The Circuit Complexity of P

**Proposition 69** *All languages in P have polynomial circuits.*

- Let $L \in P$ be decided by a TM in time $p(n)$.
- By Corollary 27 (p. 239), 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 $U = \{1^n : \text{the binary expansion of } n \text{ is in } L\}$.\(^a\)
  - $U$ is also undecidable.
  - $U \cap \{1\}^n$ can be accepted by $C_n$ that is trivially true if $1^n \in U$ and trivially false if $1^n \not\in U$.
  - The family of circuits $(C_0, C_1, \ldots)$ is polynomial in size.

\(^a\)Assume $n$’s leading bit is always 1 without loss of generality.
A Patch

• Despite the simplicity of a circuit, 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. 484 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 decide it.
Uniformly Polynomial Circuits and P

Theorem 70  \( L \in P \) if and only if \( L \) has uniformly polynomial circuits.

- One direction was proved in Proposition 69 (p. 483).
- Now suppose \( L \) has uniformly polynomial circuits.
- Decide \( x \in L \) in polynomial time as follows:
  - Let \( n = |x| \).
  - Build \( C_n \) in \( \log n \) space, hence polynomial time.
  - Evaluate the circuit with input \( x \) in polynomial time.
- Therefore \( L \in P \).
Relation to P vs. NP

- Theorem 70 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, uniformly or not.
- The above is currently the preferred approach to proving the $P \neq NP$ conjecture—without success so far.
  - Theorem 14 (p. 153) 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 71 (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.
- If the construction of $C_n$ is efficient, then $P = BPP$, an unlikely result.
The Proof

- Let $L \in \text{BPP}$ be decided by a precise NTM $N$ by clear majority.

- We shall prove that $L$ has polynomial 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 $a_i \in \{0, 1\}^{p(n)}$.

- Let $m = 12(n + 1)$.

- 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 69 (p. 483).

• The size of \( C_n \) is therefore \( O(mp(n)^2) = O(np(n)^2) \), a polynomial.

• We next prove the existence of \( A_n \) making \( C_n \) correct.
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 $x \in \{0, 1\}^n$, $1/4$ 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. 464), 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 $< 2^n 2^{-(n+1)} = 2^{-1}$.
  
  - $\Pr[A \cup B \cup \cdots] \leq \Pr[A] + \Pr[B] + \cdots$.

- 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 $C_n$ exists.
Whoever wishes to keep a secret must hide the fact that he possesses one.
— Johann Wolfgang von Goethe (1749–1832)
Cryptography

- 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.
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.\(^a\)

\(^a\)Both “zero” and “cipher” come from the same Arab word.
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 $x$ from $y$ 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 but computationally infeasible.
Conditions for Perfect Secrecy\textsuperscript{a}

- Consider a cryptosystem where:
  - The space of ciphertext is as large as that of keys.
  - Every plaintext has a nonzero probability of being used.

- It is perfectly secure if and only if the following hold.
  - A key is chosen with uniform distribution.
  - For each plaintext $x$ and ciphertext $y$, there exists a unique key $e$ such that $E(e, x) = y$.

\textsuperscript{a}Shannon (1949).
The One-Time Pad

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); allegedly used for the hotline between Russia and U.S.
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 (see also p. 501).

• 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\textsuperscript{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 $d$.

\textsuperscript{a}Diffie and Hellman (1976).
Complexity Issues

- Given $y$ and $x$, it is easy to verify whether $E(e, x) = y$.
- Hence one can always guess an $x$ and verify.
- Cracking a public-key cryptosystem is thus in NP.
- A necessary condition for the existence of secure public-key cryptosystems is $P \neq NP$.
- But more is needed than $P \neq NP$.
- It is not sufficient that $D$ is hard to compute in the worst case.
- It should be hard in “most” or “average” cases.
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 is too slow.

\(^a\)Diffie and Hellman (1976); Boppana and Lagarias (1986); Grollmann and Selman (1988); Ko (1985); Ko, Long, and Du (1986); Watanabe (1985); Young (1983).
Existence of One-Way Functions

• 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 a glass a one-way function?
Candidates of One-Way Functions

- Modular exponentiation \( f(x) = g^x \mod p \), where \( g \) is a primitive root of \( p \).
  - Discrete logarithm is hard.\(^a\)

- The RSA\(^b\) function \( f(x) = x^e \mod pq \) for an odd \( e \) relatively prime to \( \phi(pq) \).
  - Breaking the RSA function is hard.

- Modular squaring \( f(x) = x^2 \mod pq \).
  - Determining if a number with a Jacobi symbol 1 is a quadratic residue is hard—the quadratic residuacity assumption (QRA).

\(^a\)But it is in NP in some sense; Grollmann and Selman (1988).
\(^b\)Rivest, Shamir, and Adleman (1978).
The RSA Function

- Let $p, q$ be two distinct primes.
- The RSA function is $x^e \mod pq$ for an odd $e$ relatively prime to $\phi(pq)$.
  - By Lemma 49 (p. 359),
    
    $$\phi(pq) = pq \left(1 - \frac{1}{p}\right) \left(1 - \frac{1}{q}\right) = pq - p - q + 1.$$  
  
- As $\gcd(e, \phi(pq)) = 1$, there is a $d$ such that
  $$ed \equiv 1 \mod \phi(pq),$$
  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 $y = x^e \mod pq$.
- Knowing $\phi(pq)$, Bob calculates $d$ such that $ed = 1 + k\phi(pq)$ for some $k \in \mathbb{Z}$.
  - The decryption function is $y^d \mod pq$.
  - It works because $y^d = x^{ed} = x^{1+k\phi(pq)} = x \mod pq$ by the Fermat-Euler theorem when $\gcd(x, pq) = 1$ (p. 367).
The “Security” of the RSA Function

• Factoring \( pq \) or calculating \( d \) from \( (e, pq) \) seems hard.
  – See also p. 363.

• Breaking the last bit of RSA is as hard as breaking the RSA.\(^a\)

• Recommended RSA key sizes:
  – 1024 bits up to 2010.
  – 2048 bits up to 2030.
  – 3072 bits up to 2031 and beyond.

\(^a\)Alexi, Chor, Goldreich, and Schnorr (1988).
The “Security” of the RSA Function (concluded)

• 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.
  – Factorization is “harder than” calculating Euler’s phi function (see Lemma 49 on p. 359).

• Factorization cannot be NP-hard unless NP = coNP.\(^a\)

• So breaking the RSA is unlikely to imply P = NP.

\(^a\)Brassard (1979).
The Secret-Key Agreement Problem

- Exchanging messages securely using a private-key cryptosystem requires Alice and Bob possessing the same key (p. 503).
- 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.
The Diffie-Hellman Secret-Key Agreement Protocol

1: Alice and Bob agree on a large prime \( p \) and a primitive root \( g \) of \( p \); \( \{p \text{ and } g \text{ are public.}\} \)
2: Alice chooses a large number \( a \) at random;
3: Alice computes \( \alpha = g^a \mod p \);
4: Bob chooses a large number \( b \) at random;
5: Bob computes \( \beta = g^b \mod p \);
6: Alice sends \( \alpha \) to Bob, and Bob sends \( \beta \) to Alice;
7: Alice computes her key \( \beta^a \mod p \);
8: Bob computes his key \( \alpha^b \mod p \);
Analysis

- The keys computed by Alice and Bob are identical:

\[ \beta^a = g^{ba} = g^{ab} = \alpha^b \mod p. \]

- To compute the common key from \( p, g, \alpha, \beta \) is known as the **Diffie-Hellman problem**.

- It is conjectured to be hard.

- 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 Parallel History

- Diffie and Hellman’s solution to the secret-key agreement problem led to public-key cryptography.

- At around the same time (or earlier) in Britain, the RSA public-key cryptosystem was invented first before the Diffie-Hellman secret-key agreement scheme was.
  - Ellis, Cocks, and Williamson of the Communications Electronics Security Group of the British Government Communications Head Quarters (GCHQ).
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
  \[ e_{Alice}, e_{Bob}, d_{Alice}, d_{Bob}. \]
- Assume the cryptosystem satisfies the commutative property
  \[ E(e, D(d, x)) = D(d, E(e, x)). \quad (7) \]
  - 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 $E_{\text{Alice}}$ without knowing the key $d_{\text{Alice}}$.

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

\textsuperscript{a}Shamir, Rivest, and 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 $e_{\text{Alice}}d_{\text{Alice}} = e_{\text{Bob}}d_{\text{Bob}} = 1 \mod (p - 1)$;
  - This ensures that $(x^{e_{\text{Alice}}})^{d_{\text{Alice}}} = x \mod p$ and $(x^{e_{\text{Bob}}})^{d_{\text{Bob}}} = x \mod p$.

- 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

\[ a^{e_{Alice}} \mod p, b^{e_{Alice}} \mod p, c^{e_{Alice}} \mod p \]

and sends them in random order to Bob;

1: Bob picks one of the messages \( x^{e_{Alice}} \) to send to Alice;
2: Alice decodes it \( (x^{e_{Alice}})^{d_{Alice}} = x \mod p \) for her card;
3: Bob encrypts the two remaining cards
\[ (x^{e_{Alice}})^{e_{Bob}} \mod p, (y^{e_{Alice}})^{e_{Bob}} \mod p \]
and sends them in random order to Alice;
4: Alice picks one of the messages, \( (z^{e_{Alice}})^{e_{Bob}} \), encrypts it
\[ ((z^{e_{Alice}})^{e_{Bob}})^{d_{Alice}} \mod p \]
and sends it to Bob;
5: Bob decrypts the message
\[ (((z^{e_{Alice}})^{e_{Bob}})^{d_{Alice}})^{d_{Bob}} = z \mod p \] for his card;
Probabilistic Encryption\textsuperscript{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.

\textsuperscript{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 $b_1 b_2 \cdots b_k$ 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.
- 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 72  Let \( n = pq \) be a product of two distinct primes. Then a number \( y \in \mathbb{Z}_n^* \) is a quadratic residue modulo \( n \) if and only if \((y \mid p) = (y \mid q) = 1\).

- The “only if” part:
  - Let \( x \) be a solution to \( x^2 = y \mod pq \).
  - Then \( x^2 = y \mod p \) and \( x^2 = y \mod q \) also hold.
  - Hence \( y \) is a quadratic modulo \( p \) and a quadratic residue modulo \( q \).
The Proof (concluded)

• The “if” part:
  – Let $a_1^2 = y \mod p$ and $a_2^2 = y \mod q$.
  – Solve

    $$x = a_1 \mod p,$$
    $$x = a_2 \mod q,$$

    for $x$ with the Chinese remainder theorem.
  – As $x^2 = y \mod p$, $x^2 = y \mod q$, and $\gcd(p, q) = 1$,
    we must have $x^2 = y \mod pq$. 
The Protocol for Alice

1: for $i = 1, 2, \ldots, k$ do
2:  Pick $r \in Z_n^*$ randomly;
3:  if $b_i = 1$ then
4:    Send $r^2 \text{ mod } n$; \{Jacobi symbol is 1.\}
5:  else
6:    Send $r^2y \text{ mod } n$; \{Jacobi symbol is still 1.\}
7:  end if
8: end for
The Protocol for Bob

1: for $i = 1, 2, \ldots, k$ do
2:   Receive $r$;
3:   if $(r | p) = 1$ and $(r | q) = 1$ then
4:       $b_i := 1$;
5:   else
6:       $b_i := 0$;
7:   end if
8: 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.