

## 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.
- Now, identify **true**/1 with “yes” and **false**/0 with “no.”
- Then a boolean circuit with  $n$  inputs accepts certain strings in  $\{0, 1\}^n$ .
- To relate circuits with an arbitrary language, 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  $\mathcal{C} = (C_0, C_1, \dots)$  of boolean circuits, where  $C_n$  has  $n$  boolean inputs.
- For input  $x \in \{0, 1\}^*$ ,  $C_{|x|}$  outputs 1 if and only if  $x \in L$ .
- In other words,

$$C_n \text{ accepts } L \cap \{0, 1\}^n.$$

## Formal Definitions (concluded)

- $L \subseteq \{0, 1\}^*$  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  $p$ .
  - $C_n$  accepts  $L \cap \{0, 1\}^n$ .

## Exponential Circuits Suffice for All Languages

- Theorem 16 (p. 221) implies that there are languages that cannot be solved by circuits of size  $2^n / (2n)$ .
- But surprisingly, circuits of size  $2^{n+2}$  can solve *all* problems, decidable or otherwise!

## Exponential Circuits Suffice for All Languages (continued)

**Proposition 79** *All decision problems (decidable or otherwise) can be solved by a circuit of size  $2^{n+2}$  and depth  $2n$ .*

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

$$f(x_1x_2 \cdots x_n) = \begin{cases} 1, & x_1x_2 \cdots x_n \in L, \\ 0, & x_1x_2 \cdots x_n \notin L. \end{cases}$$

## The Proof (concluded)

- Clearly, any circuit that implements  $f$  decides  $L \cap \{0, 1\}^n$ .

- Now,

$$f(x_1x_2 \cdots x_n) = (x_1 \wedge f(1x_2 \cdots x_n)) \vee (\neg x_1 \wedge f(0x_2 \cdots x_n)).$$

- The circuit size  $s(n)$  for  $f(x_1x_2 \cdots x_n)$  hence satisfies

$$s(n) = 4 + 2s(n - 1)$$

with  $s(1) = 1$ .

- Solve it to obtain  $s(n) = 5 \times 2^{n-1} - 4 \leq 2^{n+2}$ .
- The longest path consists of an alternating sequence of  $\vee$ s and  $\wedge$ s.

## The Circuit Complexity of P

**Proposition 80** *All languages in P have polynomial circuits.*

- Let  $L \in P$  be decided by a TM in time  $p(n)$ .
- By Corollary 35 (p. 330), there is a circuit with  $O(p(n)^2)$  gates that accepts  $L \cap \{0, 1\}^n$ .
- The size of that circuit depends only on  $L$  and the length of the input.
- The size of that circuit is polynomial in  $n$ .

## Polynomial Circuits vs. P

- Is the converse of Proposition 80 true?
  - Do polynomial circuits accept only languages in P?
- No.
- Polynomial circuits can accept *undecidable* languages!<sup>a</sup>

---

<sup>a</sup>See p. 268 of the textbook.

## BPP's Circuit Complexity: Adleman's Theorem

**Theorem 81 (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.
  - Recall our proof of Theorem 16 (p. 221).
  - Something exists if its probability of existence is nonzero.
- It is not known how to efficiently generate circuit  $C_n$ .
  - If the construction of  $C_n$  can be made efficient, then  $P = BPP$ , an unlikely result.

## The Proof

- Let  $L \in \text{BPP}$  be decided by a precise polynomial-time NTM  $N$  by clear majority.
- We shall prove that  $L$  has polynomial circuits  $C_0, C_1, \dots$ 
  - These *deterministic* circuits do not err.
- Suppose  $N$  runs in time  $p(n)$ , where  $p(n)$  is a polynomial.
- Let  $A_n = \{ a_1, a_2, \dots, a_m \}$ , where  $a_i \in \{ 0, 1 \}^{p(n)}$ .
- Each  $a_i \in A_n$  represents a sequence of nondeterministic choices (i.e., a computation path) for  $N$ .
- Pick  $m = 12(n + 1)$ .

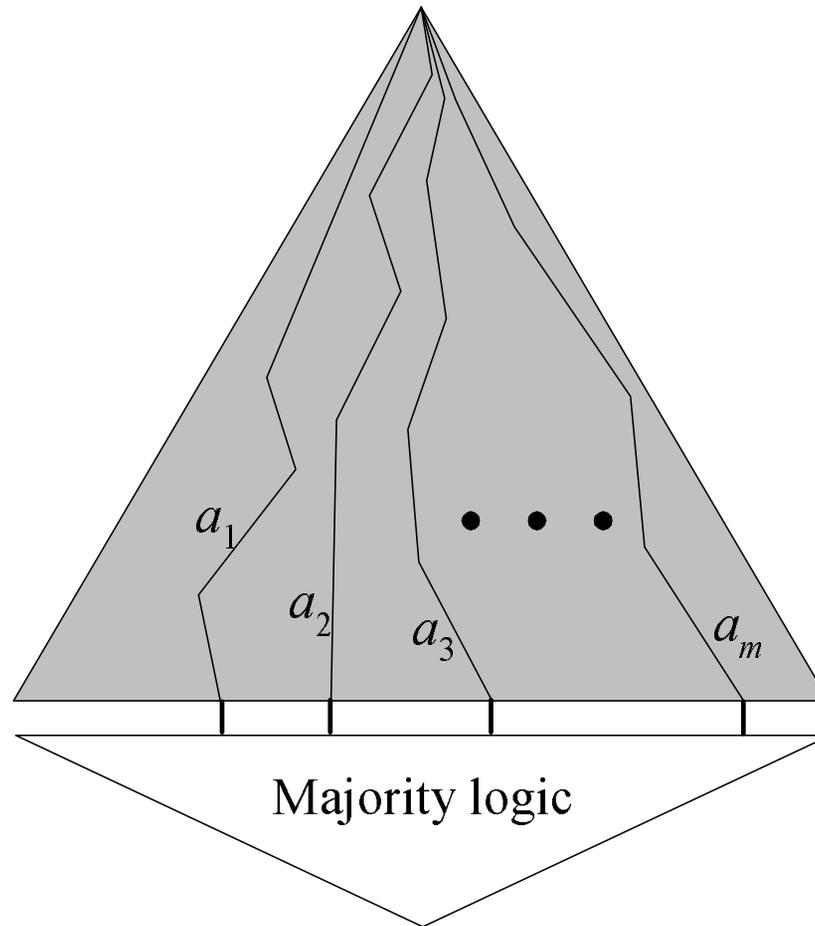
## The Proof (continued)

- Let  $x$  be an input with  $|x| = n$ .
- Circuit  $C_n$  simulates  $N$  on  $x$  with all sequences of choices in  $A_n$  and then takes the majority of the  $m$  outcomes.<sup>a</sup>
  - Note that each  $A_n$  yields a circuit.
- As  $N$  with  $a_i$  is a polynomial-time deterministic TM, it can be simulated by polynomial circuits of size  $O(p(n)^2)$ .
  - See the proof of Proposition 80 (p. 640).

---

<sup>a</sup>As  $m$  is even, there may be no clear majority. Still, the probability of that happening is very small and does not materially affect our general conclusion. Thanks to a lively class discussion on December 14, 2010.

# The Circuit



## The Proof (continued)

- The size of  $C_n$  is therefore  $O(mp(n)^2) = O(np(n)^2)$ .
  - This is a polynomial.
- We now confirm the existence of an  $A_n$  making  $C_n$  correct on *all*  $n$ -bit inputs.
- Call  $a_i$  **bad** if it leads  $N$  to an error (a false positive or a false negative) for  $x$ .
- Select  $A_n$  uniformly randomly.

## The Proof (continued)

- 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  $m/4$ .<sup>a</sup>
- Also note after fixing the input  $x$ , the circuit is a function of the random bits.

---

<sup>a</sup>So the proof will not work for NP. Contributed by Mr. Ching-Hua Yu (D00921025) on December 11, 2012.

## The Proof (continued)

- By the Chernoff bound (p. 619), 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 error probability of using the majority rule is thus

$$< 2^{-(n+1)}$$

for each  $x \in \{0, 1\}^n$ .

## The Proof (continued)

- The probability that there is an  $x$  such that  $A_n$  results in an incorrect answer is

$$< 2^n 2^{-(n+1)} = 2^{-1}$$

by the union bound (**Boole's inequality**).<sup>a</sup>

- We just showed that at least half of the random  $A_n$  are correct.
- So with probability  $\geq 0.5$ , a random  $A_n$  produces a correct  $C_n$  for *all* inputs of length  $n$ .
  - Of course, verifying this fact may take a long time.

---

<sup>a</sup>That is,  $\text{prob}[A \cup B \cup \dots] \leq \text{prob}[A] + \text{prob}[B] + \dots$ .

## The Proof (concluded)

- 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.<sup>a</sup>
- We have used the **probabilistic method**<sup>b</sup> popularized by Erdős (1947).<sup>c</sup>
- This result answers the question on p. 550 with a “yes.”

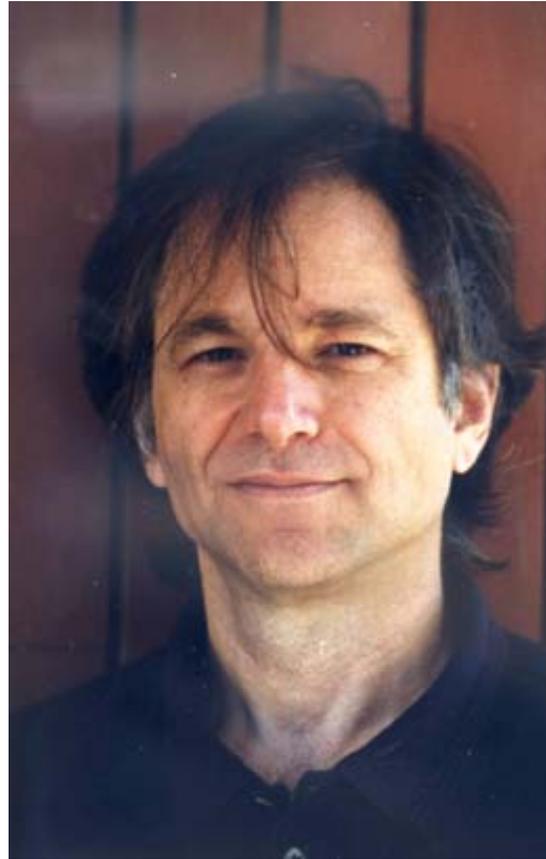
---

<sup>a</sup>Quine (1948), “To be is to be the value of a bound variable.”

<sup>b</sup>A counting argument in the probabilistic language.

<sup>c</sup>Szele (1943) and Turán (1934) were earlier.

## Leonard Adleman<sup>a</sup> (1945–)



---

<sup>a</sup>Turing Award (2002).

Paul Erdős (1913–1996)



# *Cryptography*

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  $D$ .

## Encryption and Decryption (concluded)

- 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**.<sup>a</sup>

---

<sup>a</sup>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.

## Degree 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.
  - The probability that plaintext  $\mathcal{P}$  occurs is independent of the ciphertext  $\mathcal{C}$  being observed.
  - So knowing  $\mathcal{C}$  yields no advantage in recovering  $\mathcal{P}$ .

## Degree of Security (concluded)

- 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<sup>a</sup>

- 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  $E(e, x) = y$ .

---

<sup>a</sup>Shannon (1949).

## The One-Time Pad<sup>a</sup>

- 1: Alice generates a random string  $r$  as long as  $x$ ;
- 2: Alice sends  $r$  to Bob over a secret channel;
- 3: Alice sends  $x \oplus r$  to Bob over a public channel;
- 4: Bob receives  $y$ ;
- 5: Bob recovers  $x := y \oplus r$ ;

---

<sup>a</sup>Mauborgne & Vernam (1917); Shannon (1949). It was allegedly used for the hotline between Russia and the 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.<sup>a</sup>
- The random bit string must be new for each round of communication.
- But the assumption of a private channel is problematic.

---

<sup>a</sup>See p. 661.

## Public-Key Cryptography<sup>a</sup>

- 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  $x$  via

$$D(d, E(e, x)) = x.$$

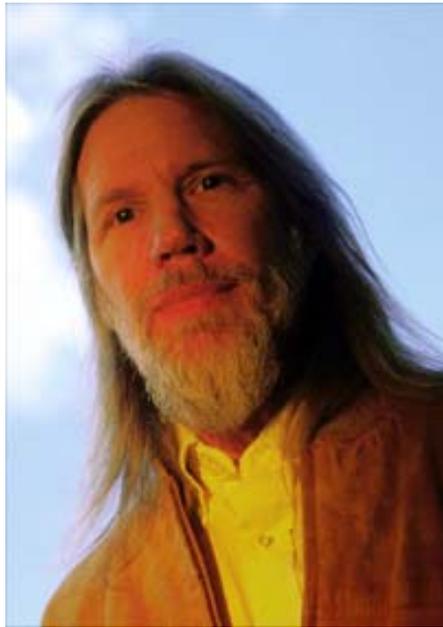
---

<sup>a</sup>Diffie & Hellman (1976).

## Public-Key Cryptography (concluded)

- 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  $d$ .

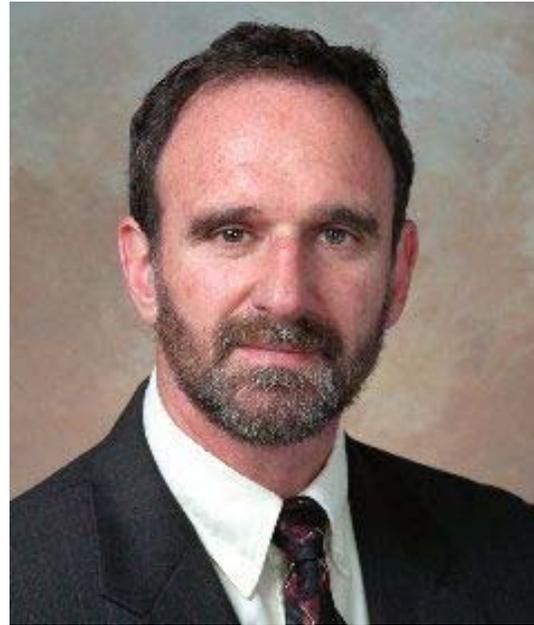
## Whitfield Diffie<sup>a</sup> (1944–)



---

<sup>a</sup>Turing Award (2016).

## Martin Hellman<sup>a</sup> (1945–)



---

<sup>a</sup>Turing Award (2016).

## 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  $P \neq NP$ .
- For instance, 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.<sup>a</sup>

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.

---

<sup>a</sup>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.<sup>a</sup>
- The RSA<sup>b</sup> function  $f(x) = x^e \bmod pq$  for an odd  $e$  relatively prime to  $\phi(pq)$ .
  - Breaking the RSA function is hard.

---

<sup>a</sup>Conjectured to be  $2^{n^\epsilon}$  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).

<sup>b</sup>Rivest, Shamir, & Adleman (1978).

## Candidates of One-Way Functions (concluded)

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

---

<sup>a</sup>Due to Gauss.

<sup>b</sup>Rabin (1979).

## The Secret-Key Agreement Problem

- Exchanging messages securely using a private-key cryptosystem requires Alice and Bob have the *same* key.<sup>a</sup>
  - An example is the  $r$  in the one-time pad.<sup>b</sup>
- 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.

---

<sup>a</sup>See p. 663.

<sup>b</sup>See p. 662.

## 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  $\alpha$  to Bob, and Bob sends  $\beta$  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

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

- To compute the common key from  $p, g, \alpha, \beta$  is known as the **Diffie-Hellman problem**.
- It is conjectured to be hard.<sup>a</sup>
- 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.

---

<sup>a</sup>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. 502),

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

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

$$ed \equiv 1 \pmod{\phi(pq)},$$

which can be found by the Euclidean algorithm.<sup>a</sup>

---

<sup>a</sup>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

$$y = x^e \pmod{pq}.$$

- 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. 507).

## A Public-Key Cryptosystem Based on RSA (continued)

- What if  $x$  is not relatively prime to  $pq$ ?<sup>a</sup>
- As  $\phi(pq) = (p - 1)(q - 1)$ ,

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

- Say  $x \equiv 0 \pmod{p}$ .
- Then

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

---

<sup>a</sup>Of course, one would be unlucky here.

## A Public-Key Cryptosystem Based on RSA (continued)

- On the other hand, either  $x \not\equiv 0 \pmod{q}$  or  $x \equiv 0 \pmod{q}$ .
- If  $x \not\equiv 0 \pmod{q}$ , then

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

by Fermat's "little" theorem (p. 505).

- If  $x \equiv 0 \pmod{q}$ , then

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

## A Public-Key Cryptosystem Based on RSA (concluded)

- By the Chinese remainder theorem (p. 504),

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

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.<sup>a</sup>
- Recommended RSA key sizes:<sup>b</sup>
  - 1024 bits up to 2010.
  - 2048 bits up to 2030.
  - 3072 bits up to 2031 and beyond.

---

<sup>a</sup>Alexi, Chor, Goldreich, & Schnorr (1988).

<sup>b</sup>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<sup>a</sup> is “harder than” breaking the RSA.
  - Factorization is “harder than” calculating Euler’s phi function (see Lemma 59 on p. 502).
  - So factorization is harder than calculating Euler’s phi function, which is harder than breaking the RSA.

---

<sup>a</sup>When the input is not factorized!

## The “Security” of the RSA Function (concluded)

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

---

<sup>a</sup>Brassard (1979).

<sup>b</sup>Shor (1994).

<sup>c</sup>Menn (2013).

## Adi Shamir, Ron Rivest, and Leonard Adleman



## Ron Rivest<sup>a</sup> (1947–)



---

<sup>a</sup>Turing Award (2002).

## Adi Shamir<sup>a</sup> (1952–)



---

<sup>a</sup>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.<sup>a</sup>

---

<sup>a</sup>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<sup>a</sup>

- 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

$$E(e, D(d, x)) = D(d, E(e, x)). \quad (17)$$

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

---

<sup>a</sup>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  $E_{\text{Alice}}$  without knowing the key  $d_{\text{Alice}}$ .

## Blind Signatures<sup>a</sup>

- 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.

---

<sup>a</sup>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;  $\{x$  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  $s = s' r^{-1} \pmod n$ ;

- This works because

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

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:

$$s^e \equiv x \pmod{n}.$$

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

## Probabilistic Encryption<sup>a</sup>

- 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.

---

<sup>a</sup>Goldwasser & Micali (1982). This paper “laid the framework for modern cryptography” (2013).

## Shafi Goldwasser<sup>a</sup> (1958–)



---

<sup>a</sup>Turing Award (2013).

Silvio Micali<sup>a</sup> (1954–)



---

<sup>a</sup>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  $x^2 = y \pmod{pq}$ .
  - Then  $x^2 = y \pmod{p}$  and  $x^2 = y \pmod{q}$  also hold.
  - Hence  $y$  is a quadratic modulo  $p$  and a quadratic residue modulo  $q$ .

## The Proof (concluded)

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

$$x = a_1 \pmod p,$$

$$x = a_2 \pmod q,$$

for  $x$  with the Chinese remainder theorem (p. 504).

- As  $x^2 = y \pmod p$ ,  $x^2 = y \pmod q$ , and  $\gcd(p, q) = 1$ , we must have  $x^2 = y \pmod{pq}$ .

## The Jacobi Symbol and Quadratic Residuacity Test

- The Legendre symbol can be used as a test for quadratic residuacity by Lemma 69 (p. 574).
- 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  $(y | pq) = (y | p)(y | q)$ .