10. PROTOCOLS TO DO SEEMINGLY IMPOSSIBLE

A protocol is an algorithm two (or more) parties have to follow to perform a communication.

A cryptographical protocol is a protocol to achieve secure communication when dealing with a communication or cooperation problem.

In this and next chapter we deal with a variety of cryptographical protocols that allow to solve seemingly unsolvable problems.

An important goal of the chapter is to show cryptographic protocols for such basic cryptographic primitives as bit commitment and oblivious transfer.

As an application we discuss and illustrate voting schemes.

COIN-FLIPPING PROTOCOLS

Coin-flipping by telephone: Alice and Bob got divorced and they do not trust each other any longer. They want to decide, communicating by phone only, who gets the car.

Protocol 1 Alice sends Bob messages *head* and *tail* encrypted by a one-way function f. Bob picks one of them and inform about his decision Alice. She then sends f to Bob.

Protocol 2 Alice chooses two large primes p, q, sends Bob $n = p \cdot q$ and keeps p, q secret.

Bob chooses a random number $y \in \{1, \ldots, \frac{n}{2}\}$, sends Alice $x = y^2 \bmod n$ and tells Alice: if you guess y, car is yours.

Alice computes four square roots $(x_1, n - x_1)$ and $(x_2, n - x_2)$ of x.

Let

$$x'_1 = \min(x_1, n - x_1), x'_2 = \min(x_2, n - x_2).$$

Since $y \in \{1, \ldots, \frac{n}{2}\}$, either $y = x'_1$ or $y = x'_2$.

Alice then guesses whether $y = x'_1$ or $y = x'_2$ and tells Bob her choice (for example by reporting the position and value of the leftmost bit in which x'_1 and x'_2 differ).

Bob tells Alice whether her guess was correct or incorrect.

(Later, if necessary, Alice reveals p and q, and Bob reveals y.)

BIT COMMITMENT PROTOCOLS (BCP)

Basic ideas and solutions I

In a bit commitment protocol Alice chooses a bit and gets **committed** to it, in the following sense:

Bob has no way of knowing which commitment Alice has made, and Alice has no way of changing her commitment once she has made it; say after Bob announces his guess as to what Alice has chosen.

An example of a "pre-computer era" BCP is that Alice writes her commitment on a paper, locks it in a box, sends the key to Bob and, in the opening phase, she sends also box to Bob.

Complexity era solution I. Alice chooses a one-way function f and an even (odd) x if she wants to commit 0 (1) and sends to Bob $f(x)$ and f.

Problem: Alice may know an even x_1 and an odd x_2 such that $f(x_1) = f(x_2)$.

Complexity era solution II. Alice chooses a one-way function f, two random x_1, x_2 and a bit b she wishes to commit to, and sends to Bob $(f(x_1, x_2, b), x_1)$ – a commitment.

When times comes for Alice to reveal her bit she sends to Bob f and the triple (x_1, x_2, b).

BIT COMMITMENT SCHEMES

The basis of bit commitment protocols are bit commitment schemes:

A bit commitment scheme is a mapping $f : \{0,1\} \otimes X \to Y$, where X and Y are finite sets.

A **commitment** to a $b \in \{0,1\}$, or an encryption of b, is any value (called **blow**) $f(b,x), x \in X$.

Each commitment protocol has two stages:

Commitment stage: The sender sends a bit b he wants to commit to, in encrypted form to the receiver.

Opening stage: If required, the senders sends to receiver additional information that enables the receiver to get b.

Each bit commitment scheme should have three properties:

Hiding: For no $b \in \{0,1\}$ and $x \in X$, it is feasible for Bob to determine b from $B = f(b,x)$.

Binding: Alice can "open" her commitment B, by revealing (opening) x and b such that $B = f(b,x)$, but she should not be able to open a commitment (blow) B as both 0 and 1.

Viability: If both, the sender and the receiver follow the protocol, the receiver will always recover the committed value.

TWO SOLUTIONS

Bit commitment scheme I. p, q are large primes, $n = pq$, $m \in QNR(n)$, $X = Y = \mathbf{Z}_n^*$, n, m are public.

$$f(b, x) = m^b x^2 \bmod n.$$

Since computation of quadratic residues is in general infeasible this bit commitment scheme is hiding.

Since $m \in QNR(n)$, there are no x_1, x_2 such that $mx_1^2 = x_2^2 \pmod{n}$ and therefore the scheme is binding.

Bit commitment scheme II. p is a large Blum prime, $X = \{0, 1, \ldots, p-1\} = Y$, α is a primitive element of $\mathbf{Z}_p^*$.

$$f(b, x) = \begin{cases} \alpha^x \bmod p, & \text{if } SLB(x) = b; \\ \alpha^{p-x} \bmod p & \text{if } SLB(x) \neq b \end{cases}$$

where

$$SLB(x) = \begin{cases} 0 & \text{if } x \equiv 0, 1 \pmod{4} \\ 1 & \text{if } x = 2, 3 \pmod{4}; \end{cases}$$

The fact that this bit commitment scheme is binding follows from the fact that in the case of discrete logarithms modulo Blum primes there is no effective way to determine second least significant bit (SLB) of the discrete logarithm.

COIN TOSSING BY PHONE- revisited

Each bit commitment scheme can be used to solve coin tossing by phone problem as follows:

- Alice tosses a coin, commits itself to its outcome b_A (say heads =0, tails =1) and sends the commitment to Bob.
- Bob also tosses a coin and sends the outcome b_B to Alice.
- Alice open her commitment.
- Both Alice and Bob compute $b = b_A \oplus b_B$.

Observe that if at least one of the parties follow the protocol, that is it tosses a random coin, the outcome is indeed a random bit.

Note: If the hiding or the binding property of a commitment protocol deepends on the complexity of a computational problem, we speak about computational hiding and computational binding.

In case, the binding or the hidding property does not depend on the complexity of a computational problem, we speak about unconditional hiding or unconditional binding.

A commitment scheme based on discr. log.

Alice commits herself to an $m \in \{0, \ldots, q-1\}$.

Scheme setting:

Bob randomly chooses primes p and q such that $q|(p-1)$.

Bob chooses random generators $g \neq 1 \neq v$ of the subgroup G of order q in $\mathbf{Z}_n^*$.

Bob sends p, q, g and v to Alice.

Commitment phase: Alice verifies commitment phase. To commit to an $m \in \{0, \ldots, q-1\}$, she chooses a random $r \in \{0, \ldots, q-1\}$, and sends $c = g^r v^m$ to Bob.

Opening phase: Alice sends r and m to Bob who then verifies whether $c = g^r v^m$.

COMMENTS

- If Alice, commited to an m, could open her commitment as $\bar{m} \neq m$, then $g^r v^m = g^{\bar{r}} v^{\bar{m}}$ and therefore

$$lg_g v = (r - \bar{r})(\bar{m} - m)^{-1}.$$

  Hence, Alice could commpute $\lg_g v$ of a randomly chosen element $v \in G$, what contradicts the assumption that computation of discrete logarithms in G is infeasible.

- Since g and v are generators of G, then g^r is a uniformly chosen random element in G, perfectly hiding v^m and m in $g^r v^m$, as in the encryption with ONE-TIME PAD cryptosystem.

ELECTRONIC VOTING

Let com$(r,m) = g^r v^m$ denote commitment to m in the commitment scheme based on discrete logarithm. If $r_1, r_2, m_1, m_2 \in \{0,\dots,q-1\}$, then

$$\text{com}(r_1,m_1) \times \text{com}(r_2,m_2) = \text{com}(r_1+r_2, m_1+m_2).$$

Commitment schemes with such a property are called **homomorphic commiment schemes**.

Homomorphic scemes can be use to cast yes-no votes of n voters $V_1,\dots,V_n$ by trusted center T for whom e_T and d_T are ElGamal encryption and decryption algorithms.

Each voter V_i chooses his vote $m_i \in \{0,1\}$, a random $r_i \in \{0,\dots,q-1\}$ and computes his voting commitment $c_i =$ com(r_i, m_i). Then V_i makes c_i public and sends $e_T(g^{r_i})$ to T who computes

$$d_T(\prod_{i=1}^{n} e_T(g^{r_i})) = \prod_{i=1}^{n} g^{r_i} = g^r,$$

where $r = \Sigma_{i=1}^n r_i$ and makes public g^r.

Now, anybody can compute the result s of voting from publically known c_i and g^r since

$$v^s = \frac{\Pi_{i=1}^n}{c_i},$$

with $s = \Sigma_{i=1}^n m_i$.

s can be derived from v^s by computing $v^1, v^2, v^3, \dots$ and comparing with v^s if the number of voters is not too large.

OBLIVIOUS TRANSFER

Story: Alice knows a secret and wants to send secret to Bob in such a way that he gets secret with probability $\frac{1}{2}$, and he knows whether he got secret, but Alice has no idea whether he received secret. (Or Alice has several secrets and Bob wants to buy one of them but he does not want that Alice knows which one he bought.)

Oblivious transfer problem: Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability $\frac{1}{2}$ and "garbage" with the probability $\frac{1}{2}$. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

Protocol (1) Alice chooses two large primes p and q and sends $n = pq$ to Bob.
(2) Bob chooses a random number x and sends $y = x^2 \bmod n$ to Alice.
(3) Alice computes four square roots $\pm x_1, \pm x_2$ of $y \pmod n$ and sends one of them to Bob. (She can do it, but has no idea which of them is x.)
(4) Bob checks whether the number he got is congruent to x. If yes, he has received no new information. Otherwise, Bob has two different square roots modulo n and can factor n. Alice has no way of knowing whether this is the case.

Mental poker playing by phone – two players

Basic requirements:

- All hands (sets of 5 cards) are equally likely.
- The hands of Alice and Bob are disjoint.
- Both players know their own hand but not that of the opponent.
- Each player can detect eventual cheating of the other player.

A commutative cryptosystem is used with all functions kept secret. Players agree on numbers $w_1, \ldots, w_{52}$ as the names of 52 cards.

Protocol:

(1) Bob shuffles cards, encrypts them with e_B, and tells $e_B(w_1), \ldots, e_B(w_{52})$, in a randomly chosen order, to Alice.

(2) Alice chooses five of the items $e_B(w_i)$ as Bob's hands and tells them Bob.

(3) Alice chooses another five of $e_B(w_i)$, encrypt them with e_A and sends to Bob.

(4) Bob applies d_B to five values $e_A(e_B(w_i))$ he got from Alice and sends $e_A(w_i)$ to Alice as Alice's hands.

Remarque: The cryptosystem that is used cannot be public-key in the normal sense. Otherwise Alice could compute $e_B(w_i)$ and deal with the cards accordingly – a good hand for B but slightly better for herself.

Mental poker with three players

1. Alice encrypts 52 cards $w_1, \ldots, w_{52}$ with e_A and sends them in a random order to Bob.
2. Bob, who cannot read the cards, chooses 5 of them, randomly. He encrypts them with e_B, and sends $e_B(e_A(w_i))$ to Alice and the remaining 47 encrypted messages $e_A(w_i)$ to Carol.
3. Carol, who cannot read any of the messages, chooses five at random, encrypts them with her key and sends Alice $e_C(e_A(w_i))$.
4. Alice, who cannot read encrypted messages from Bob and Carol, decrypt them with her key and sends back to the senders,

   five $d_A(e_B(e_A(w_i))) = e_B(w_i)$ to Bob,

   five $d_A(e_C(e_A(w_i))) = e_C(w_i)$ to Carol.
5. Bob and Carol decrypt the messages to learn their hands.
6. Carol chooses randomly 5 other messages $e_A(w_i)$ from the remaining 42 and sends them to Alice.
7. Alice decrypt messages to learn her hands.

Additional cards can be dealt with in a similar manner. If either Bob or Carol wants a card, they take an encrypted message $e_A(w_i)$ and go through the protocol with Alice. If Alice wants a card, whoever currently has the deck sends her a card.

SECURE ELECTIONS

The ideal voting protocol should have at least the following properties:

1. Only authorized voters can vote.
2. No one can vote more than once.
3. No one can determine for whom anyone else voted.
4. No one can change anyone else vote without being discovered.
5. All voters can make sure that their votes were counted.

Additional requirement: Everyone knows who voted and who didn't.

Very simple voting protocol I.

- All voters encrypt their vote with the public key of a Central Election Board (CEB).
- All voters send their votes to the CEB.
- CEB decrypts votes, tabulates them and makes the result public.

The protocol has problem with some of the required properties.

Simple voting protocol II.

- Each voter V_i signs his/her vote v_i with his/her private key – $d_{V_i}(v_i)$.
- Each voter encrypts his/her signed vote with the CEB's public key – $e_{CEB}(d_{V_i}(v_i))$.
- All voters send their votes to CEB.
- CEB decrypts the votes, verifies signatures, tabulates votes and makes the result public.

Voting protocol (Nurmi, Salomaa, Santean, 69)

• CEB publishes a list of all legitimate voters.
• Within a given deadline, everybody intended to vote reports his/her intention to CEB.
• CEB publishes a list of voters participating in elections.
• Each voter V receives an identification number, i, using a special protocol that very likely assigns different numbers to different users.
• Each voter V creates a public encryption function e_V and secret decryption function d_V.
•* If v is a vote of the voter V, then V generates the following message and sends it to CEB:

$$(i, e_V(i, v))$$

• The CEB acknowledges the receipt of the vote by publishing $e_V(i, v)$.
• Each voter V sends to CEB the pair (i_V, d_V).
• The CEB uses d_V to decrypt the vote $(i, e_V(i, v))$.
• At the end of the elections CEB publishes the results of the election and, for each different vote, the list of all $e_V(i, v)$-values that contained that vote.
• It is possible that two voters get the same identification number. In such a case, the CEB generates a new identification number, i_1, chooses one of two votes, and publishes: $(i_1, e_V(i, v))$. The owner of that vote recognizes that and sends in a second vote, repeating step ($\star$) with the new identification number i_1.

Anonymous money order

Digital cash idea has one big problem: how to hide to whom you gave the money.

Protocol 1

(1) Alice prepares 100 anonymous money order for 1000\$.
(2) Alice puts one money order, and a piece of carbon paper, into each of 100 different envelopes and gives them to the bank.
(3) The bank opens 99 envelopes and confirms that each is a money order for 1000\$.
(4) The bank signs the remaining unopened envelope. The signature goes through the carbon paper to the money order. The bank hands the unopened envelope back to Alice and deletes 1000\$ from her account.
(5) Alice opens the envelope and spends the money order with a merchant.
(6) The merchant checks for the bank's signature to make sure the money order is legitimate.
(7) The merchant takes the money order to the bank.
(8) The bank verifies its signature and credits \$1000 to the merchnt's account.

(Alice has a 1% chance of cheating – the bank can make penalty for cheating so large that this does not pay of.)

Multi-authority election scheme

Basic idea:

- There are many voters and an n-member election boards.
- Voting is an YES-NO voting and majority of votes decides.
- Election Board uses El Gamal public key with trapdoor information y.
- A Central Authority uses Shamir's (n, t)-secret sharing scheme to distribute (secret) y to all n members of election board with member M_i geting secret share y_i.
- During voting each voter V_i commits himself to a vote $v_i \in \{1, -1\}$. V_i commits himself to vote v_i by encrypting it with the election board public key and sends the outcome to publically accessible common memory of election board.
- Since ElGamal commitment scheme is homomorphic election board can compute encrypted version of the sum of votes v_i.
- After elections are over, everybody can get the result of the voting provided t members of the election board cooperate with him.