Conditions for Perfect Secrecy[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 $E(e, x) = y$.

[a]Shannon (1949).

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); 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. 441).
- The random bit string must be new for each round of communication.
- 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 therefore $\mathrm{P} \neq \mathrm{NP}$.
- But more is needed than $\mathrm{P} \neq \mathrm{NP}$.
- 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" or "average" 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 $\mathrm{P} \neq \mathrm{NP}$, 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 $f(x) = g^x \bmod p$, 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.
- 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] 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 $x^e \bmod pq$ for an odd e relatively prime to $\phi(pq)$.
  - By Lemma 50 (p. 335),

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

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

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

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. 342).

The "Security" of the RSA Function

- Factoring pq or calculating d from (e, pq) seems hard.
  - See also p. 339.
- Breaking the last bit of RSA is as hard as breaking the RSA.[a]
- 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.
- Recommended RSA key sizes: 1024 bits up to 2010, 2048 bits up to 2030, and 3072 bits up to 2031 and beyond.

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

The Secret-Key Agreement Problem

- Exchanging messages securely using a private-key cryptosystem requires Alice and Bob possessing the same key (see p. 443).
- 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 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:

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

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

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).

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 $b_1 b_2 \cdots b_k$ 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 71 *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 $(r \,|\, p) = 1$ and $(r \,|\, q) = 1$ **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**.

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

$$E(e, D(d, x)) = D(d, E(e, x)). \tag{6}$$

  - 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. (6).
- The claim of authenticity is founded on the difficulty of inverting E_{Alice} without knowing the key d_{Alice}.

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 $p \nmid x$?"
- 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.
- The setup is very much like the Turing test.[a]

[a] Turing (1950).

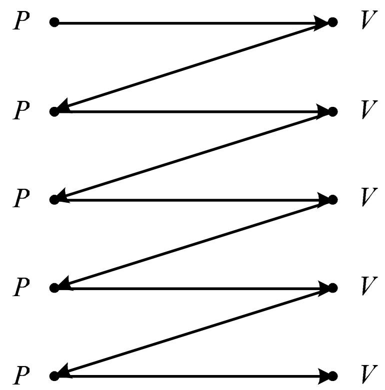
Interactive Proof Systems

- 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 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.[b]
- Similar things cannot be said of the soundness condition when $x \notin L$.
- Verifier's coin flips can be public.[c]

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

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.[a]

[a]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 $G_1 \cong G_2$ (**isomorphic**).
- No known polynomial-time algorithms.
- The problem is in NP (hence IP).
- But it is not likely to be NP-complete.[a]

[a]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 $G_1 \ncong G_2$ (**nonisomorphic**).
- Again, no known polynomial-time algorithms.
  - It is in coNP, but how about NP or BPP?
  - It is not likely to be coNP-complete.
- Surprisingly, GRAPH NONISOMORPHISM $\in$ IP.[a]

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

A 2-Round Algorithm

```
Victor selects a random i ∈ {1, 2};
Victor selects a random permutation π on {1, 2, ..., n};
Victor applies π on graph G_i to obtain graph H;
Victor sends (G_1, H) to Peggy;
if G_1 ≅ 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 Alice of this by giving her the assignment.
- But then I give her more knowledge than 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)

- Digital signatures authenticate *documents* but not *individuals*.
- They hence do not solve the problem.
- Suppose I always give Alice random bits.
- Alice's extracts no knowledge from me by any measure, but I prove nothing.
- Question 1: Can we design a protocol to convince Alice of (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, and 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 "$x \in L$."
- For all practical purposes "whatever" can be done after interacting with a zero-knowledge prover can be done by just believing that the claim is indeed valid.
- 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 (concluded)

- 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."
- There is no zero-knowledge requirement when $x \notin L$.
- *Computational* zero-knowledge proofs are based on complexity assumptions.
- It is known that if one-way functions exist, then zero-knowledge proofs exist for every problem in NP.[a]

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

Will You Be Convinced?

- A newspaper commercial for hair-growing products for men.
  - A (for all practical purposes) bald man has a full head of hair after 3 months.
- A TV commercial for weight-loss products.
  - A (by any reasonable measure) overweight woman loses 10 kilograms in 10 weeks.

Zero-Knowledge Proof of Quadratic Residuosity

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

Analysis

- Assume extracting the square root of a quadratic residue modulo a product of two primes is hard without knowing the factors.
- Suppose x is a quadratic nonresidue.
  - Peggy can answer only one of the two possible challenges.
    - Reason: a is a quadratic residue if and only if xa is a quadratic nonresidue.
  - So Peggy will be caught in any given round with probability one half.

Analysis (continued)

- Suppose x is a quadratic residue.
  - Peggy can answer all challenges.
  - So Victor will accept x.
- How about the claim of zero knowledge?
- The transcript between Peggy and Victor when x is a quadratic residue can be generated without Peggy!
  - So interaction with Peggy is useless.

Analysis (continued)

- Here is how.
- Suppose x is a quadratic residue.
- In each round of interaction with Peggy, the transcript is a triplet (y, i, z).
- We present an efficient algorithm Bob that generates (y, i, z) with the same probability *without* accessing Peggy.

Analysis (concluded)

Bob chooses a random $z \in Z_n^*$;
Bob chooses a random bit i;
Bob calculates $y = z^2 x^{-i} \bmod n$;
Bob writes (y, i, z) into the transcript;

Comments

- Bob cheats because (y, i, z) is *not* generated in the same order as in the original transcript.
  - Bob picks Victor's challenge first.
  - Bob then picks Peggy's answer.
  - Bob finally patches the transcript.
  - 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.

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 an encryption scheme randomly 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 coloring of u and v and "proves" that they correspond to their encryption;
  else
    Peggy stops;
  end if

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

```
10:    if the "proof" provided in Line 6 is not valid then
11:       Victor rejects and stops;
12:    end if
13:    if π(φ(u)) = π(φ(v)) or π(φ(u)), π(φ(v)) ∉ {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.
- If the graph is not 3-colorable and Victor follows the protocol, then however Peggy plays, Victor will accept with probability $\leq (1 - m^{-1})^{m^2} \leq e^{-m}$, where $m = |E|$.
- Thus the protocol is valid.
- 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 more intricate.