Οικονομικό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής
ΠΜΣ στα Πληροφοριακά Συστήματα

Κρυπτογραφία και Εφαρμογές

Μαριάς Ιωάννης
marias@aueb.gr

Μαρκάκης Ευάγγελος
markakis@gmail.com

Outline

- Key Establishment
  - ✓ Key Distribution Centers
  - ✓ Diffie-Hellman protocol
  - ✓ Shamir's protocol
- Secret Sharing
  - ✓ (t, n)-threshold schemes (Shamir)
  - ✓ (t, t)-threshold schemes
- Secret sharing with more general access structures
  - ✓ The Monotone Circuit Construction
- Bit Commitment Protocols
  - ✓ With Symmetric Cryptography
  - ✓ With Public-key Cryptography
  - ✓ With Hash Functions

Part 1: Key Establishment

Key Distribution

How should 2 entities agree on a key?

- It can be a key for a symmetric cryptosystem, or for various other applications
- Types of keys:
  - Session keys: used only for 1 session
  - Long-lived keys (also known as terminal keys): used for more than 1 session, need to be securely stored
  - Master keys: A key that can be used for creating a session key or a long-lived key.
- Objectives of the adversary:
  - To fool Alice and Bob into accepting an invalid key as valid
  - To make Alice or Bob believe that they have exchanged a key when they have not done so
  - To determine (partial) information about the key being exchanged

Key Distribution

Key establishment

✓ It can be implemented in 2 ways

- Using a Key Distribution Center (KDC), also referred to as a Trusted Authority

- Without the use of a Center: Alice and Bob execute some protocol on their own to agree on a key (such protocols are then referred to as Key Agreement Protocols)

Key Distribution Centers

- Suppose that a set of entities want to communicate with each other in a single session
- The session key is produced either by the KDC or by one of the communicating entities
- Each entity is using a long-lived key to communicate with KDC, i.e., we use long-lived keys to protect session keys

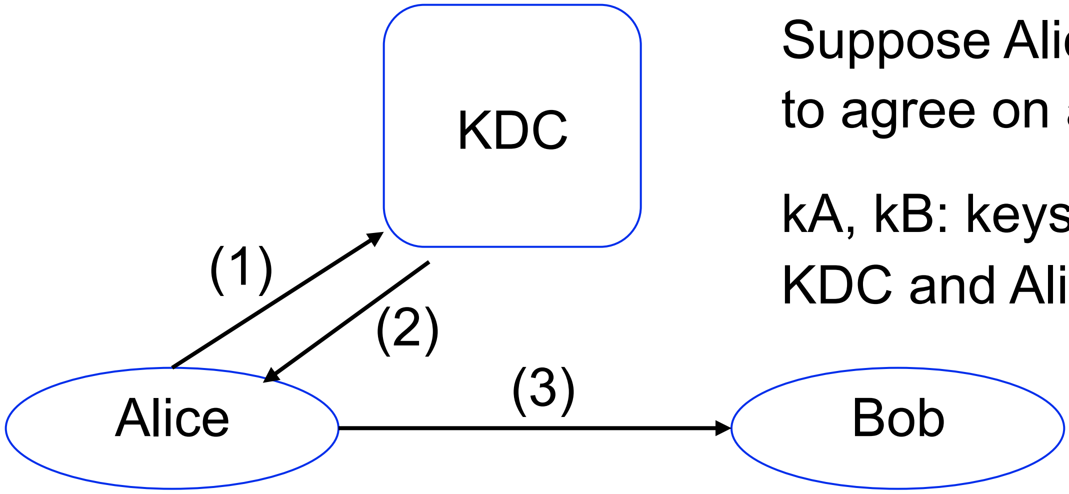
Main Advantage:

- Less long-lived keys required: For a total number of n entities communicating with each other, we need only n long-lived keys instead of $O(n^2)$ in the absence of KDC

Concerns:

- Security is based on the security of the long-lived keys (they must be securely produced and stored)

Key Distribution Centers



Suppose Alice and Bob want to agree on a session key K_s

kA, kB: keys shared between KDC and Alice (resp. Bob)

(1) Alice sends a request for a session key to the center

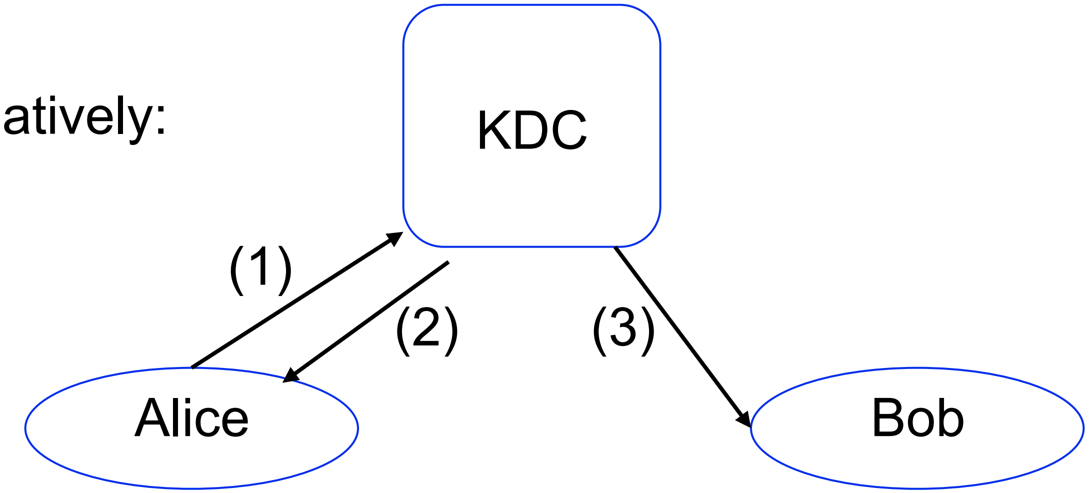
(2) KDC generates the session key K_s and sends to Alice $e_{kA}(K_s)$ and $e_{kB}(K_s)$

(3) Alice decrypts $e_{kA}(K_s)$ and also sends $e_{kB}(K_s)$ to Bob who can decrypt it

We need a secure channel for the exchange of the long-lived keys kA, kB between KDC and the entities

Key Distribution Centers

Alternatively:



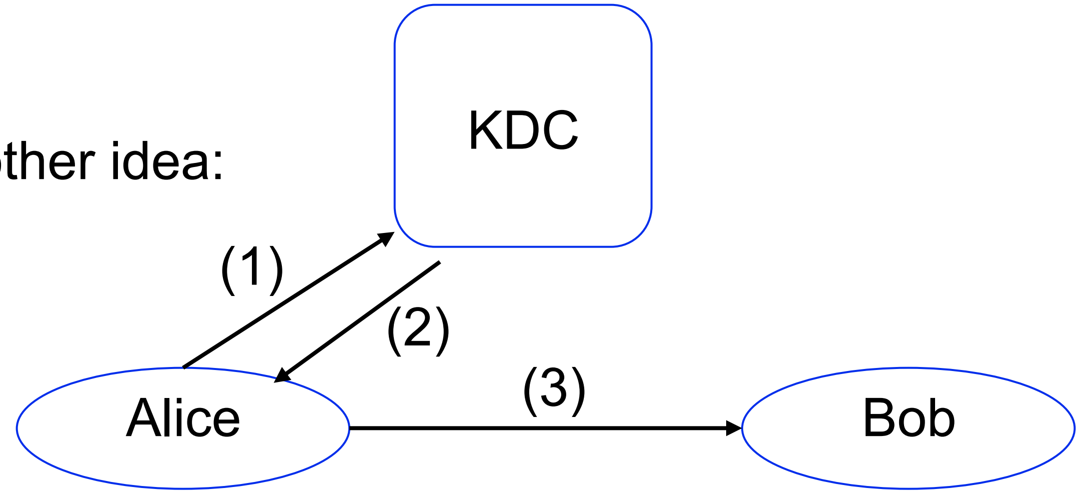
(1) Alice sends a request for a session key to the center

(2) KDC generates the session key K_s and sends to Alice $e_{kA}(K_s)$

(3) KDC sends to Bob $e_{kB}(K_s)$

Key Distribution Centers

And yet another idea:



(1) Alice generates the key K_s and sends $e_{kA}(K_s)$ to KDC

(2) KDC decrypts using kA, then encrypts using kB and sends to Alice $e_{kB}(K_s)$

(3) Alice sends $e_{kB}(K_s)$ to Bob who can decrypt it

In another variation, KDC can send $e_{kB}(K_s)$ directly to Bob

Key Distribution Centers

- Variations used in modern protocols to prevent active attacks:
  - ✓ Timestamps: Adding current time to the messages exchanged (h:min:sec)
  - ✓ Nonce: a unique number used in each transmission of information (it sometimes replaces timestamps)
  - ✓ Used for message and entity authentication
    - ✓ These numbers become part of the message and protect a message from being re-used again by an adversary in the future
  - ✓ **Requirements:** It should not be easy for the adversary to predict the nonce numbers; the algorithm for producing nonce numbers should not repeat numbers (or should not do so in an easy to decide pattern)

Key Distribution Centers

✓ Examples:

- One-Pass Protocols

(1) Alice → KDC: ID_A || ID_B || n_A //n_A = nonce number derived by Alice
ID_B and n_A can be also transmitted in an encrypted form

(2) KDC → Alice: $e_{kA}(n_A \,||\, ID_B \,||\, K_s \,||\, e_{kB}(K_s \,||\, ID_A))$

(3) Alice → Bob: $e_{kB}(K_s \,||\, ID_A)$

- Challenge-and-Response Protocols

(1) Alice → KDC: ID_A || ID_B

(2) KDC → Alice: $e_{kA}(K_s \,||\, ID_B \,||\, t_k \,||\, e_{kB}(K_s \,||\, ID_A \,||\, t_k))$

(3) Alice → Bob: $n_A \,||\, e_{kB}(K_s \,||\, ID_A \,||\, t_k)$ //key tied with timestamp

(4) Bob → Alice: $n_B \,||\, e_{Ks}(n_A)$

(5) Alice → Bob: $e_{Ks}(n_B)$

Key Distribution Centers

- Examples in practice:
  - Kerberos: network authentication protocol
  - Developed by MIT
  - Various versions have been made available throughout the years
  - Used by many UNIX or UNIX-like systems
  - Windows 2000 and later also use it as an authentication method
  - Later versions have incorporated more features such as
    - AES encryption
    - Public-key cryptography

Key Distribution without KDC

- Kerberos as well as many other protocols need a KDC
- Can we eliminate the need for a KDC?
- [Diffie-Hellman 1976]: ideas of public-key cryptography used for key agreement protocols

Diffie-Hellman Key Agreement Protocol

- Recall the Discrete Logarithm Problem (DLP):
  - ✓ Given a group Z^*_p for a prime number p, a generator g of Z^*_p, and an element $\beta \in Z^*_p$, find an integer x, $0 \leq x \leq p-1$, such that $g^x = \beta \bmod p$
- 2 related problems:
  - ✓ Computational Diffie-Hellman (CDH):
  - ✓ Given a group Z^*_p for a prime number p, a generator g of Z^*_p, and the elements $g^x \bmod p$ and $g^y \bmod p$, find $g^{xy} \bmod p$
  - ✓ Decision Diffie-Hellman (DDH):
  - ✓ Given a group Z^*_p for a prime number p, a generator g of Z^*_p, and the elements $g^x \bmod p$, $g^y \bmod p$, and $g^z \bmod p$, determine whether $z = xy \bmod p$
- DDH reduces to CDH, which reduces to DLP

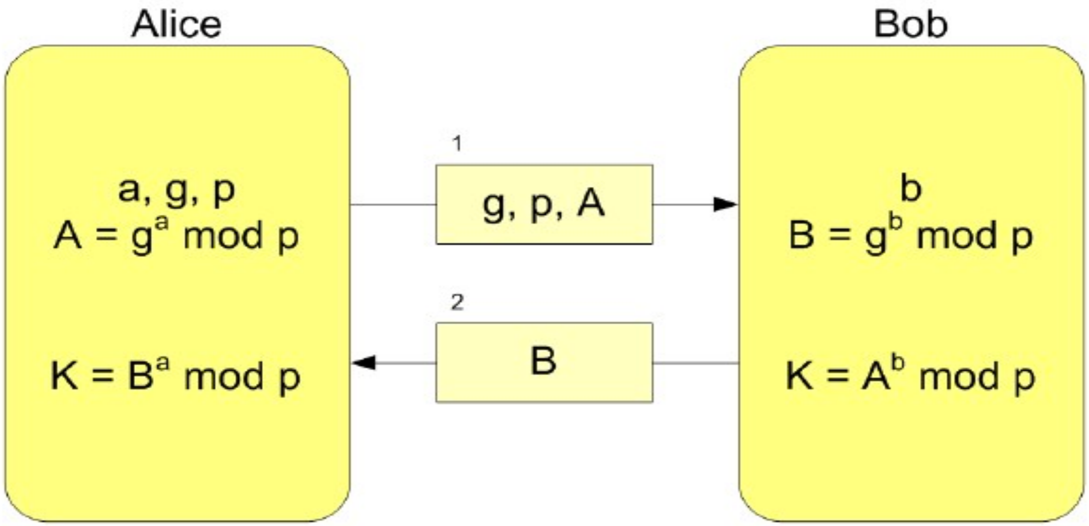
Diffie-Hellman Key Agreement Protocol

- ✓ Assume Alice and Bob communicate through an insecure channel
  - Step 1: Alice and Bob agree on a prime number p, and a generator g of Z^*_p
  - Step 2: Random numbers generation
    - Alice chooses an integer a, $0 < a < p-1$ and computes $Y_A = g^a \bmod p$
    - Bob chooses an integer b, $0 < b < p-1$ and computes $Y_B = g^b \bmod p$
  - Step 3: Alice and Bob exchange the values Y_A and Y_B
    - The values of a and b are kept secret
  - Step 4: Key generation
    - Alice computes $K = (Y_B)^a \bmod p$
    - Bob computes $K = (Y_A)^b \bmod p$

Diffie-Hellman Key Agreement Protocol

$$Y_A = g^a \bmod p$$

$$Y_B = g^b \bmod p$$



The 2 computations produce the same outcome:

$$K = (Y_B)^a \bmod p = (g^b \bmod p)^a \bmod p = (g^b)^a \bmod p = g^{ab} \bmod p$$

$$= g^{ab} \bmod p = (g^a \bmod p)^b \bmod p = (Y_A)^b \bmod p$$

Diffie-Hellman Key Agreement Protocol

Example: Suppose p = 71

Consider the generator g = 7

Alice and Bob choose a=5 and b=12 respectively

The corresponding public quantities are

For Alice: $Y_A = 7^5 \bmod 71 = 51 \bmod 71$

For Bob: $Y_B = 7^{12} \bmod 71 = 4 \bmod 71$

After exchanging the quantities Y_A and Y_B, they can compute the key K:

Alice: $K = (Y_B)^a \bmod p = (4 \bmod 71)^5 \bmod 71 = 4^5 \bmod 71 = 30 \bmod 71$

Bob: $K = (Y_A)^b \bmod p = (51 \bmod 71)^{12} \bmod 71 = 51^{12} \bmod 71 = 30 \bmod 71$

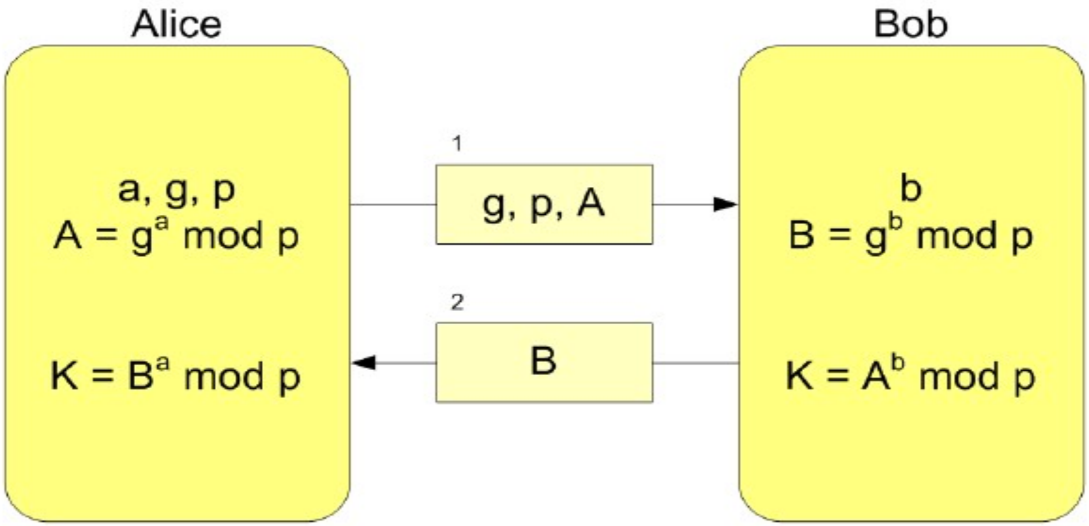
Diffie-Hellman Key Agreement Protocol

- ✓ Implementation on elliptic curves
  - Step 1: Alice and Bob agree on an elliptic curve mod p, say $y^2 = x^3 + ax + b$ modp and one of its generators, $G = (x_1, y_1)$
    - If the curve itself is not a cyclic group we select a generator for a large cyclic subgroup of the elliptic curve
  - Step 2: Random numbers generation
    - Alice chooses an integer a, $1 < a < p$ and computes $Y_A = a \cdot G$ modp
    - Bob chooses an integer b, $1 < b < p$ and computes $Y_B = b \cdot G$ modp
  - Step 3: Alice and Bob exchange the values Y_A and Y_B
    - The values of a and b are kept secret
  - Step 4: Key generation
    - Alice and Bob compute $K = a \cdot b \cdot G$ modp
- ✓ In general, we can use any other cyclic group

Diffie-Hellman Key Agreement Protocol

$$Y_A = g^a \bmod p$$

$$Y_B = g^b \bmod p$$



Security of the protocol to passive attacks:

- Either Oscar will attempt to find a (or b) by trying to solve $Y_A = g^a \bmod p$ (DLP)
- Or he can try to find the key $K = g^{ab} \bmod p$, given that he knows $g^a \bmod p$ and $g^b \bmod p$ (CDH)
- He has to solve either DLP or CDH

Diffie-Hellman Key Agreement Protocol

Man-in-the-middle attack (active attack)

Alice		Oscar		Bob
g, p, a		c		b
	$g, p, g^a \bmod p$ →		$g, p, g^c \bmod p$ →	
	← $g^c \bmod p$		← $g^b \bmod p$	

Oscar pretends to Alice that he is Bob and to Bob that he is Alice

- He establishes the key $g^{ac} \bmod p$ between Alice and himself
- He establishes the key $g^{bc} \bmod p$ between Bob and himself
- Alice and Bob think they talk to each other

Shamir's Key Agreement Protocol

Alice		Bob
k, p, a		b
	$k^a \bmod p \longrightarrow$	
	$\longleftarrow (k^a)^b \bmod p$	
	$(k^{ab})^{a^{-1}} \bmod p \longrightarrow$	

$0 < a, b < p-1$

$\gcd(a, p-1) = 1$

$\gcd(b, p-1) = 1$

- k = agreed key
- Last step gives Bob k^b modp
- Raising to b^{-1} (mod p-1) yields k (by Fermat's theorem)

Shamir's Key Agreement Protocol

- ✓ Passive attacks: We are safe, the adversary would have to solve DLP in order to find k
- ✓ Active attacks: vulnerable to the man-in-the-middle-attack, same as Diffie-Hellman
- ✓ Solutions for man-in-the-middle: Protocols that use entity authentication during the key generation process (and not before)

Part 2: Secret Sharing

- Example 1

  - ✓ A bank has a vault that must be opened every day
  - ✓ The bank has hired 3 senior employees
  - ✓ The bank manager wants a system where:
    - no single employee can have access to the combination
    - any 2 of the employees can have access to the vault

- Example 2
  - ✓ Suppose the secret is a monetary amount with 6 digits
  - ✓ We could break it into 2 parts
  - ✓ Agent A receives the first one, and B the second.
    - E.g. A receives 968 and B receives 345
    - A realizes immediately that the amount is >968000 and <=999999
  - ✓ Partial information disclosure is usually not a desirable property in secret sharing
  - ✓ We usually want to enforce that a secret share makes all possible values for the secret equiprobable
  - ✓ Hence we should not just distribute segments of the secret

- Example 3
  - ✓ Suppose the secret S is a 4-bit string and there are 2 involved entities
    - E.g. S = 1011
  - ✓ Suppose we flip a coin 4 times resulting in HTTH = 0110 = s1
  - ✓ S XOR s1 = 1101 = s2
  - ✓ Distribute s1 and s2 to the 2 entities
  - ✓ None of s1, s2 can discover S
    - s1 is a random string
    - s2 also behaves like a random string
  - ✓ Can both entities together recover S?
    - Yes: s1 XOR s2

Secret Sharing Schemes

- A *secret sharing scheme* refers to a technique for distributing a *secret* among a group of participants
  - Each participant is allocated some partial information called a *share*
  - The secret is reassembled only when a *sufficient* number of the shares is combined together
- Secret sharing is a core cryptographic primitive for developing many *distributed cryptographic protocols* in which certain operations require collaboration among several participants
- Security assurance relies on the assumption that *a fraction of the participants follow the prescribed protocol honestly*

Secret Sharing Schemes

- Properties

- *Perfect*: If someone has access to less than the specified number of secret shares, then all possible values for the secret are *equiprobable*

- *Ideal*: Length of a secret share = length of the secret

- Unlike crypto-systems, the security of the schemes does not depend on some (unproven) hypothesis (e.g., hardness of factoring)

(*t*, *n*)-Threshold Schemes

- *(t, n)-threshold schemes* involve a sharing phase and a reconstruction phase:

  i. <u>The *sharing phase*:</u> A *dealer D*, who holds some secret *M*, calculates $n \geq 2$ shares $z_1,\ldots,z_n$ of *M* and distributes them privately to a *set of n participants P* so that
     - Any $t \leq n$ shares enable one to recover the secret
     - t - 1 shares do not reveal any information about the secret
     - The sharing phase usually consists of an initialization phase and a share distribution phase.

  ii. <u>The *reconstruction phase*:</u> A subset of the participants $B \subseteq P$ combine their shares together to recover the secret *M*
     - If *t* or more participants pool their shares together, the secret should be recovered
     - If less than *t* participants pool their shares together, the secret should not be recovered

Hence, *the presence of at least t honest participants allows the secret to be reassembled in the reconstruction phase*

Example: Shamir's Threshold Scheme (1979)

The essential idea: *t points suffice to define a polynomial of degree t - 1*

The sharing phase:

- Assume that M lies in a finite field F, where $|F| > n$
- D constructs a random polynomial g of degree t - 1, where the constant term is equal to M, i.e., $g(0) = M$
- D computes n points $z_1 = g(a_1),\ldots,z_n = g(a_n)$ on the curve, where $a_1,\ldots,a_n$ are arbitrary nonzero elements of F
- D gives share (a_i, z_i) to participant i, $1 \leq i \leq n$

The reconstruction phase:

- Any subset of the participants $B \subseteq P$ where $|B| \geq t$ can recover M
- Any set of t points suffices to reconstruct g and thus compute $g(0) = M$ by means of polynomial interpolation
- Any set of t - 1 points does not reveal any information about z
  - ✓ There are many possible choices for the polynomial g, making every value for M equally likely

Example: Shamir's Threshold Scheme (1979)

The essential idea: *t points suffice to define a polynomial of degree t - 1*

The sharing phase:

- Let $F = F_{17} = GF(17) = (Z_{17}, +, *)$ with addition and multiplication mod 17
- Let $M = 13$, $n = 10$, $t = 6$
  - D constructs the polynomial $g(x) = 3x^5 + 10x^3 + 11x^2 + 5x + 13$
  - D computes 10 points $z_1 = g(1), \ldots, z_{10} = g(10)$ and gives point (i, z_i) to participant i, $1 \leq i \leq n$

The reconstruction phase:

- Any subset of the participants $B \subseteq P$ where $|B| \geq 6$ can recover z
- Any set of 6 points suffices to reconstruct g and thus compute $g(0) = 13$
- Any set of 5 points does not reveal any information about M

(t, n)-Threshold Schemes

- A simplified *(t, t)-threshold scheme* (when the secret lies in Z_m for some m>0):

  1) The dealer D chooses independently at random *t-1* elements of Z_m, say $y_1, y_2, \ldots, y_{t-1}$
  2) D computes
     - $y_t = M - (y_1 + y_2 + \ldots + y_{t-1}) \bmod m$
  3) For $i = 1,\ldots,t$, D gives the share y_i to participant *i*

- The *reconstruction phase*:
  - If all t participants pool their shares together, the secret can be recovered since $M = y_1 + y_2 + \ldots + y_{t-1} + y_t \bmod m$
  - If *t-1* participants pool their shares together, the secret cannot be recovered
  - For example if everyone except *i* collaborates, then they can infer the value of $M - y_i$
  - *But y_i is a uniform random variable. Hence so is* $M - y_i$ *(all values are possible for M)*

Access Structures

- Sometimes we may want to impose different types of constraints on which groups of participants can recover the secret [Ito, Saito, Nishizeki '87]
- The most general situation: Let Γ be a set of subsets of P. We want
  - every subset of Γ to be able to recover the secret
  - every other subset to not be able to recover the secret
- Γ is called an *access structure*, and any subset of Γ is called an *authorized subset*

Definition: A *perfect* secret sharing scheme realizing the access structure Γ is a scheme among n participants so that the following hold

- Any authorized subset S of Γ can determine the value of M if they pool their shares together
- Any unathorized subset can determine nothing about the value of M (all possible values for M are equally likely)

Shamir's (t, n)-threshold scheme realizes the access structure

$$\Gamma = \{S \subseteq P : |S| \geq t\}$$

Access Structures

Access structures usually satisfy the *monotone property:*

If $S \in \Gamma$ and T is a superset of S ($S \subseteq T$), then $T \in \Gamma$

(if a set of people can recover the secret by pooling their shares together, then any set with more people can also recover the secret)

- A set S of Γ is a minimal authorized subset if for any $A \subseteq S$ with $A \neq S$, $A \notin \Gamma$
- Γ_0 = the set of all minimal authorized subsets of Γ
- Γ_0 is called a basis of Γ (Γ is determined uniquely if we are given Γ_0)

Examples:

1. Shamir's (t, n)-threshold scheme: $\Gamma_0 = \{S: |S| = t\}$
2. If $\Gamma_0 = \{ \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3\} \}$ then

$$\Gamma = \Gamma_0 \cup \{ \{1, 2, 3\}, \{2, 3, 4\}, \{1, 2, 3, 4\} \}$$

Monotone Circuits

- Let C be a boolean circuit with
  - ✓ *n* boolean inputs $x_1, x_2, \ldots, x_n$ corresponding to the participants $P_1, P_2, \ldots, P_n$
  - ✓ 1 boolean output y
  - ✓ only OR and AND gates
  - ✓ each gate can have arbitrary fan-in (input wires) but fan-out = 1 (1 output wire)
- Such circuits are called monotone circuits
  - ✓ changing any input from 0 to 1 can never cause the output to change from 1 to 0
- For each truth assignment to the boolean variables let

$$S(x_1, x_2, \ldots, x_n) = \{ P_i : x_i = 1\}$$

- Each monotone circuit corresponds to the monotone access structure $\Gamma(C) = \{S(x_1, x_2, \ldots, x_n) : C(x_1, x_2, \ldots, x_n) = 1 \text{ (i.e. } y=1)\}$
  - ✓ Follows from the monotonicity of the circuit

The Monotone Circuit Construction

Idea for a secret sharing scheme realizing an access structure Γ (due to [Benaloh, Leichter '90])

1. First construct a circuit C such that Γ(C) = Γ
2. Then starting from the output, implement a secret sharing scheme on each gate assuming as "virtual" participants the input wires
3. The share of each participant P_i will be all the values calculated for gates that receive x_i as an input wire

Implementing Step 1

- Consider an access structure Γ
- Let Γ_0 be a basis for Γ
- The formula $OR_{B \in basis}$ ($AND_{i \in B}$ P_i) defines the desired circuit

Example:

Suppose $\Gamma_0 = \{ \{P_1, P_2\}, \{P_2, P_3, P_4\}, \{P_1, P_3, P_4\} \}$ is a basis of Γ

Then derive the disjunctive normal form formula:

$$\varphi = (P_1 \text{ AND } P_2) \text{ OR } (P_2 \text{ AND } P_3 \text{ AND } P_4) \text{ OR } (P_1 \text{ AND } P_3 \text{ AND } P_4)$$

Theorem: The circuit C implementing φ realizes Γ, i.e., $\Gamma(C) = \Gamma$

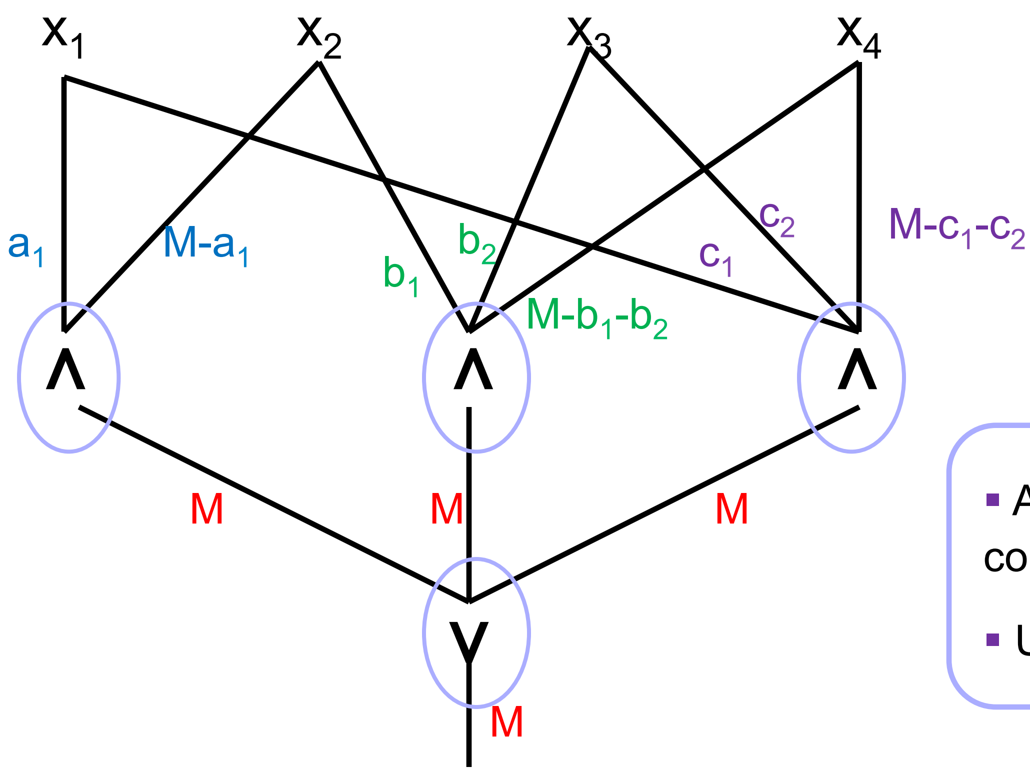
Observations:

- C is a circuit of depth 2
- Any other formula equivalent to φ is good (e.g. we could transform φ to conjunctive normal form or any other form we want).
- The circuit corresponding to the new formula would be equally good.

Implementing Steps 2-3

- Assign the value of M to the output wire
- Start from the bottom of the circuit and keep going up
- For each AND gate encountered, implement the (t, t)-scheme by considering
  - ✓ the input wires of the gate as the participants of the scheme
  - ✓ the value of the output wire as the secret
- For each OR gate
  - ✓ assign to the input wires the value of the output wire
- The share of participant P_i consists of all the values of input wires that start from x_i

The Monotone Circuit Construction



- M is in Z_m
- All calculations are mod m

- All authorized subsets can compute the secret
- Unauthorized subsets cannot

- Participant P_1 receives (a_1, c_1)
- Participant P_2 receives $(M-a_1, b_1)$
- Participant P_3 receives (b_2, c_2)
- Participant P_4 receives $(M-b_1-b_2, M-c_1-c_2)$

The secret sharing scheme more formally

- Set $f(y) = M$
- For all other wires W, $f(W)$ is initially undefined
- While there exists a wire W with $f(W)$ undefined {
  - ✓ Find a gate G such that $f(W_G)$ is defined (W_G = output wire of G) but $f(W)$ is not defined for all input wires W of G
  - ✓ If G is an OR gate
    - Set $f(W) = f(W_G)$
  - ✓ If G is an AND gate and it has t input wires $W_1, W_2, \ldots, W_t$ then implement the (t, t)-threshold scheme among these t participants and with $f(W_G)$ as the secret, i.e.:
    - Choose independently at random t-1 elements from Z_m
    - For i = 1,…, t-1, set each $f(W_i)$ to the i-th random element
    - Set $f(W_t) = f(W_G) – (f(W_1) + f(W_2) + \ldots + f(W_{t-1})) \bmod m$
- }
- Give to each participant P_i the values of all input wires that receive x_i as their input

Part 3: Bit Commitment Protocols

Bit Commitment Protocols

- Suppose that Alice and Bob want to play rock-paper-scissors by email or by phone
- We want Alice and Bob to commit first to their action before they send emails to each other with what they played.
- We should enforce that Alice cannot change what she played after she sees Bob's email.
- How can Bob be sure about that?

- Commitment protocols: they enforce Alice to commit to a certain value without forcing her to reveal the value.
- In the future, whenever Bob wants to see the actual value, Alice will be able to convince Bob that this was the value she committed to

Bit Commitment Protocols

A nice way to think about what we want to achieve:

- Alice is asked to commit to a message m without revealing it (m can be a monetary value, or a piece of text, a contract,…)
- She puts m in a safe and sends the safe to Bob
- She does not send Bob the combination (key) for the safe
- Later, when Bob wants to actually see m, Alice sends the key to Bob and he can open the safe
- Assuming that safes are secure and nobody can change their content, Bob sees the message and he is convinced that Alice had committed to this exact message in the past

- Q: Can we implement “virtual” safes by means of protocols?
- We can think first about committing just 1 bit

Bit Commitment Protocols

- Bit commitment protocols have 2 phases
  - ✓ Commit phase: Alice commits to her value (this involves some communication between Alice and Bob)
  - ✓ Reveal phase: Alice reveals her value (communication from Alice to Bob, who then checks if Alice tells the truth)

Bit Commitment Protocols

Implemetation using symmetric cryptography

- Let b = bit of Alice that Bob wants her to commit to
- Commit phase:
  - ✓ Bob → Alice: A random message m chosen by him
  - ✓ Alice then chooses a key and encrypts m || b
  - ✓ Alice → Bob: e_k(m || b)
- Reveal phase:
  - ✓ Alice sends the key k to Bob
  - ✓ Bob decrypts and checks to see that the first part consists of the message m. If yes, then he is convinced that the last bit of what he sees is the committed bit
  - ✓ If Alice does not send him the right key, the decrypted text will not be in the form m || b

Implemetation using ideas from public key cryptography

A) Based on quadratic residues

- A number y is a quadratic residue modn if there exists a number x such that $y = x^2$ modn
- Let b = bit of Alice that Bob wants her to commit to
- Commit phase:
  - ✓ Alice and Bob agree on a large composite number n, and on an element y from Z^*_n such that y is not a quadratic residue modn
  - ✓ Alice selects a number x from Z^*_n
  - ✓ Alice → Bob:

$$g = \begin{cases} x^2 \bmod n, & \text{if } b = 0 \\ yx^2 \bmod n, & \text{if } b = 1 \end{cases}$$

- Reveal phase:
  - ✓ Alice sends x to Bob

Bit Commitment Protocols

Implemetation using ideas from public key cryptography

A) Based on quadratic residues

- At the commit phase Bob cannot distinguish whether g is a quadratic residue
- He would need to decide if g has a square root modn
- Finding square roots modn, when n is composite, is equivalent to factoring

Implementation using ideas from public key cryptography

B) Based on DLP

- Let b = bit of Alice that Bob wants her to commit to
- Commit phase:
  - ✓ Alice and Bob agree on a large prime number p, on a generator α of Z^*_p, and on an element s from Z^*_p
  - ✓ Alice selects an integer x from Z^*_p
  - ✓ Alice → Bob:

$$g = \begin{cases} a^x \bmod p, & if\ b = 0 \\ sa^x \bmod p, & if\ b = 1 \end{cases}$$

- Reveal phase:
  - ✓ Alice sends x to Bob
- Bob would need to solve DLP in order to distinguish the form of g at the commit phase

Bit Commitment Protocols

Implemetation using hash functions

- Let x be the value of Alice that Bob wants her to commit to
- Commit phase:
  - ✓ Alice and Bob agree on a collision resistant hash function h
  - ✓ Alice → Bob: $y = h(x)$
- Reveal phase:
  - ✓ Alice sends x to Bob
- If h is collision resistant, it is difficult for Alice to find a value x' such that $h(x') = h(x)$