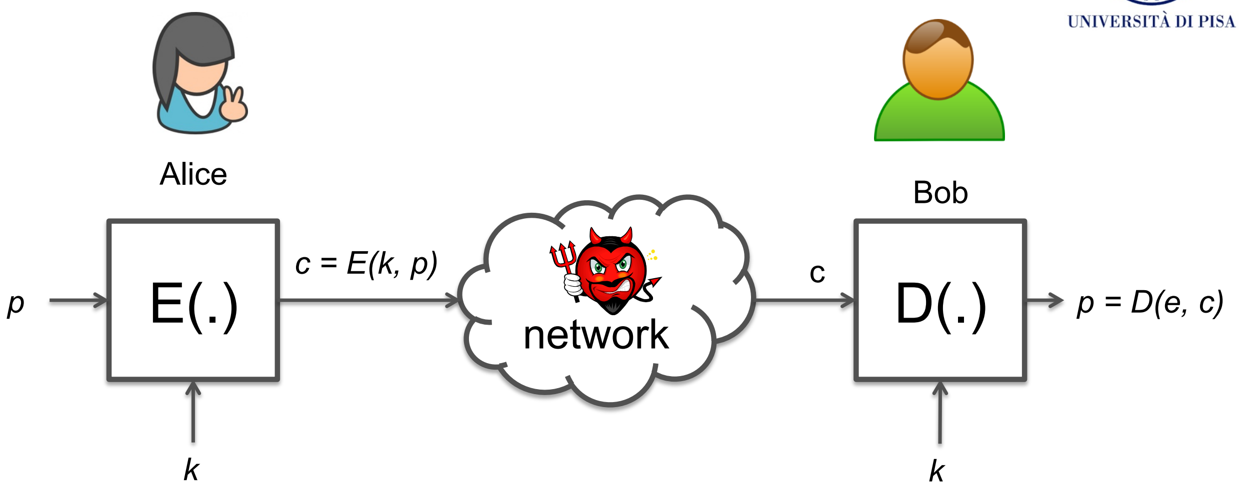
Key Management

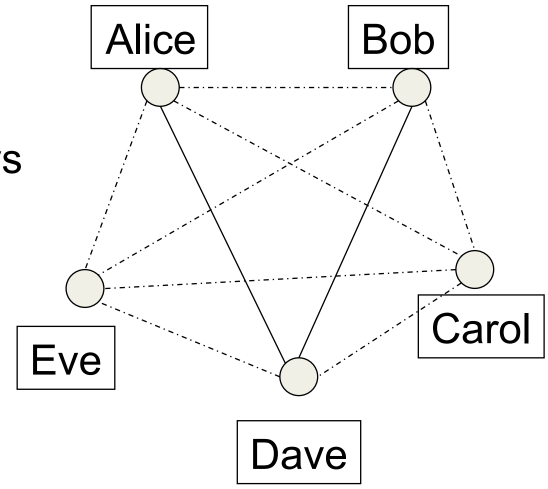
Shared key



- *k*: **shared secret key** (128 bits)

Pairwise keys

- Each pair of users shares an *long-term* secret key
- Properties
  - Every user stores (*n* -1) keys
  - The overall number of keys is $O(n^2)$

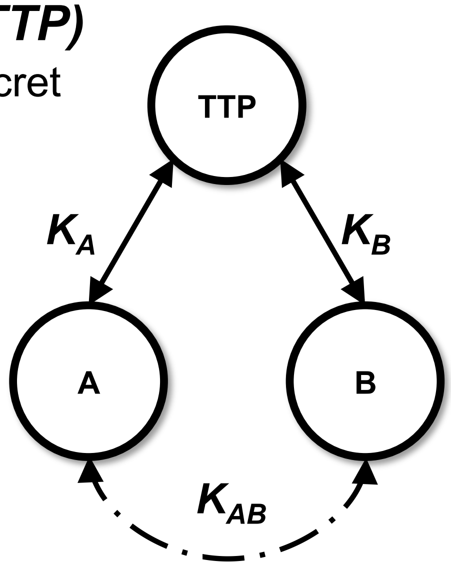


Pairwise keys

- **Pros**
  - If a subject is compromised only its communications are compromised;
    - communications between two other subjects are not compromised
    - We cannot do any better!
- **Cons**
  - Poor scalability: the number of keys is quadratic in the number of subjects
  - Poor scalability: a new member's joining and a member's leaving affect all current members

Trusted Third Party

- *Online Trusted Third Party (TTP)*
  - Each user shares a long-term secret key with TTP
  - Every user stores one key
  - The overall number of keys is n
- TTP is a single point of failure
  - TTP must be always online
  - TTP knows all the keys
    - TTP can read all msg between Alice and Bob
    - TTP can impersonate any party



Key distribution: a toy protocol

Alice wants a shared key with Bob. Eavesdropping security only

Bob (k_B) **Alice** (k_A) **TTP**

Alice → TTP: "Alice wants key with Bob"

TTP: choose random k_{AB}

TTP → Alice: $E(K_A, \text{"A, B"} \,||\, K_{AB})$

ticket $\leftarrow E(K_B, \text{"A, B"} \,||\, K_{AB})$

Alice → Bob: ticket

Bob: k_{AB} Alice: k_{AB}

Subject to replay attacks

Key distribution: toy protocol

UNIVERSITÀ DI PISA

- Insecure against replay attacks (active adversary)
  - Attacker records session between Alice and merchant Bob
    - For example: an order
  - Attacker replays session to Bob
    - Bob thinks Alice is ordering another copy of the book

TTP: to sum up

UNIVERSITÀ DI PISA

Pros

- It is easy to add and remove entities from the network
- Each entity needs to store only one long-term secret key

Cons

- TTP must be always online (availability)
- TTP must be efficient (performance)
- The TTP must store *n* long-term keys
- If the TTP is compromised, all communications are insecure (confidentiality and integrity)

Key question

UNIVERSITÀ DI PISA

- ***Can we generate shared keys without an online TTP?***
- Answer: YES!
- Starting point of public-key cryptography
  - Merkle (1974)
  - Diffie-Hellman (1976)
  - RSA (1977)
  - More recently: ID-based encryption (2001), functional encryption (2011)…

Key Management

DIFFIE-HELLMAN (1976)

Public key distribution system

UNIVERSITÀ DI PISA

- A **public key distribution system** allows two users to securely **exchange a key** over an **insecure channel**

2^{nd} most influential paper

UNIVERSITÀ DI PISA

- Whitfield Diffie and Martin Hellman, **New directions in cryptography**, IEEE Transactions of Information Theory, 22(6), pp. 644-654

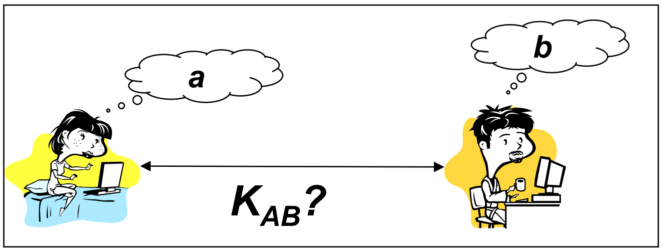
Number theory

- Multiplication is commutative
- $(a \times b) = (a \times b) \bmod n$
- Power of power is commutative
- $(a^b)^c = a^{bc} = a^{cb} = (a^c)^b \bmod n$

Number theory

- **Parameters**
  - Let p be **prime**
  - Let $g \in (1, p)$ be a **generator (primitive root)**, i.e., $\forall\ 1 \leq x < p$ ($\mathbb{Z}_p$), $\exists\ t$ s.t. $g^t \bmod p = x$
- **DISCRETE EXPONENTIATION**
  - Given g, p and x, to compute $\boldsymbol{y = g^x \bmod p}$ is ***computationally easy***
- **DISCRETE LOGARITHM**
  - Given g, $1 \leq y \leq p$-1, it is *computationally difficult* to determine x ($1 \leq x \leq p - 1$) s.t. $y = g^x \bmod p$

Diffie-Hellman



- Let p be a large prime (600 digits, 2000 bits)
- Let $1 \leq g < p$
- Let p e g publicly known

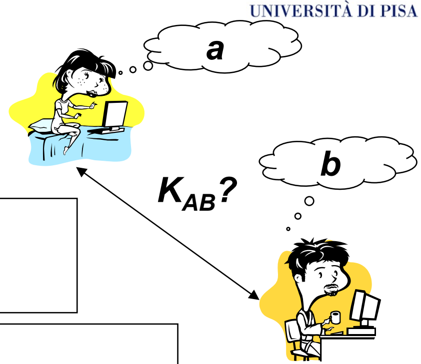
Alice chooses a random number a
Bob chooses a random number b

M1 A $\rightarrow$ B: A, $Y_A = g^a \bmod p$
M2 B $\rightarrow$ A: B, $Y_B = g^b \bmod p$

Alice computes $K_{AB} = (Y_B)^a \bmod p = g^{ab} \bmod p$
Bob computes $K_{AB} = (Y_A)^b \bmod p = g^{ab} \bmod p$

Diffie-Hellman with small numbers

Let p = 11, g = 7



Alice chooses a = 3 and computes $Y_A = g^a \bmod p = 7^3 \bmod 11 = 343 \bmod 11 = 2$

Bob chooses b = 6 and computes $Y_B = g^b \bmod p = 7^6 \bmod 11 = 117649 \bmod 11 = 4$

$A \rightarrow B$: 2
$B \rightarrow A$: 4

Alice receives 4 and computes $K_{AB} = (Y_B)^a \bmod p = 4^3 \bmod 11 = 9$

Bob receives 2 and computes $K_{AB} = (Y_A)^b \bmod p = 2^6 \bmod 11 = 9$

Security of Diffie-Hellman

UNIVERSITÀ DI PISA

- Eavesdropper sees p, g, Y_A and Y_B and wants to compute K_{AB}
- ***Diffie-Hellman Problem***
  - Given p, g, $Y_A = g^a \pmod{p}$ and $Y_B = g^b \pmod{p}$, compute g^{ab} modp
  - How hard is this problem?

Security of Diffie-Hellman

UNIVERSITÀ DI PISA

- If logs (mod *p*) are easily computed then DH-Problem can be easily solved
- There is no proof of the converse, i.e., if logs (mod *p*) are difficult then DH is secure
- We don't see any way to compute K_{AB} from Y_A and Y_B without first obtaining either *a* or *b*

How hard is Diffie-Hellman Problem

- Let p be a prime, $p < 2^n$
  - All quantities are representable as n-bit numbers
- Exponentiation takes at most $2 \times \log_2 p < 2n$ multiplications (mod p)
  - Linear in the exponent n
  - Exponent n may be very large
- Taking logs (mod p) requires $p^{½} = 2^{n/2}$ operations
- Example $n = 512$
  - Exponentiation requires at most 1024 multiplications
  - Taking logs mod p requires $2^{256} = 10^{77}$ operations

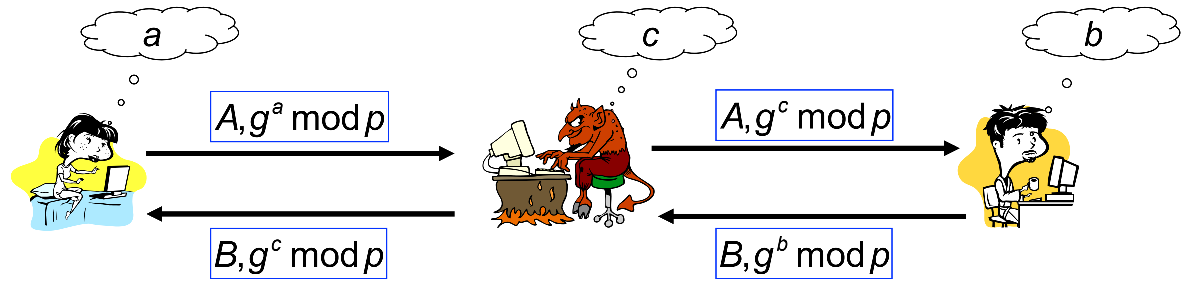
How hard is Diffie-Hellman

Cipher key size	modulus size	elliptic curve size
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	15360 bits	512 bits

slow

Slow transition from (mod p) to elliptic curves

Man-in-the-middle

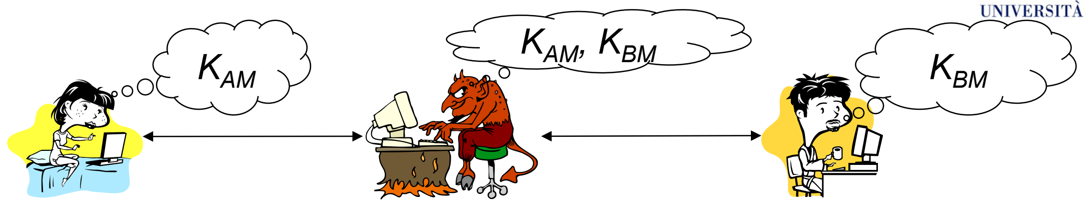


$K_{AM} = g^{ac} \bmod p$

$K_{AM} = g^{ac} \bmod p$, e
$K_{BM} = g^{bc} \bmod p$

$K_{BM} = g^{bc} \bmod p$

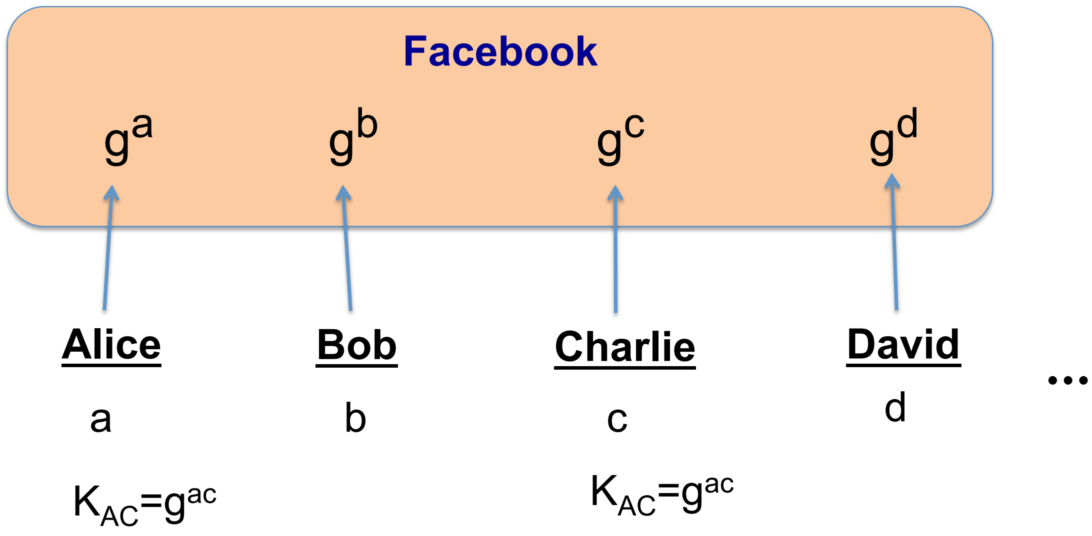
Man-in-the-middle



- Alice believes to communicate with Bob by means of K_{AM}
- Bob believes to communicate with Alice by means of K_{BM}
- The adversary can
  - read messages between Alice and Bob
  - impersonate Alice and Bob
- ***DH is insecure against an active attack***

Diffie-Hellman is not-interactive

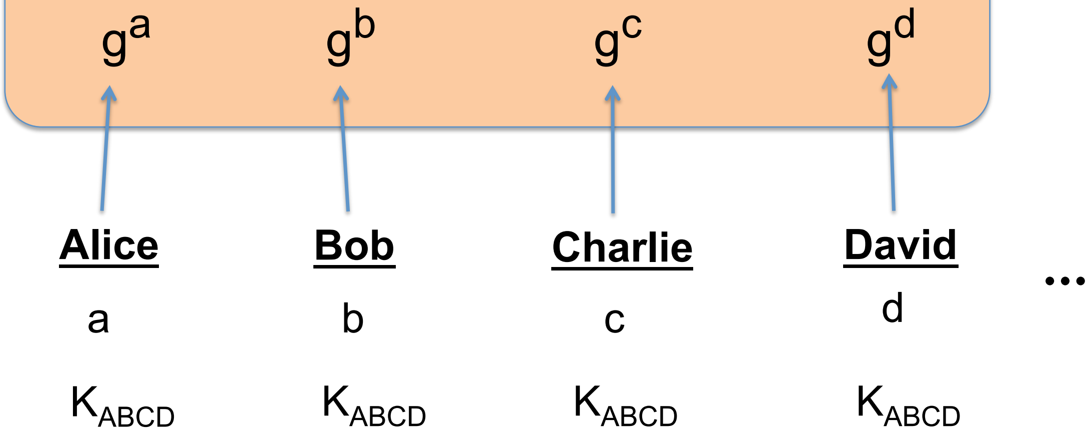
UNIVERSITÀ DI PISA



Diffie-Hellman: an open problem

UNIVERSITÀ DI PISA

n = 2 (DH)
n = 3 (Joux)
n ≥ 4: open



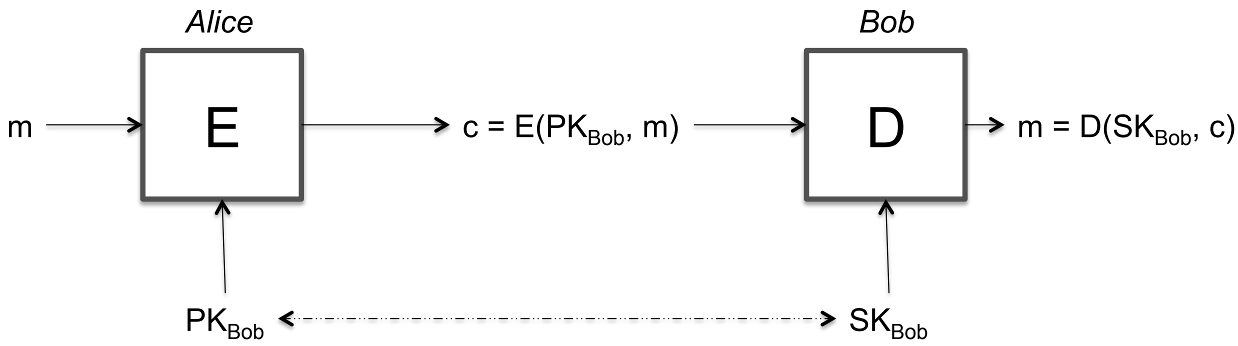
PUBLIC KEY ENCRYPTION

The fundamental question

UNIVERSITÀ DI PISA

- ***Can we generate shared keys without an online TTP?***
- PK encryption provides yet another answer to this question
  - DH is a public-key key distribution scheme
  - Public-key encryption is a more general encryption scheme

Public key encryption



PK: public key

SK: secret key

Public key encryption

- **DEF**. A public key encryption scheme is a triple of algs (G, E, D) s.t.
- **G**: randomized alg. for key generation (pk, sk)
- **E(pk, m)**: *randomized* alg. that takes m ∈ M and outputs c∈ C
- **D(sk, m)** deterministic alg. that takes c ∈ C and outputs m ∈ M or ⊥
- **Consistency**. ∀(pk, sk), ∀ m ∈ M, D(sk, E(pk, m)) = m

Security

- Known pk ∈ K and c ∈ C, it is computationally infeasible to find the message m ∈ M such that E(e, m) = c
- Known the public key pk ∈ K, it is computationally infeasible to determine the corresponding secret key sk ∈ K

- *Constructions generally rely on hard problems form number theory and algebra*

A PK encryption is not perfect

- PK encryption scheme is not perfect according to Shannon
  – Adversary intercepts CT c
  – Adversary selects PT m s.t. $Pr[M = m] \neq 0$
  – Adversary computes c' = E(pk, m)
  – If $c \neq c'$, then $Pr[M = m \mid C = c] = 0$

PK encryption - basic protocol

UNIVERSITÀ DI PISA

Alice — **Bob**

$(pk, sk) \leftarrow G()$

"Alice", pk →

Msg x

← Bob, $c \leftarrow E(pk, x)$

$x \leftarrow D(sk, c)$

Basic key transport protocol

UNIVERSITÀ DI PISA

Alice — **Bob**

$(pk, sk) \leftarrow G()$

"Alice", pk →

choose random key $x \in \{0,1\}^{128}$

← Bob, $c \leftarrow E(pk, x)$

$x \leftarrow D(sk, c)$

$z \leftarrow AES(x, msg)$ →

$msg \leftarrow AES(x, z)$

x: *session key*

Establish a secret key

Alice | **Bob**

$(pk, sk) \leftarrow G()$

"Alice", pk →

choose random $x \in \{0,1\}^{128}$

← Bob, $c \leftarrow E(pk, x)$

$x \leftarrow D(sk, c)$

x: shared key

Insecure against MITM

UNIVERSITÀ DI PISA

As described, the protocol is insecure against **active** attacks

Alice | **MiTM** | **Bob**

$(pk, sk) \leftarrow G()$ | $(pk', sk') \leftarrow G()$

"Alice", pk → | "Alice", pk' →

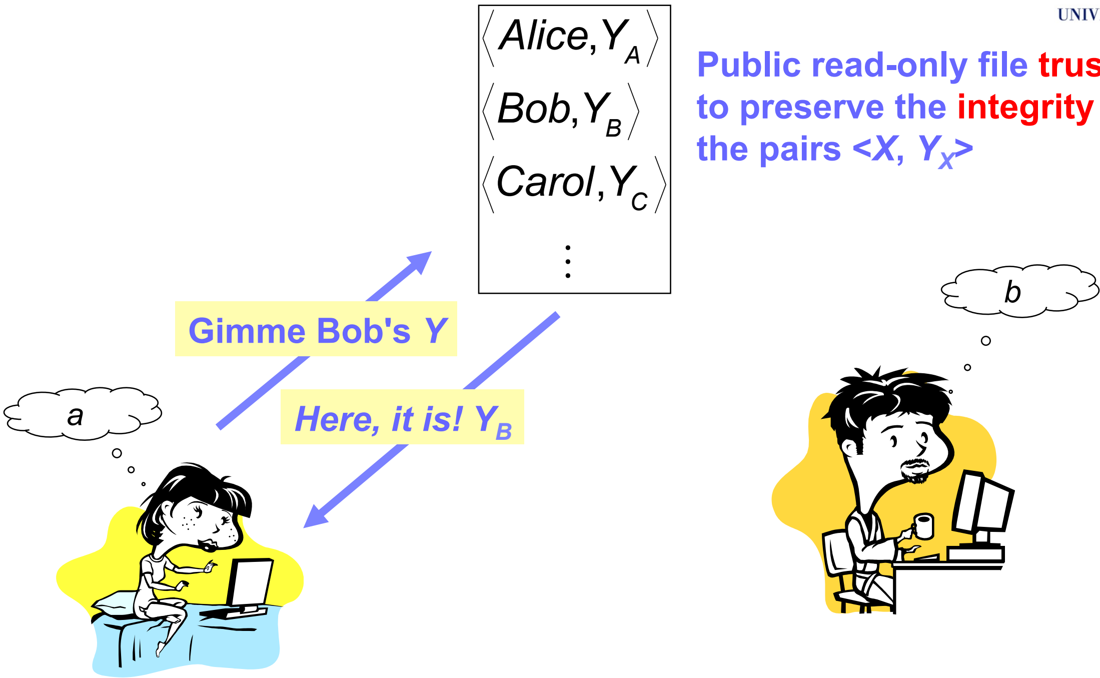
choose random $x \in \{0,1\}^{128}$

← "Bob", E(pk, x) | ← "Bob", E(pk', x)

x

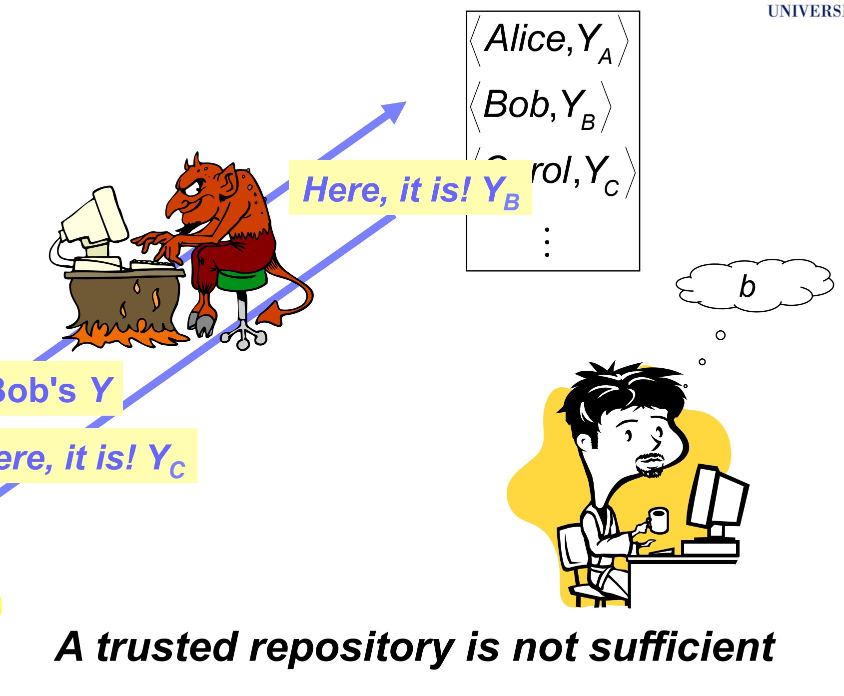
A trusted repository

Public read-only file **trusted** to preserve the **integrity** of the pairs $<X, Y_X>$



Diffie-Hellman protocol

The man-in-the-middle always lies in wait



A trusted repository is not sufficient

Key distribution with public encryption

UNIVERSITÀ DI PISA

- **Pros**
  - No TTP is required
  - The public file could reside with each entity
  - Only *n* public keys need to be stored to allow secure communications between any pair of entities, assuming that the only attack is that by a **passive adversary**
- **Cons**
  - Key management becomes more difficult in the presence of an ***active adversary***

Key distribution with public keys

UNIVERSITÀ DI PISA

- **A TTP certifies the public key of each entity**
- **S_T and V_T are the signing and verification keys of the TTP**

Public file

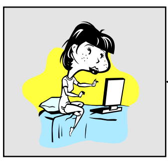
certificate

$\langle Alice, Y_A, s_A = S_T(Alice \| Y_A) \rangle$

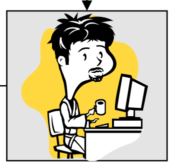
$\langle Bob, Y_B, s_B = S_T(Bob \| Y_B) \rangle$

$\langle Dave, Y_D, s_D = S_T(Dave \| Y_D) \rangle$

M



c



$V_T(Alice \| e_A, s_A)$

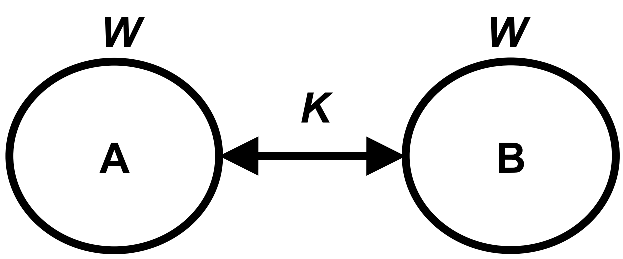
$c = E_{e_A}(m)$

Key Management

ESTABLISHING EPHEMERAL KEYS

Session/Ephemeral key

UNIVERSITÀ DI PISA



- *A* and *B a priori* share a ***long term key W***
- *A* and *B* wants to establish a ***session key K***

- Session key is used for bulk encryption
- A session key is used for one communication session
- Long term key is used for many runs of the key establishment protocols; in each run, the key encrypts a small amount of data

Establishing an ephemeral/ session key

UNIVERSITÀ DI PISA

one-pass

$$M1 \quad A \rightarrow B: \quad E\left(W, t_A \,||\, "B,A" \,||\, K\right)$$

- **t_A is a timestamp (a "fresh" quantity) requires synchronized clocks**

with challenge-response

$$M1 \quad A \leftarrow B: \quad n_B$$

$$M2 \quad A \rightarrow B: \quad E_W\left(W, n_B \,||\, "A,B" \,||\, K\right)$$

- **n_B is a nonce (a "fresh" quantity)**

both parties contribute to the session key

$$M1 \quad A \leftarrow B: \quad n_B$$

$$M2 \quad A \rightarrow B: \quad E\left(W, K_A \,||\, n_B \,||\, n_A \,||\, "A,B"\right)$$

$$M3 \quad A \leftarrow B \quad E\left(W, K_B \,||\, n_A \,||\, n_B \,||\, "B,A"\right)$$

- **n_A and n_B are nonces**
- **K_A and K_B are keying materiale**
- **$K = K_A \oplus K_B$**

A good design choice

UNIVERSITÀ DI PISA

- It is always a **good design practice** to assume that a

  1. session key is compromised and that

  2. the adversary holds it as well all the messages that lead to that key establishment (*the protocol run*)