Public Key Encryption

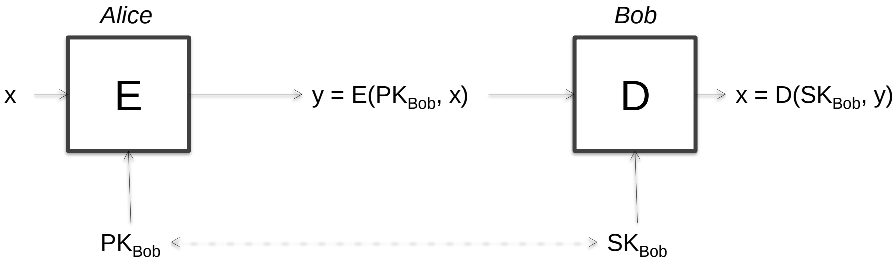
Gianluca Dini
Dept. Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it

Version: 11/03/2025

Public Key Cryptography

INTRODUCTION

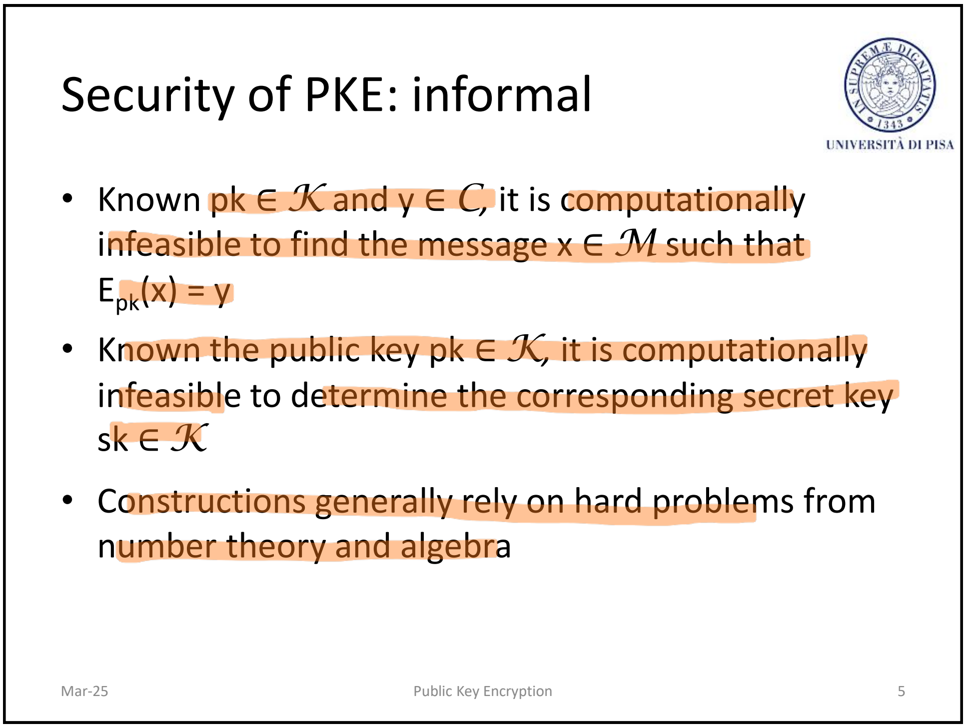
Communication model



- $pubK_{Bob}$: public key
- $privK_{Bob}$: private key
- Alice knows Bob’s public key $pubK_{Bob}$
- Bob keeps secret his own private key $privK_{Bob}$

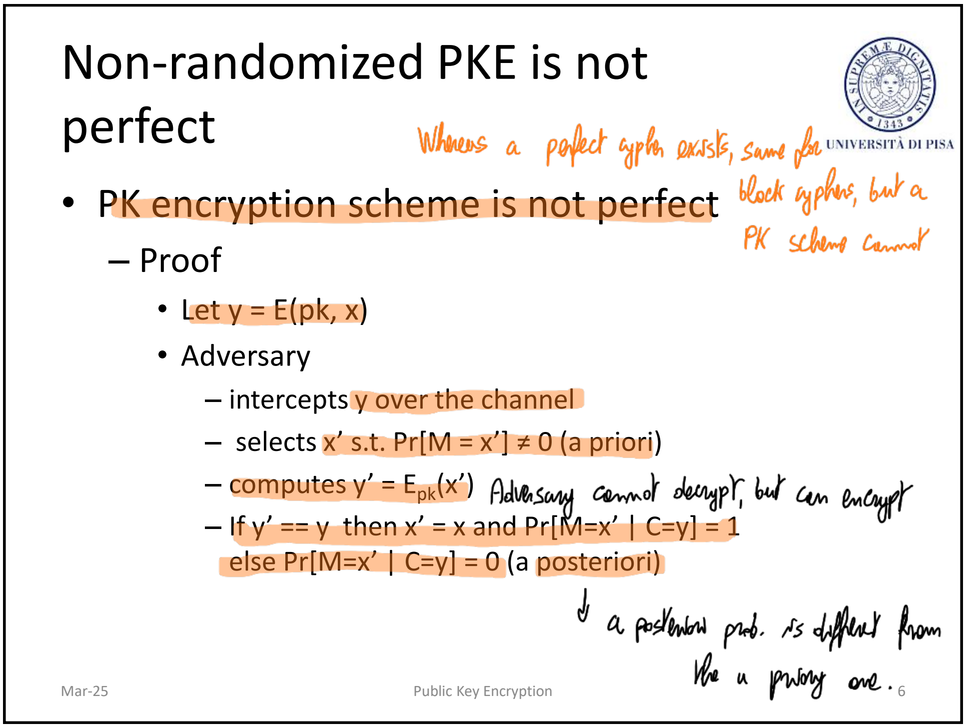
Public key encryption - Definition

- A public key encryption scheme is a triple of algs (G, E, D) s.t.
  - **G** is a randomized alg. for key generation (pk, sk)
  - y = **E**(pk, x) is a (*randomized*) alg. that takes x ∈ $\mathcal{M}$ and outputs y ∈ $\mathcal{C}$
  - x = **D**(sk, y) is deterministic alg. that takes y ∈ $\mathcal{C}$ and outputs x ∈ $\mathcal{M}$
  - fulfills the **Consistency Property**
    - ∀(pk, sk), ∀ x ∈ $\mathcal{M}$, D(sk, E(pk, x)) = x



Security of PKE: informal

- Known $pk \in \mathcal{K}$ and $y \in \mathcal{C}$, it is computationally infeasible to find the message $x \in \mathcal{M}$ such that $E_{pk}(x) = y$
- Known the public key $pk \in \mathcal{K}$, it is computationally infeasible to determine the corresponding secret key $sk \in \mathcal{K}$
- Constructions generally rely on hard problems from number theory and algebra



Non-randomized PKE is not perfect

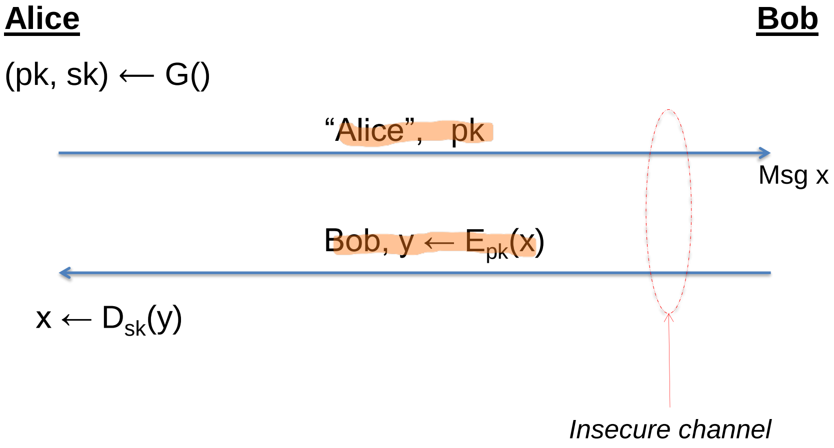
Whereas a perfect cypher exists, same for block cyphers, but a PK scheme cannot

- PK encryption scheme is not perfect
  - Proof
    - Let y = E(pk, x)
    - Adversary
      - intercepts y over the channel
      - selects x' s.t. Pr[M = x'] ≠ 0 (a priori)
      - computes $y' = E_{pk}(x')$ Adversary cannot decrypt, but can encrypt
      - If y' == y then x' = x and Pr[M=x' | C=y] = 1 else Pr[M=x' | C=y] = 0 (a posteriori)

↓ a posteriori prob. is different from the a priori one.

PKE basic protocol

Naive use



Alice **Bob**

(pk, sk) ← G()

“Alice”, pk

Msg x

Bob, $y \leftarrow E_{pk}(x)$

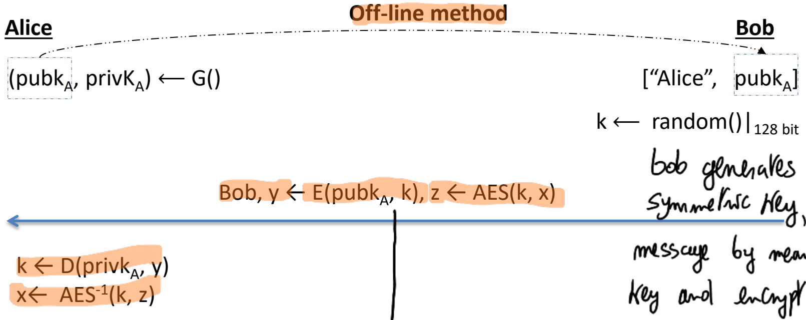
$x \leftarrow D_{sk}(y)$

Insecure channel

Digital envelope

- Public key cryptography is 2-3 orders of magnitude slower than symmetric key cryptography Unconvenient from performances POV
  - Public-key performance can be a more serious bottleneck in constrained devices, e.g., mobile phones or smart cards, or on network servers that have to compute many public-key operations per second
- A digital envelope uses two layers for encryption:
  - Symmetric key encryption is used for message encryption and decryption.
  - Public key encryption is used to send symmetric key to the receiving party

Hybrid protocol: digital envelope



Off-line method

Alice

$(pubk_A, privK_A) \longleftarrow G()$

Bob

$[\text{“Alice”}, \ pubk_A]$

$k \longleftarrow random()|_{128\ bit}$

bob generates a symmetric key, encrypts message by means of key and encrypt key by means of alice key

$Bob,\ y \leftarrow E(pubk_A, k),\ z \leftarrow AES(k, x)$

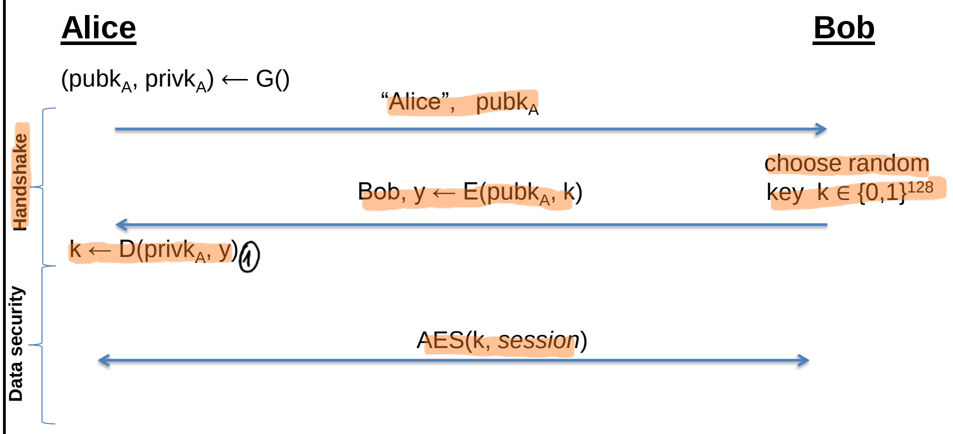
$k \leftarrow D(privk_A, y)$

$x \leftarrow AES^{-1}(k, z)$

PK enc is used on a very small amount of data (16 bytes).

This method doesn't use shared secrets.

Basic key transport protocol



Alice

Bob

$(pubk_A, privk_A) \longleftarrow G()$

“Alice”, $pubk_A$

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

Generalised

$Bob,\ y \longleftarrow E(pubk_A, k)$

$k \longleftarrow D(privk_A, y)$①

$AES(k, \textit{session})$

k: *session key*

Disclaimer: toy protocol!

① Achieved shared secret without sharing a secret in the first place!

Public Key Encryption

PUBLIC KEY CRYPTOGRAPHY

Families of pub key algs

- Built on the common principle of *one-way function*
- A function f() is a one-way function if:
  - y = f(x) is computationally easy, and
  - x = $f^{-1}(y)$ is computationally infeasible
- Two popular one-way functions
  - Integer factorization
  - Discrete logarithm [log in a subset of nrs]

Cyphers are not one way functions

Families of PK Cryptography

- Integer factorization schemes (mid 70s)
  - Most prominent scheme: RSA
- Discrete Logarithm Schemes (mid 70s)
  - Most prominent schemes: DHKE, ElGamal, DSA

→ invented algorithm for DS

→ Diffie Hellman Key establishment

- Elliptic Curves Schemes (mid 80s)
  - EC schemes are a generalization of the Discrete Logarithm algorithm
  - Most prominent schemes: ECDH, ECDSA

Completely broken by quantum attacks

Families of PK Cryptography

- Other schemes
  - Multivariate Quadratic, Lattice
    - They lack maturity
    - Poor performance characteristics
  - Hyperelliptic curve cryptosystems
    - Secure and efficient
    - They have not gained widespread adoption

→ PK schemes based on lattices seem to be resistant to quantum computing

Encryption schemes based on Lattice technology cannot be broken polinomially by quantum computing

ONE of the main usage of public key encryption is key transport. Shared secret without sharing a secret beforehand

Main security mechanisms

- Encryption
  - RSA and ElGamal
- Key establishment
  - Establishing keys over an insecure channel
  - DHKE, RSA key transport
- Non repudiation and message integrity
  - Digital signatures
  - RSA, DSA, ECDSA
- Identification
  - Challenge-response protocol together digital signatures

Key Lenghts and Security Level

- An algorithm has *security level* of n bit, if the best known algorithm requires 2^n steps (to break the encryption scheme)
- Symmetric algorithms with security level of *n* have a ① key of length of *n* bits if best known attack is bruteforce, we require n bits of key to have SL of n (accepted that cypher is secure)
- In asymmetric algorithms, the relationship between security level and cryptographic strengh is no as straightforward

① ASSUMPTION: there exist no better attack than bruteforce

Key Lenghts and Security Level

Algorithm Family	Cryptosystem	Security Level			
		80	128	192	256
Integer Factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete Logarithm	DH, DSA, ElGamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

↑ SkipJack

RULE OF THUMB - The computational complexity of the three public key algorithm families grows roughly with the cube of bit length

For RSA, to have 80b as security level i need 1024bits.
With elliptic curves we only need 160bit

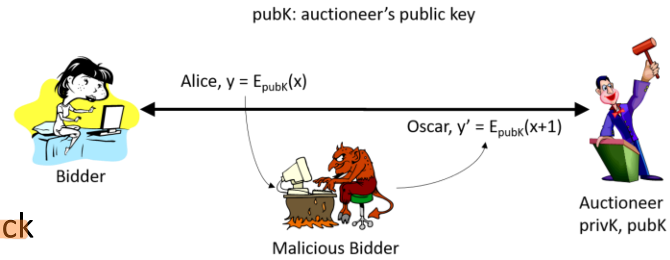
NOTE: the fact that for the same SL we need longer keys is an indication that PKE is slower than SKE.

Elliptic curves are more comparable

Public Key Cryptography

THE NEED FOR ENCRYPTION RANDOMIZATION

Attack against a small plaintex space



①
- The attack
  - Intercept y
  - Try all the possible x's until find x* such that $y = E_{pubK}(x^*)$, then $x^* == x$
  - Let $x' = x^* + 1$
  - Send $y' = E_{pubK}(x')$

Attack against a small plaintex space

- Attack complexity
  - If bid x is an integer, then up to 2^{32} attempts
  - If bid $x \in [x_{min}, x_{max}]$, then #attempts $\ll 2^{32}$

Malicious bidder wants to win the auction by offering the least amount of money. To do that bidder has to know amount of bid of Alice. ①

Trick is that now the adversary can encrypt. No perfect cypher, remember.

NOTE; we are not attacking the keys, just the set of messages.

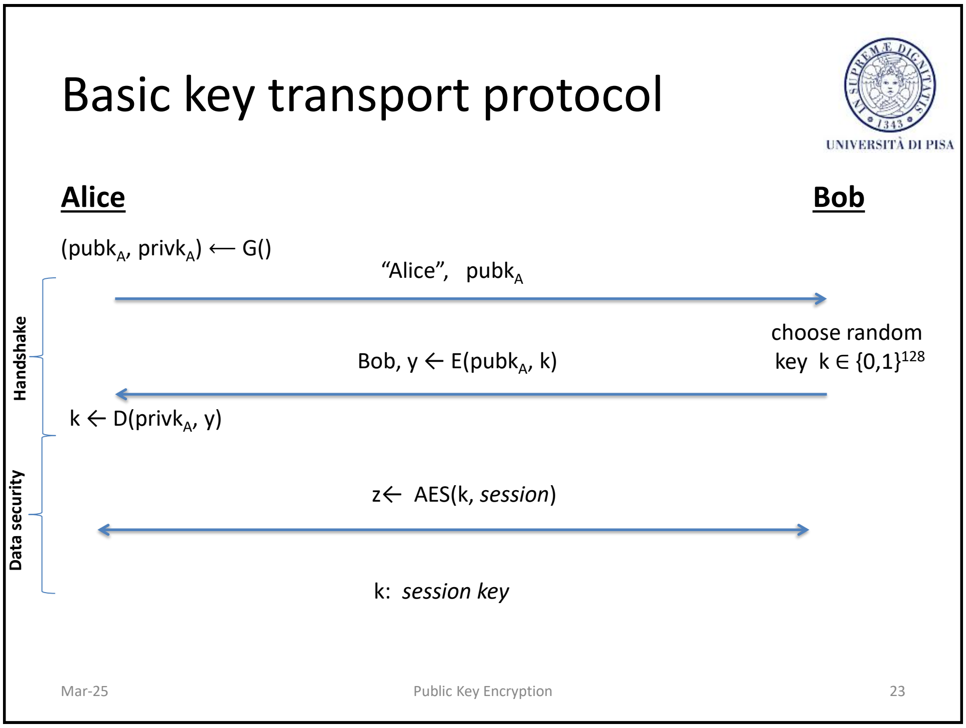
Attack against a small plaintex space

- Countermeasure: salting: introduce randomization
  - Bidder side
    - Salt s ← random()$|_{\text{r-bit}}$ Random sequence of r bits
    - Bid b ← (s, x) s concatenated to salt
    - y = E_{pubK}(b)
  - Auctioneer side
    - (s, x) ← D_{privK}(b) and retain x
  - Adversary
    - Try alle the possible pairs (bid, salt)
    - Attack complexits gets multiplied by 2^r

- I discard the salt

Public Key Cryptography

KEY AUTHENTICATION



Basic key transport protocol

Alice **Bob**

$(pubk_A, privk_A) \longleftarrow G()$

“Alice”, $pubk_A$ →

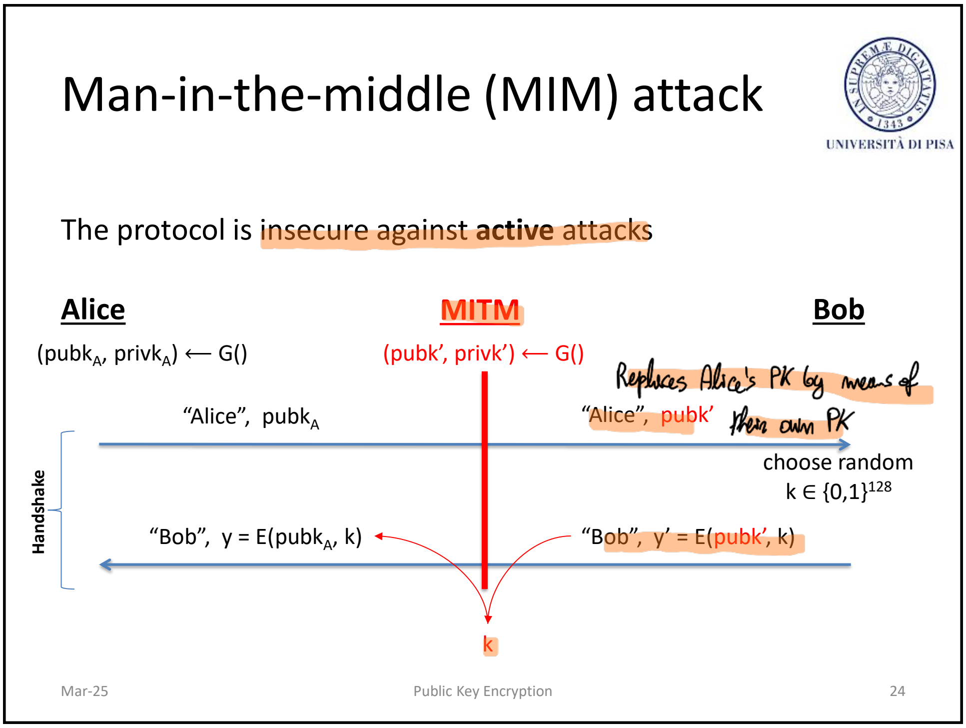
choose random key $k \in \{0,1\}^{128}$

← Bob, $y \leftarrow E(pubk_A, k)$

$k \leftarrow D(privk_A, y)$

↔ $z \leftarrow AES(k, \textit{session})$

k: *session key*



Man-in-the-middle (MIM) attack

The protocol is insecure against **active** attacks

Alice **MITM** **Bob**

$(pubk_A, privk_A) \longleftarrow G()$

$(pubk', privk') \longleftarrow G()$

Replaces Alice's PK by means of their own PK

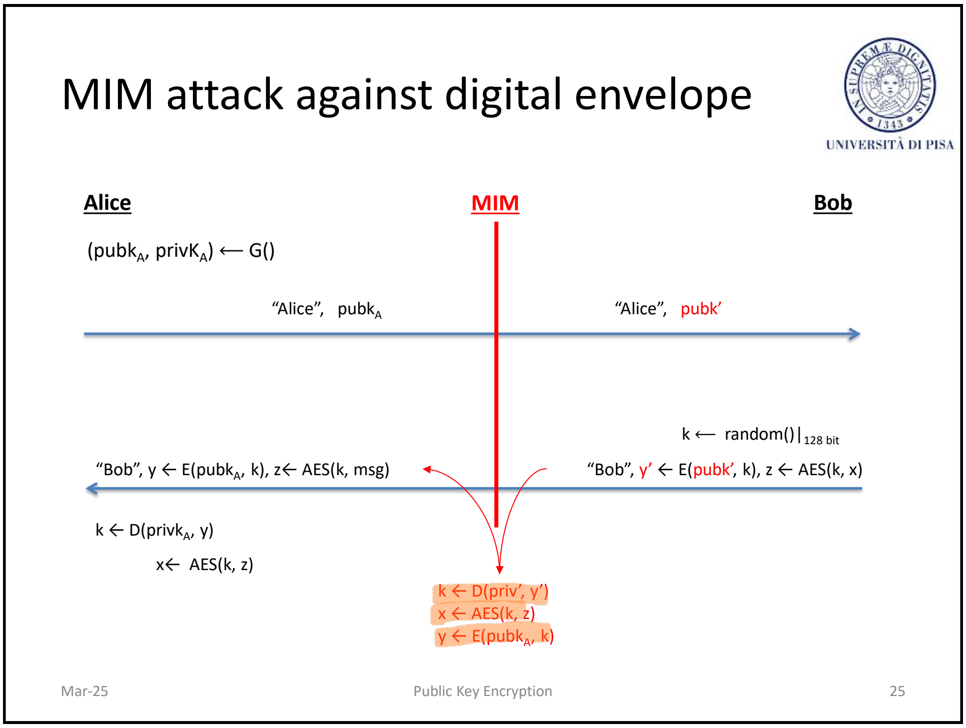
“Alice”, $pubk_A$ → “Alice”, pubk’ →

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

← “Bob”, $y = E(pubk_A, k)$ ← “Bob”, $y' = E(pubk', k)$

k

IF adversary becomes active, PK encryption is not enough to solve the problem of working with a key. Nothing ensures that the public key I get is the one associated to the client. Answer: certificate.



MIM attack against digital envelope

Alice **MIM** **Bob**

$(pubk_A, privK_A) \longleftarrow G()$

"Alice", $pubk_A$

"Alice", pubk'

$k \longleftarrow random()|_{128\ bit}$

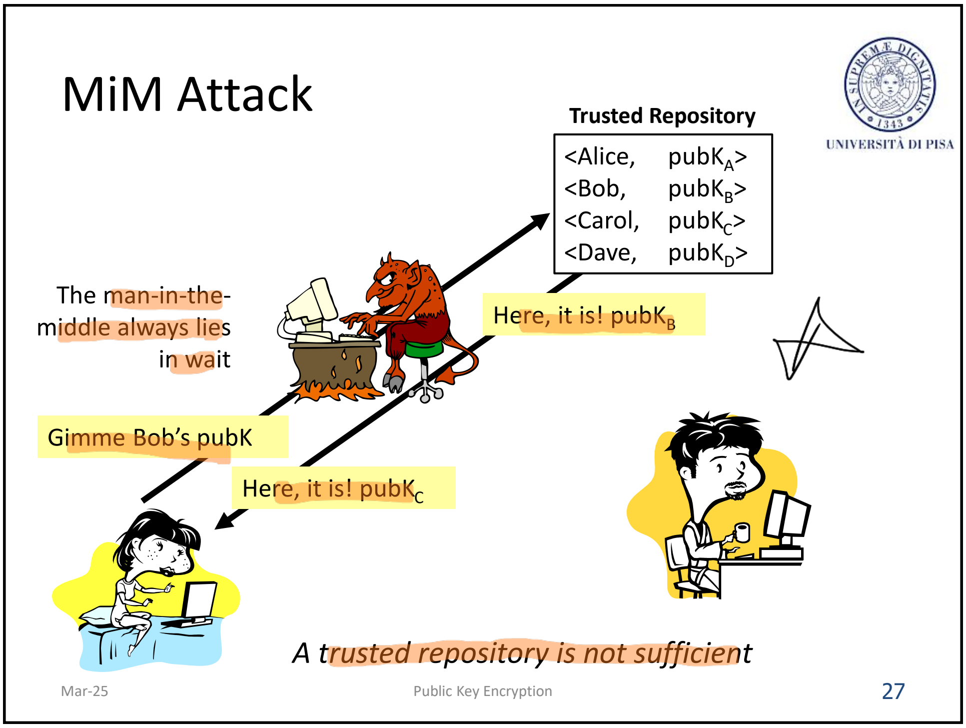
"Bob", $y \leftarrow E(pubk_A, k)$, $z \leftarrow AES(k, msg)$

"Bob", $y' \leftarrow E(pubk', k)$, $z \leftarrow AES(k, x)$

$k \leftarrow D(privk_A, y)$

$x \leftarrow AES(k, z)$

$k \leftarrow D(priv', y')$
$x \leftarrow AES(k, z)$
$y \leftarrow E(pubk_A, k)$



MiM Attack

Trusted Repository

<Alice, $pubK_A$>
<Bob, $pubK_B$>
<Carol, $pubK_C$>
<Dave, $pubK_D$>

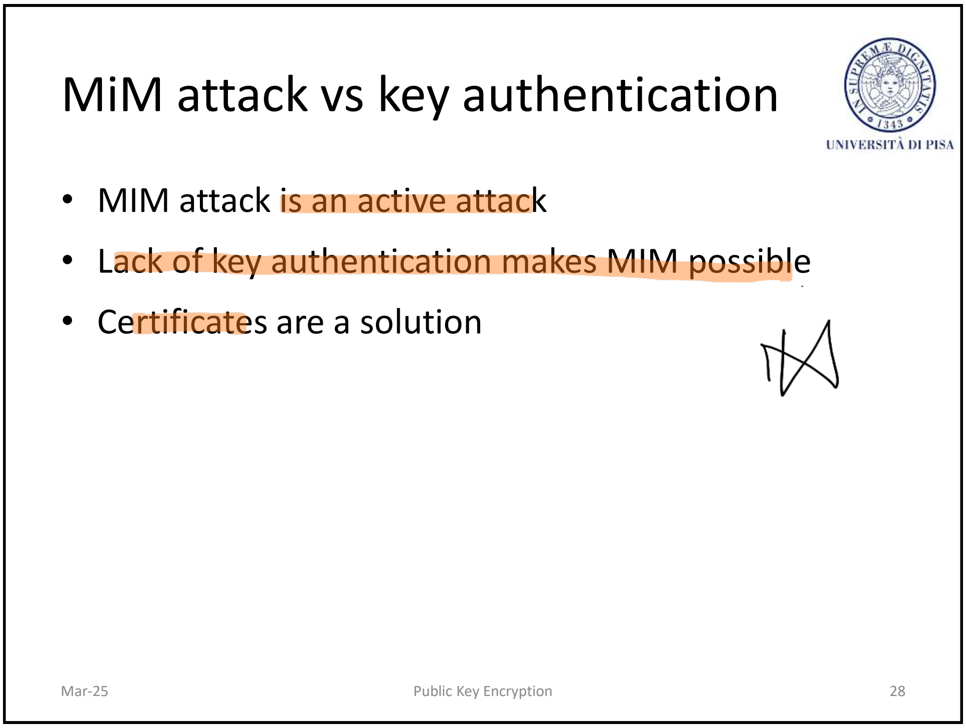
The man-in-the-middle always lies in wait

Here, it is! $pubK_B$

Gimme Bob's pubK

Here, it is! $pubK_C$

A trusted repository is not sufficient



MiM attack vs key authentication

- MIM attack is an active attack
- Lack of key authentication makes MIM possible
- Certificates are a solution