Alice** (A) wants to send a message to **Bob** (B) over a channel monitored by **Eve** (eavesdropper).
- The protocol should be such that the message is known only to Alice and Bob.
- The art and science of keeping messages secure is **cryptography**.

$$\text{Alice} \xrightarrow{\quad\text{Eve}\quad} \text{Bob}$$

---

[a] "Whoever wishes to keep a secret must hide the fact that he possesses one." — Johann Wolfgang von Goethe (1749–1832).

Encryption and Decryption

- Alice and Bob agree on two algorithms E and D—the **encryption** and the **decryption algorithms**.
- Both E and D are known to the public in the analysis.
- Alice runs E and wants to send a message x to Bob.
- Bob operates D.
- Privacy is assured in terms of two numbers e, d, the **encryption** and **decryption keys**.
- Alice sends $y = E(e, x)$ to Bob, who then performs $D(d, y) = x$ to recover x.
- x is called **plaintext**, and y is called **ciphertext**.

Some Requirements

- D should be an inverse of E given e and d.
- D and E must both run in (probabilistic) polynomial time.
- Eve should not be able to recover y from x without knowing d.
  - As D is public, d must be kept secret.
  - e may or may not be a secret.

Degrees of Security

- **Perfect secrecy**: After a ciphertext is intercepted by the enemy, the a posteriori probabilities of the plaintext that this ciphertext represents are identical to the a priori probabilities of the same plaintext before the interception.
- Such systems are said to be **informationally secure**.
- A system is **computationally secure** if breaking it is theoretically possible, just computationally infeasible.

The One-Time Pad[a]

1: Alice generates a random string r as long as x;

2: Alice sends r to Bob over a secret channel;

3: Alice sends $r \oplus x$ to Bob over a public channel;

4: Bob receives y;

5: Bob recovers $x := y \oplus r$;

---

[a]Mauborgne and Vernam, 1917, Shannon, 1949.

Analysis

- The one-time pad uses $e = d = r$.
- This is said to be a **private-key cryptosystem**.
- Knowing x and knowing r are equivalent.
- Because r is random and private, the one-time pad achieves perfect secrecy.
- The random bit string must be new for each round of communication.
  - **Cryptographically strong pseudorandom generators** require exchanging only the seed once.
- The assumption of a private channel is problematic.

Public-Key Cryptography[a]

- Suppose only d is private to Bob, whereas e is public knowledge.
- Bob generates the (e, d) pair and publishes e.
- Anybody like Alice can send $E(e, x)$ to Bob.
- Knowing d, Bob can recover x by $D(d, E(e, x)) = x$.
- The assumptions are complexity-theoretic.
  - It is computationally difficult to compute d from e.
  - It is computationally difficult to compute x from y without knowing d.

[a] Diffie and Hellman, 1976.

Complexity Issues

- Given y and x, it is easy to verify whether $E(e, x) = y$.
- A public-key cryptosystem in some sense is within NP.
- A necessary condition for the existence of secure public-key cryptosystems is P $\neq$ NP.
- But more is needed than P $\neq$ NP.
- For example, it is not sufficient that D is hard to compute in the worst case.
- We want it to be hard to compute in "most" cases.

One-Way Functions

- We say that f is a **one-way function** if:
  - f is one-to-one.
  - For all $x \in \Sigma^*$, $|\, x \,|^{1/k} \leq |f(x)| \leq |\, x \,|^k$ for some $k > 0$.
  - f can be computed in polynomial time.
  - f^{-1} cannot be computed in polynomial time.
    * Exhaustive search works, but it is too slow.
- Even if P $\neq$ NP, there is no guarantee that one-way functions exist.
- No functions have been proved to be one-way.
- Breaking a glass is a one-way function?

Candidates of One-Way Functions

- Modular exponentiation $f(x) = g^x \bmod p$.
  - **Discrete logarithm** is hard.
- The RSA[a] function $f(x) = x^e \bmod pq$ for an odd e relatively prime to $\phi(pq)$.
  - Breaking the RSA function is hard.
- Modular squaring $f(x) = x^2 \bmod pq$.
  - Determining whether a number with a Jacobi symbol 1 is a quadratic residue is hard—the **quadratic residuacity assumption** (**QRA**).

[a] Rivest, Shamir, and Adleman, 1978.

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 53 (p. 331),

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

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

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

which can be found by the Euclidean algorithm.

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 $y = x^e \bmod pq$.
- Knowing $\phi(pq)$, Bob calculates d such that $ed = 1 + k\phi(pq)$ for some $k \in \mathbb{Z}$.
  - 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. 338).

Implications of the "Security" of the RSA Function

- Factoring pq or calculating d from (e, pq) seems hard.
  - See also p. 335.
- It is known that breaking the last bit of RSA is as hard as breaking the RSA.
- Recall that problem A is harder than problem B if solving A results in solving B.
  - Factorization is "harder than" breaking the RSA.
  - Calculating Euler's phi function is "harder than" breaking the RSA.

Probabilistic Encryption[a]

- 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!
- What is required is a scheme that does not "leak" *partial* information.
- The first solution to the problems of skewed distribution and partial information was based on the QRA.

[a]Goldwasser and Micali, 1982.

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.
- To send bit string $b_1 b_2 \ldots b_k$ 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.
- A sequence of residues and nonresidues are sent.
- Knowing the factorization of n, Bob can efficiently test quadratic residuacity and thus read the message.

A Useful Lemma

Lemma 77 *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.
  - 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 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.
- This scheme is both polynomially secure and **semantically secure**.

A Probabilistic Encryption Based on RSA

The Protocol for Alice.

for $i = 1, 2, \ldots, k$ **do**
  Pick $r \in \{1, 2, \ldots, pq/2\}$ randomly;
  Send $(2r + b_i)^e \bmod pq$;
end for

The Protocol for Bob.

for $i = 1, 2, \ldots, k$ **do**
  Receive y;
  $b_i := (y^d \bmod pq) \bmod 2$;
end for

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

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

- Assume the cryptosystem satisfies the commutative property

$$E(e, D(d, x)) = D(d, E(e, x)). \tag{7}$$

  - As $(x^d)^e = (x^e)^d$, the RSA system satisfies it.
  - Every cryptosystem guarantees $D(d, E(e, x)) = x$.

[a]Diffie and 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. (7).

- The claim of authenticity is founded on the difficulty of inverting E_{Alice} without knowing the key d_{Alice}.

- Warning: If Alice signs anything presented to her, she might inadvertently decrypt a ciphertext of hers.

Mental Poker[a]

- Suppose Alice and Bob have agreed on 3 n-bit numbers $a < b < c$, the cards.
- They want to randomly choose one card each, so that:
  - Their cards are different.
  - All 6 pairs of distinct cards are equiprobable.
  - Alice's (Bob's) card is known to Alice (Bob) but not to Bob (Alice), until Alice (Bob) announces it.
  - The person with the highest card wins the game.
  - The outcome is indisputable.
- Assume Alice and Bob will not deviate from the protocol.

---

[a] Shamir, Rivest, Adleman, 1981.

The Setup

- Alice and Bob agree on a large prime p;
- Each has two *secret* keys $e_{\text{Alice}}, e_{\text{Bob}}, d_{\text{Alice}}, d_{\text{Bob}}$ such that $e_{\text{Alice}} d_{\text{Alice}} = e_{\text{Bob}} d_{\text{Bob}} = 1 \bmod (p-1)$;
  - This ensures that $(x^{e_{\text{Alice}}})^{d_{\text{Alice}}} = x \bmod p$ and $(x^{e_{\text{Bob}}})^{d_{\text{Bob}}} = x \bmod p$.
- The protocol lets Bob pick Alice's card and Alice pick Bob's card.
- Cryptographic techniques make it plausible that Alice's and Bob's choices are practically random, for lack of time to break the system.

The Protocol

1: Alice encrypts the cards

$$a^{e_{\text{Alice}}} \bmod p, b^{e_{\text{Alice}}} \bmod p, c^{e_{\text{Alice}}} \bmod p$$

and sends them in random order to Bob;

1: Bob picks one of the messages $x^{e_{\text{Alice}}}$ to send to Alice;

2: Alice decodes it $(x^{e_{\text{Alice}}})^{d_{\text{Alice}}} = x \bmod p$ for her card;

3: Bob encrypts the two remaining cards $(x^{e_{\text{Alice}}})^{e_{\text{Bob}}} \bmod p, (y^{e_{\text{Alice}}})^{e_{\text{Bob}}} \bmod p$ and sends them in random order to Alice;

4: Alice picks one of the messages, $(z^{e_{\text{Alice}}})^{e_{\text{Bob}}}$, encrypts it $((z^{e_{\text{Alice}}})^{e_{\text{Bob}}})^{d_{\text{Alice}}} \bmod p$, and sends it to Bob;

5: Bob decrypts the message $(((z^{e_{\text{Alice}}})^{e_{\text{Bob}}})^{d_{\text{Alice}}})^{d_{\text{Bob}}} = z \bmod p$ for his card;

What Is a Proof?

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

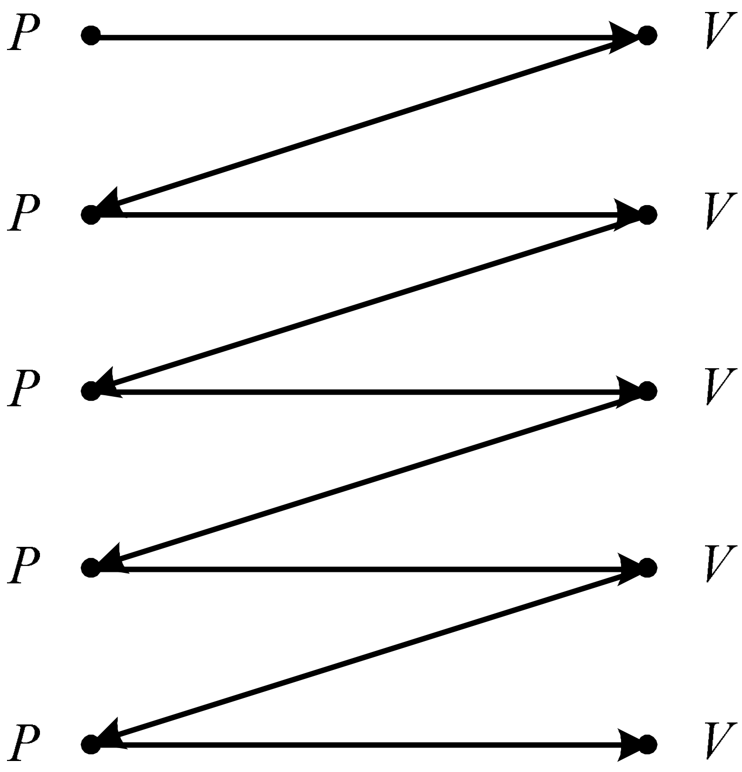
Interactive Proof Systems

- 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 based on the knowledge he acquired in the proof process whether the claim is true or false.
- The verifier must be a probabilistic polynomial-time algorithm.
- The prover runs an exponential-time algorithm.
  - If the prover is not more powerful than the verifier, no interaction is needed.

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[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 accepts with certainty without affecting IP.
- Similar things cannot be said of the soundness condition when $x \notin L$.

[a]Goldwasser, Micali, Rackoff, 1985.

The Relations of IP with Other Classes

- NP $\subseteq$ IP.
  - IP becomes NP when the verifier is deterministic.
- BPP $\subseteq$ IP.
  - IP becomes BPP when the verifier ignores the prover's messages.
- IP actually coincides with PSPACE (see pp. 728ff for a proof).[a]

[a] Shamir, 1990.

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 **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 $G_1 \not\cong G_2$ (**nonisomorphic**).
- No known polynomial-time algorithms.
  - It is in coNP, but how about NP or BPP?
  - The complementary problem $G_1 \cong G_2$ is in NP.
  - But it is not likely to be NP-complete.

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;
else
    Victor rejects;
end if

Analysis

- Victor runs in probabilistic polynomial time.
- Suppose the two graphs are not isomorphic.
  - Peggy is able to tell which G_i is isomorphic to H.
  - So Victor always accepts.
- Suppose the two graphs are isomorphic.
  - No matter which i is picked by Victor, Peggy or any prover sees 2 identical graphs.
  - 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 another person of this by giving him the assignment.
- But then I give him more knowledge than necessary.
  - For example, he might claim that he was the first to discover the assignment!
  - Login authentication is essentially the same problem.
- Can I convince him of the fact without revealing anything else?

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 a probabilistic 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.