T-79.5502 Advanced Course in Cryptology

Lecture 1, November 1, 2007

Introduction to cryptographic security problems

- Coin flipping
- Oblivious transfer: Rabin OT, 1-out-of-2 OT

Textbook: Section 1

Coin flipping over telephone

We start with an informal definition:

Property 1.1: Let A and B be sets. We call a function $f\colon \mathrm{A}\to\mathrm{B}$ *magic* if it satisfies the following two conditions:

(I) For every $x\in\mathrm{A}$, it is easy to compute $f(x)$, while given any value $y \in f(\mathrm{A}) \subset \mathrm{B}$ it is impossible to find any information of any $x\in\mathrm{A}$ such that $y = f(x)$.

(II) It is impossible to find $x_1 \in \mathrm{A}$ and $x_2 \in \mathrm{A}$ such that $f(x_1) = f(x_2)$.

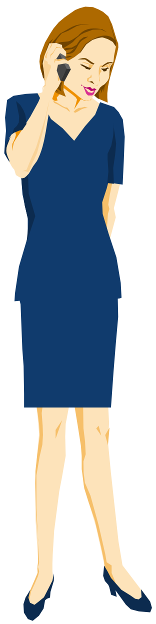
Protocol premises

Alice and Bob have agreed on

1. a large set A of integers
2. a small set B of integers
3. A magic function f: A$\to$B (in the sense of Property 1.1)
4. An even number x $\in$ A represents HEADS and an odd number x $\in$ A represents TAILS

Protocol

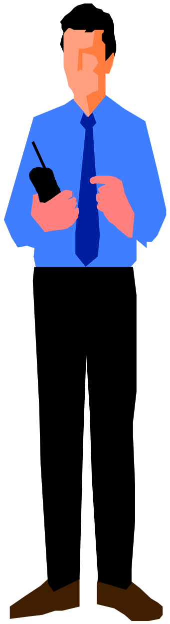
ALICE



1. Alice picks $x \in A$ and computes $f(x)$; she reads $f(x)$ to Bob over the phone

3. Alice receives Bob's guess. Then Alice reads x to Bob (or sends it to Bob over Internet)

BOB



2. Bob writes down the value a given by Alice. Then Bob guesses HEADS or TAILS and tells Alice his guess.

4. Bob receives x and computes $f(x)$ and verifies if $a = f(x)$. If yes, Bob checks if x is even or odd.

Discussion

In Property 1.1:

- What is easy, what is impossible?
- How to quantify degree of difficulty?

Security requirements:

- Alice cannot cheat. Bob has equal chances to get his guess right and Alice cannot change his chances during the protocol.
- Bob cannot cheat.
- A third party cannot cheat. How could a third party cheat?
- What can a party achieve by cheating?

Rudimentary security analysis

Alice can cheat if:

- She can find an even $x_1 \in \mathrm{A}$ and an odd $x_2 \in \mathrm{A}$ such that $f(x_1) = f(x_2)$.
- Impossible by Property II of the magic f.

Bob can cheat if:

- Given $a = f(x) \in \mathrm{B}$ Bob can tell if x is even or odd.
- Impossible by Property I of the magic f.

Security model and assumptions

Model

- What are the parties?
- Are the communications protected or not?
- Definition of the attacks (e.g. what it means that Bob can cheat)
- Security requirements (what the protocol wants to achieve, e.g, Bob has 50% chance to get his guess correct)

Assumptions

- Assumptions about the cryptographic primitives
- Other security assumptions

Explicitness

- Be explicit about all assumptions needed
  - Do we assume the selection of x by Alice be uniform?
- Be explicit about exact security services to be offered
  - Coin flipping over telephone offers commitments, but no confidentiality, authentication or proof-of-knowledge
- Be explicit about special cases in mathematics
  - $N = pq$ such that factoring of N is hard. If $p \approx q$, factoring of N is not hard.

Exercise 1.2

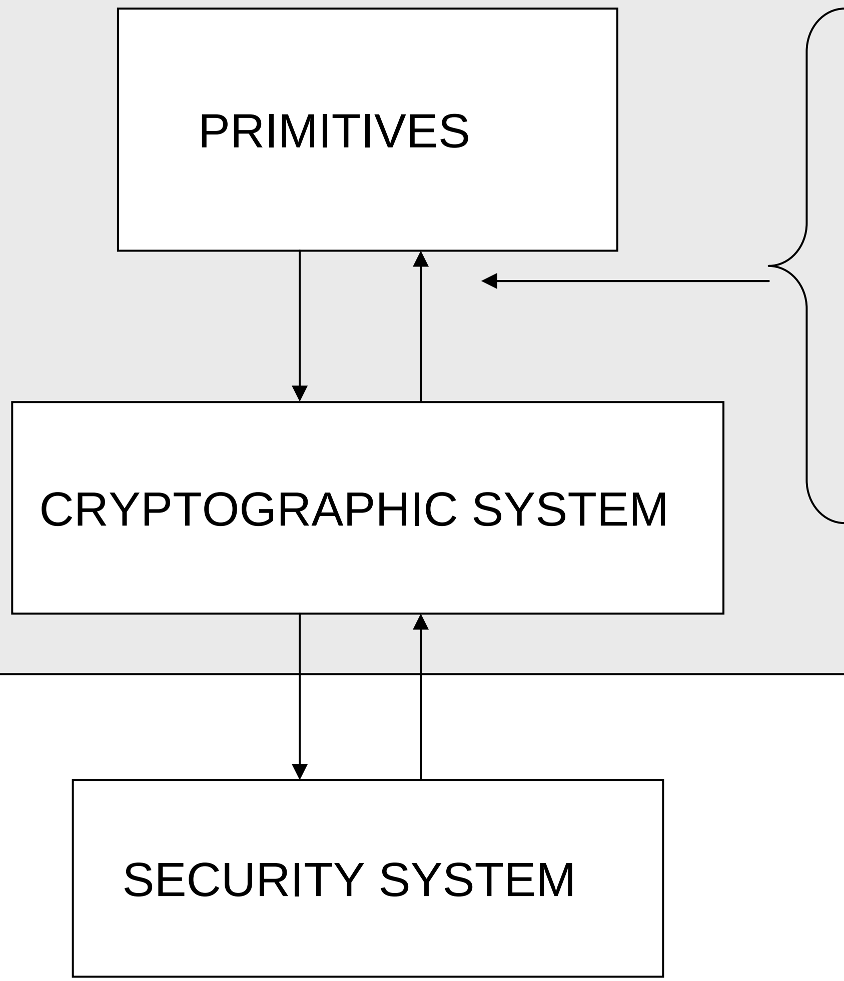
Problem: Alice can decide HEADS or TAILS. This is not true coin flipping and may be an unfair advantage for some applications. Modify the protocol that Alice has no longer this advantage.

Solution: Set

- HEADS: Bob’s guess is correct
- TAILS: Bob’s guess is incorrect

Then outcome of the protocol, HEADS or TAILS, may not be selected by Alice or Bob.

Cryptologic research



Formalizes the relationship between mathematical properties of the primitives and the security of a cryptographic system or protocol

e.g., assuming that one-way functions exist, a secure cryptographic system exists, and vice versa.

Dolev-Yao Threat Model

Adversary called “Malice”

- He can obtain any message passing through the network.
- He is a legitimate user of the network, and thus in practice can initiate a conversation with any other user.
- He will have an opportunity to become a receiver to any principal.
- He can send message to any principal by impersonating any other principal.

Malice can be an individual adversary, a coalition of a group of adversaries, and he can, as a special case, be a legitimate principal in the protocol.

What Malice cannot do

- guess random numbers drawn uniformly from a sufficiently large;
- break perfect encryption, that is, without the knowledge of the secret key he cannot retrieve plaintext from a given ciphertext, nor create valid (!) ciphertext from given plaintext;
- break secure message authentication codes
- break public keys;
- invert one-way functions;

(these are all informal security assumptions about cryptographic primitives)

- cannot access private areas of computing or communications environment.

Rabin OT

Two players: sender (Alice) and receiver (Bob)

Goal: Alice has one bit. Bob is allowed to try once to get the bit. His success probability is ½ . Alice does not know, if Bob gets the bit or not.

Protocol:

1. Alice sets up an RSA cryptosystem: p, q, n, a, b, with $ab \equiv 1 \bmod \Phi(n)$.
2. Alice encrypts the bit s, gets $c = \{\text{encode}(s)\}^b \bmod n$, and sends c, b and n to Bob.
3. Bob selects x, $0 < x < n$, at random, computes $y = x^2 \bmod n$, and sends y to Alice.
4. Alice finds the four square roots of y and picks one, say z, of them and sends it to Bob.
5. If $z \neq \pm x \bmod n$, Bob can factor n, compute $a = b^{-1} \bmod \Phi(n)$, and decrypt c, with probability ½. Alice does not know if $z \neq \pm x \bmod n$.

1-out-of-2 OT using RSA

Two players: sender (Alice) and receiver (Bob)

Goal: Alice has two secret bits. Bob is allowed to see exactly one of them. Alice does not know, which of the two bits Bob gets.

Alice's inputs: two bits a_0 and a_1

Bob's input: one bit s

Protocol: OT(a_0, a_1; s)

Output to Alice: nothing

Output to Bob: $a_s = (s \oplus 1)\, a_0 \oplus s\, a_1$

Next we see how to implement OT(a_0, a_1; s) assuming Bob is honest, which is the case of "private information retrieval".

OT(a_0, a_1; s)

PREMISES: Alice sets up an RSA cryptosystem: p, q, n, a, b, with $ab \equiv 1 \bmod \phi(n)$, and sends n and b to Bob.

ASSUMPTION: Hard-core bit for the RSA function: For randomly chosen x, given y, n, b, where $y = x^b \bmod n$ finding the lsb of x is essentially as hard as finding all of x (see Chapter 9, Lecture 4)

1. Bob selects a random m with lsb r_s and computes the ciphertext $c_s = m^b \bmod n$. Bob selects $c_{1\text{-}s}$ at random, and sends c_s and $c_{1\text{-}s}$, that is, c_0 and c_1 to Alice.
2. Alice decrypts c_0 and c_1 and gets the lsb:s r_0 and r_1 of the plaintexts. She then conceals the bits a_0 and a_1 by computing $a'_0 = r_0 + a_0 \pmod 2$ and $a'_1 = r_1 + a_1 \bmod 2$, and sends a'_0 and a'_1 to Bob.
3. Bob then gets a_s from a'_s as he knows r_s. Alice does not know s.