UNIVERSITÄT PASSAU

Fakultät für Informatik und Mathematik

6090: Security of Computer and Embedded Systems

Week 8: Signatures and PKIs

Elif Bilge Kavun

elif.kavun@uni-passau.de

December 07, 2021

This Week’s Outline

- Digital Signatures
- Public-key Infrastructures (PKIs)
- Key Revocation and Recovery

Motivation

"Public-key cryptography was born in May 1975, the child of two problems …

- ***… the problem of key distribution …***
- ***… the problem of signatures …***

…

Whitfield Diffie

The First Ten Years of Public-key Cryptography, Proceedings of the IEEE, 1988

Motivation

“Public-key cryptography was born in <u>May 1975</u>, the child of two problems …

- ***… the problem of key distribution …***
- ***… the problem of signatures …***

…

The discovery consisted not of a solution, but of the recognition that the two problems, each of which seemed unsolvable by definition, could be solved at all and that the solutions to both came in one package.”

Whitfield Diffie

The First Ten Years of Public-key Cryptography, Proceedings of the IEEE, 1988

Motivation

Today:

- To what extent are these problems solved?

Motivation

Today:

- To what extent are these problems solved?

- Large scale PKI
- Sign once – verify often
- High speed, low power
- Improved hash functions
- Quantum era

Motivation

Today:
- To what extent are these problems solved?

- Large scale PKI
- Sign once – verify often
- High speed, low power
- Improved hash functions
- Quantum era

- RSA
- ECC
- PQ-resistant crypto
  - Lattice-based problems
  - Code-based problems
  - …

Motivation

Recommendations

- Read

Whitfield Diffie, *The First Ten Years of Public-key Cryptography*, Proceedings of the IEEE, 1988
https://www.cs.miami.edu/home/burt/manuscripts/crypto_for_intelligence/diffie.pdf

- Listen

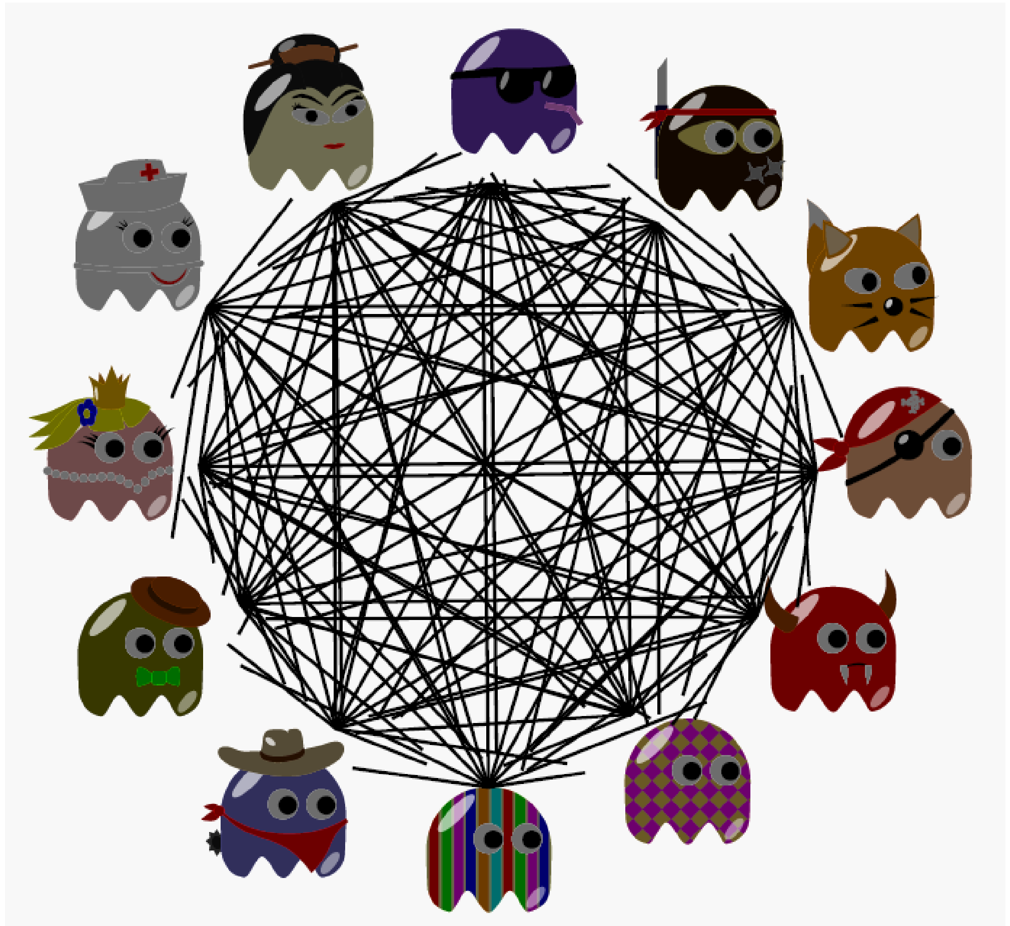
Whitfield Diffie, “Information Security – Before & After Public Key Cryptography”
https://youtu.be/1BJuuUxCaaY

The Key Distribution Problem

- How to provide a strong link between a key and an identity?

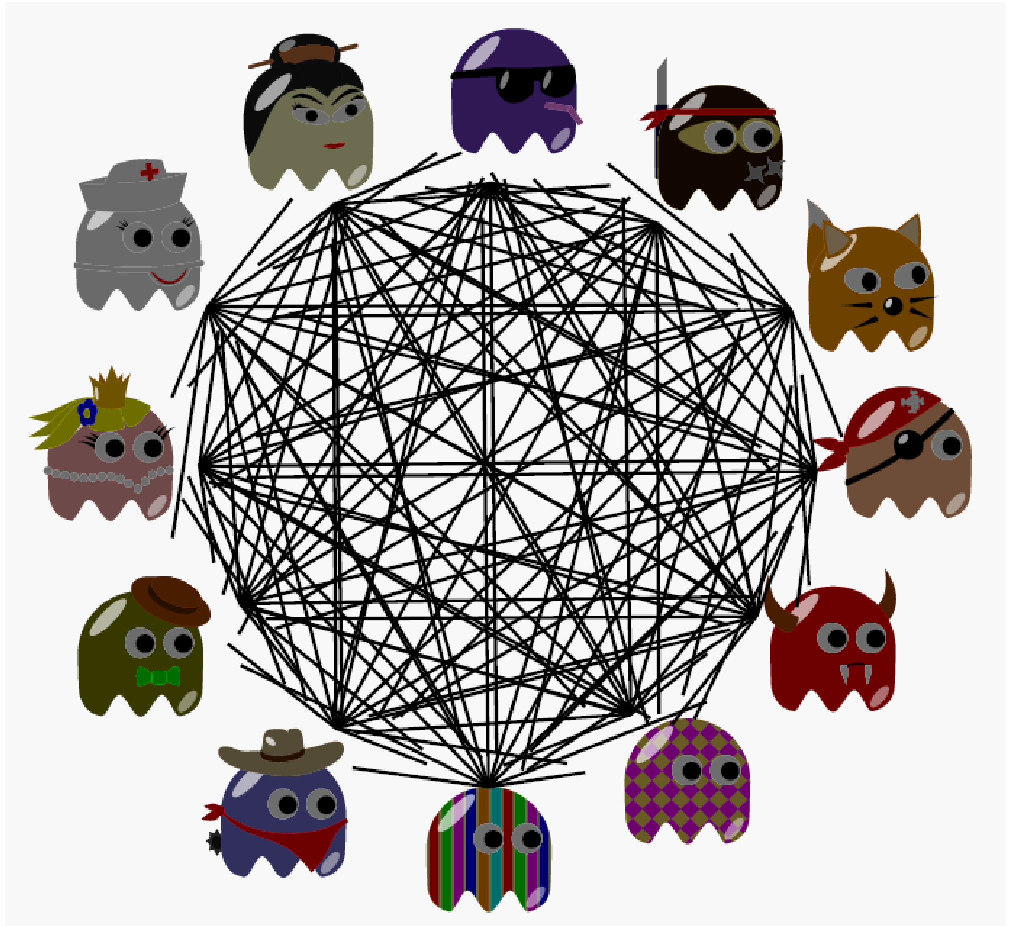
The Key Distribution Problem

- How to provide a strong link between a key and an identity?



The Key Distribution Problem

- How to provide a strong link between a key and an identity?



Symmetric Keys: $\frac{12 * 11}{2}$ = 66 keys

For ***n*** users, we need $\frac{n * (n - 1)}{2}$ symmetric keys

The Key Distribution Problem

- How to provide a strong link between a key and an identity?

The Key Distribution Problem

- How to provide a strong link between a key and an identity?

<u>Asymmetric Keys</u>: 12 keys

The Key Distribution Problem

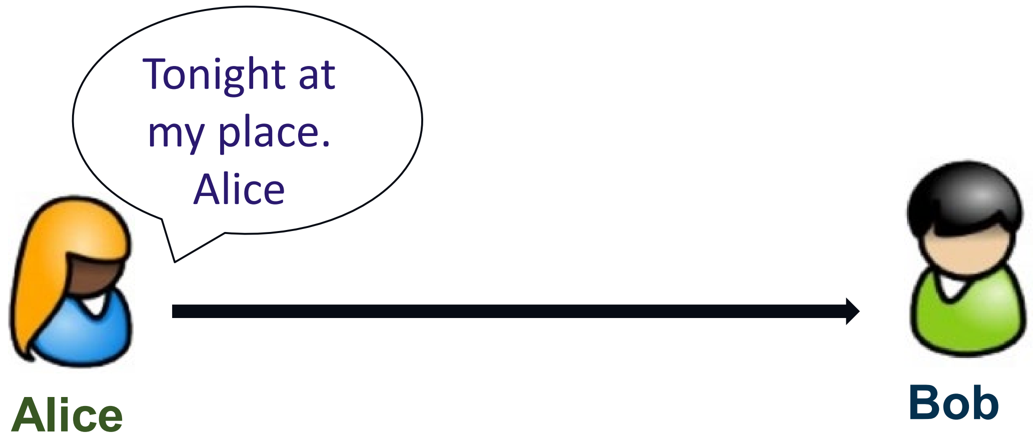
- How to provide a strong link between a key and an identity?

Asymmetric Keys: 12 keys

How to ensure the authenticity of public keys?

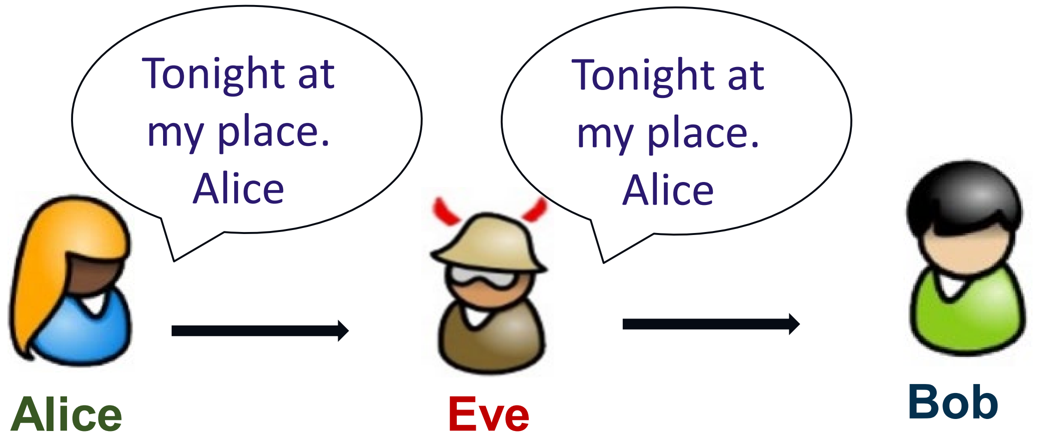
Digital Signatures

- The digital signature problem



Digital Signatures

- The digital signature problem



Digital Signatures

- The digital signature problem
  - **Problem:** Proof of data origin

  How do we know, or prove, that a message originated from a particular entity?

  - Public-key cryptography supports both knowing and proving to others
    - The non-repudiation property: A person (user) cannot deny having performed an operation (transaction)

  Would this be possible using a shared key?

Digital Signature Requirements

- Signature is fundamental in authentication and non-repudiation

- Nomenclature and set-up:
  - $\mathcal{M}$ is set of messages that can be signed
  - $\mathcal{S}$ is set of elements called **signatures**, e.g., *n*-bit strings
  - $S_A : \mathcal{M} \rightarrow \mathcal{S}$ is a **signing transformation** for an entity A, and kept secret by A
  - $V_A : \mathcal{M} \times \mathcal{S} \rightarrow \{\text{true}, \text{false}\}$ is a **verification transformation** for A's signature and is publicly known

- S_A and V_A constitute the main algorithms (transformations) of the digital signature scheme for A

Digital Signature Scheme

- Example:

$S_A(m_0) = s_0$	$V_A(m_0, s_0) = true$	$V_A(m_1, s_0) = false$	$V_A(m_2, s_0) = false$
$S_A(m_1) = s_1$	$V_A(m_0, s_1) = false$	$V_A(m_1, s_1) = true$	$V_A(m_2, s_1) = false$
$S_A(m_2) = s_2$	$V_A(m_0, s_2) = false$	$V_A(m_1, s_2) = false$	$V_A(m_2, s_2) = true$

- **Signing procedure**
  - A creates a signature for $m \in \mathcal{M}$ by computing $s = S_A(m)$ and transmits the pair (m, s)

- **Verification procedure**
  - B verifies A's signature for (m, s) by computing $u = V_A(m, s)$. B accepts A's signature if $u = true$

- **Important!**
  - It is hard for any entity other than A to find, for any $m \in \mathcal{M}$, a valid $s \in \mathcal{S}$, where $V_A(m, s) = true$

Implementing Digital Signatures (1/2)

- Can be based on (reversible) public-key encryption schemes

- Consider $E_e : \mathcal{M} \to \mathcal{C}$ is a public-key transformation

  Moreover, suppose that $\mathcal{M} = \mathcal{C}$

  If D_d is the decryption transformation corresponding to E_e , then since both are permutations

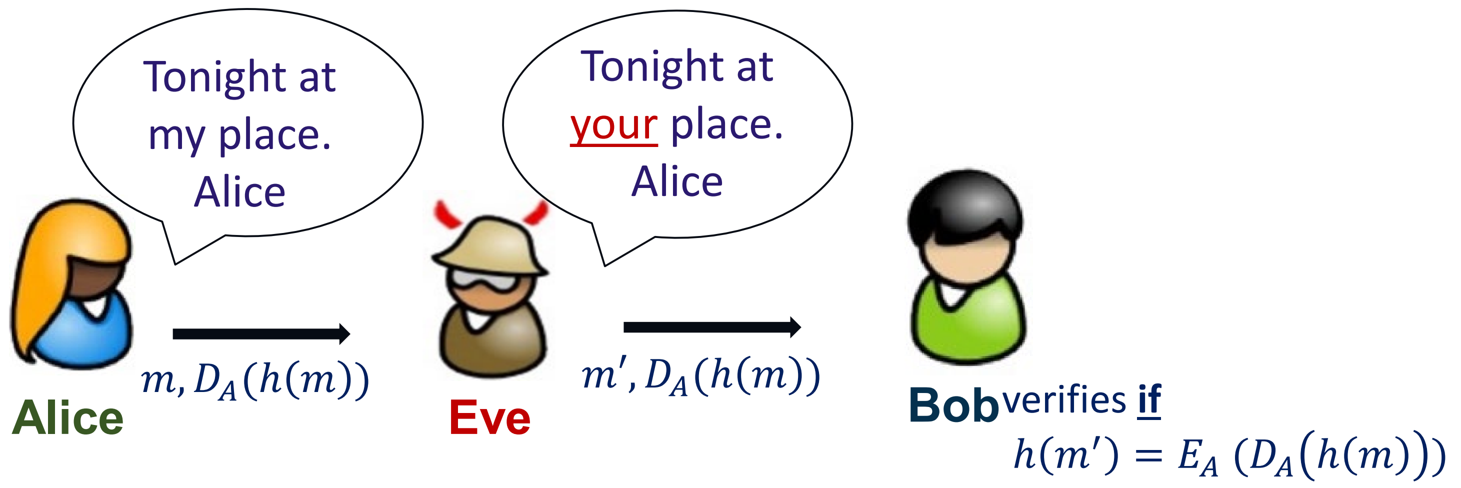
  $$D_d\big(E_e(m)\big) = E_e\big(D_d(m)\big) = m \qquad \text{for all } m \in \mathcal{M}$$

  A public-key encryption scheme of this type is called reversible

- Construction of a digital signature scheme
  1. Let $\mathcal{M}$ and $\mathcal{C}$ be the message and signature space, respectively, with $\mathcal{M} = \mathcal{C}$
  2. Let (e, d) be a key pair for the public-key encryption scheme
  3. Define the signing function S_A to be D_d , i.e., $s = D_d(m)$
  4. Define V_A as follows

  $$V_A\,(m, s) = \begin{cases} \text{true}, & \text{if } E_e(s) = m, \\ \text{false}, & \text{otherwise} \end{cases}$$

Implementing Digital Signatures (2/2)


Tonight at my place. Alice
Tonight at your place. Alice
$m, D_A(h(m))$
$m', D_A(h(m))$
Alice
Eve
Bob verifies if
$h(m') = E_A\,(D_A(h(m)))$

- RSA provides a realization of digital signatures

$$D_d(E_e(m)) = E_e(D_d(m)) = m$$

- To prevent forgery, sign messages with a fixed structure, e.g.,
  - Message names its sender

  OR (more typically)
  - Cryptographic hash signed, sent with the message

Pairs can additionally be encrypted to ensure the confidentiality property

Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

- By definition:

  A public key encryption/signature scheme needs a (key distribution) mechanism

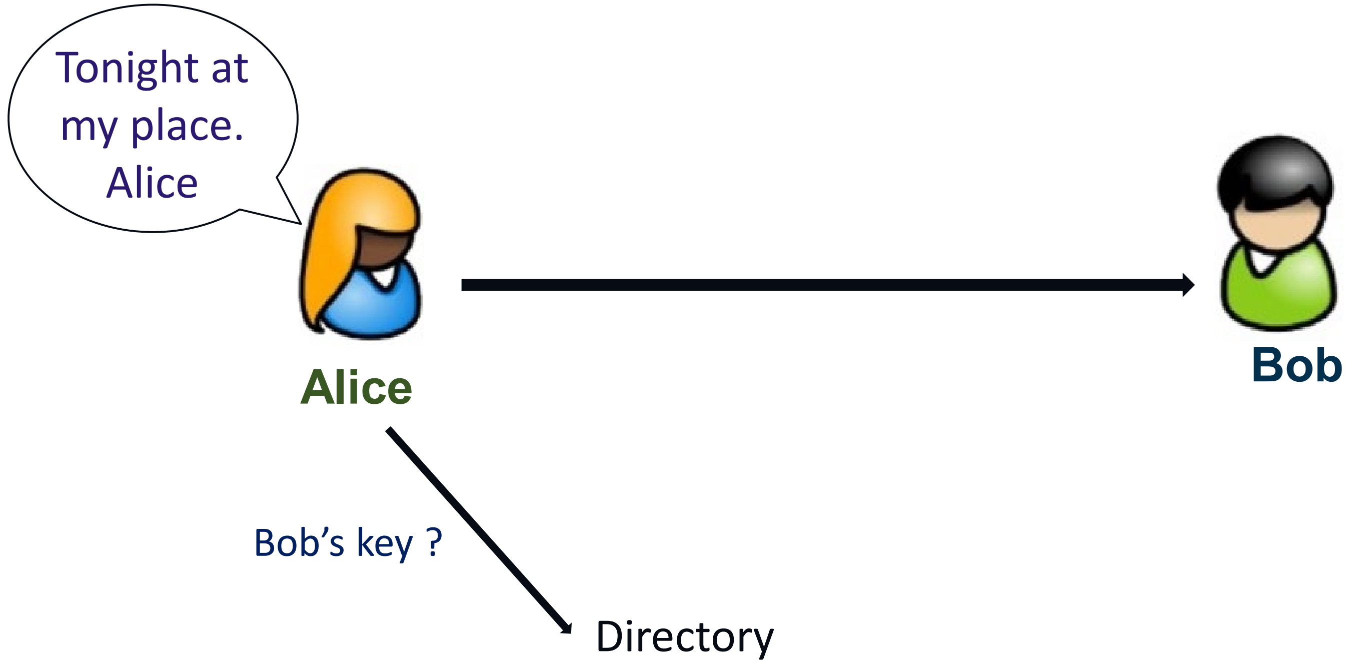
  - It guarantees that the used public-key belongs to the correspondent
  - The key management infrastructure is known as Public-key Infrastructure (PKI)

- If this mechanism is absent, it is possible to carry out a **Man-In-The-Middle** attack (MITM) for any public key encryption/signature scheme

Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

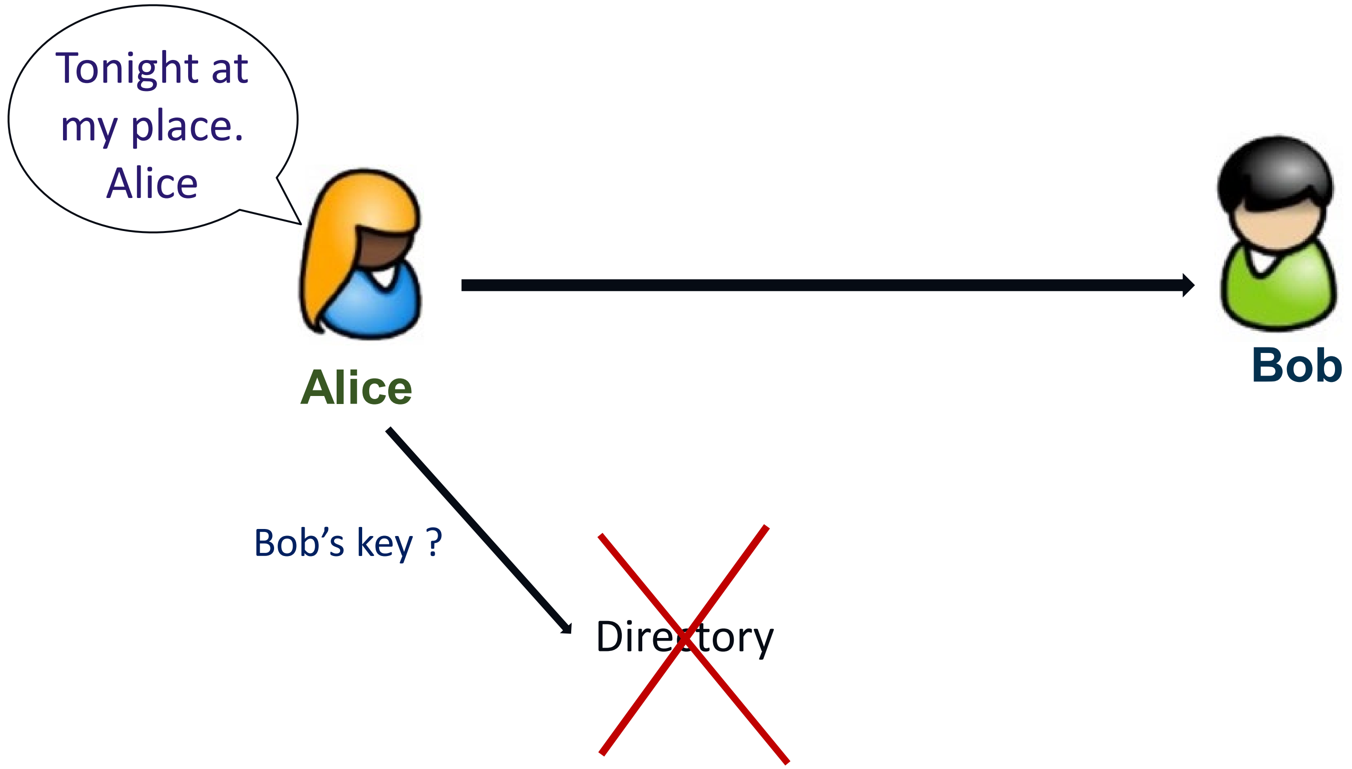
- Man-in-the-Middle Attack



Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

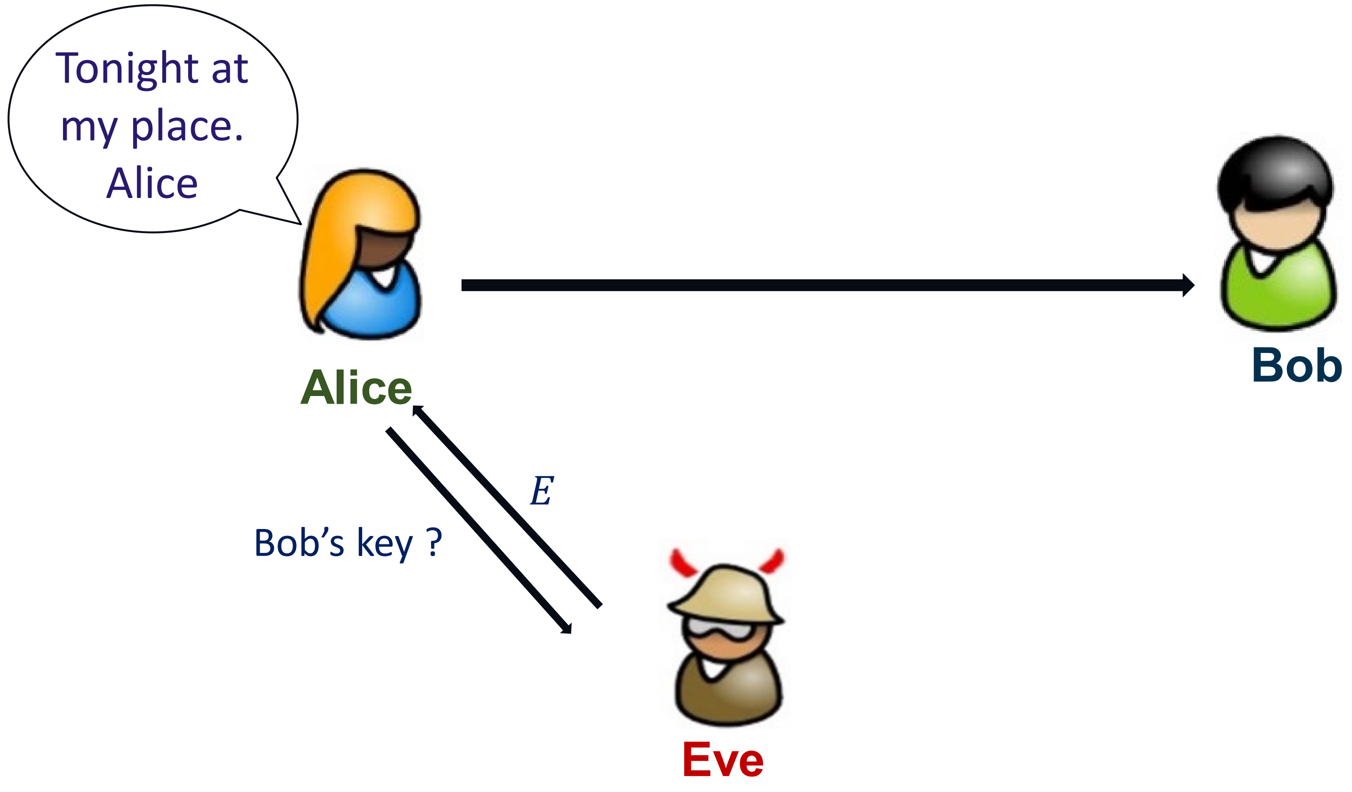
- Man-in-the-Middle Attack



Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

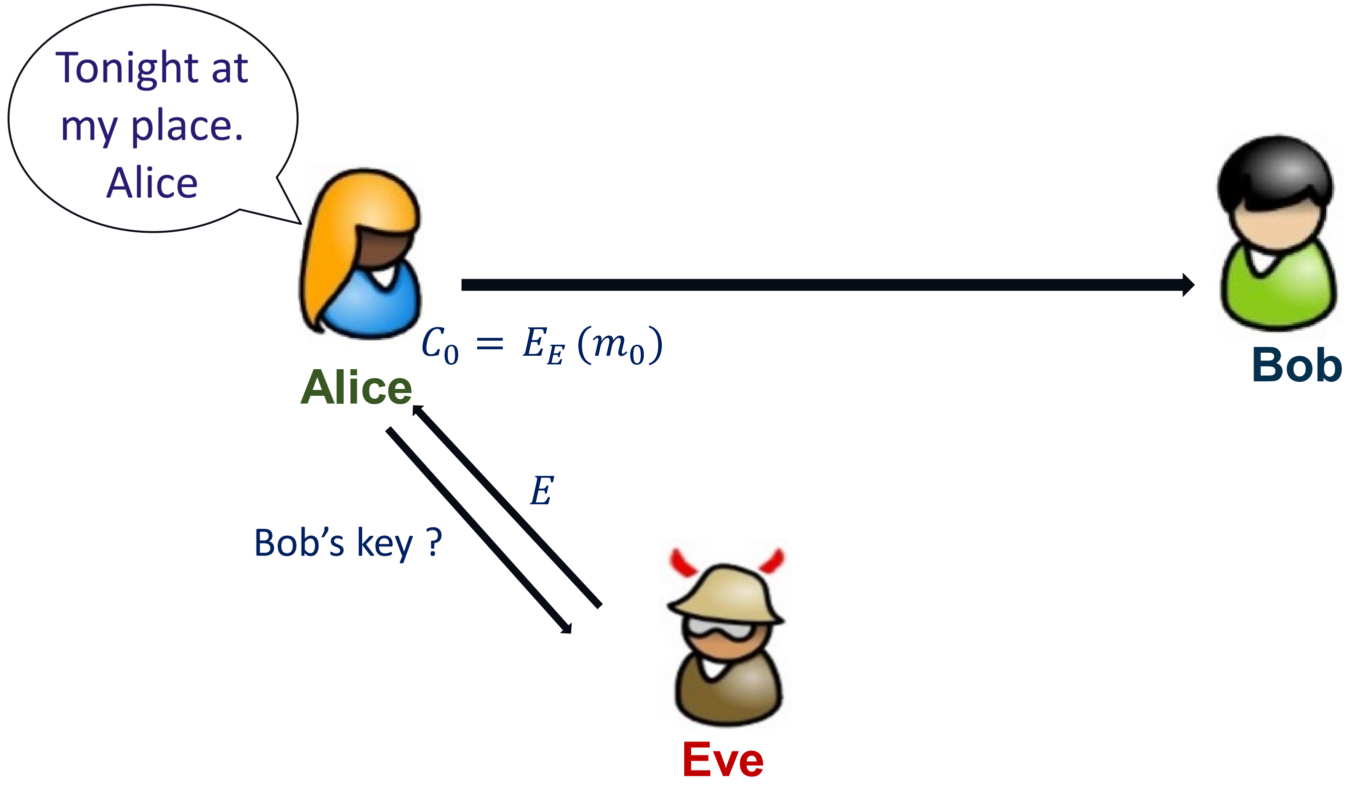
- Man-in-the-Middle Attack



Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

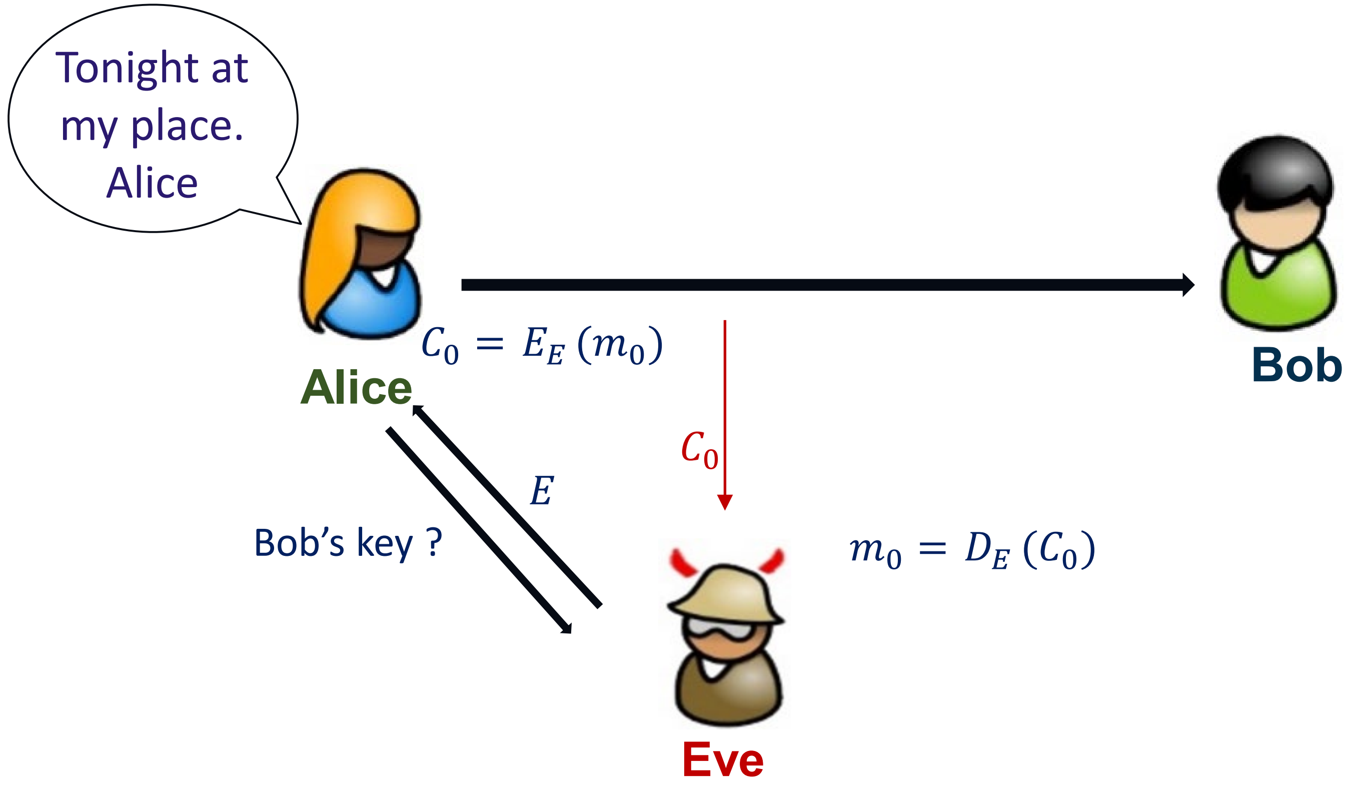
- Man-in-the-Middle Attack



Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

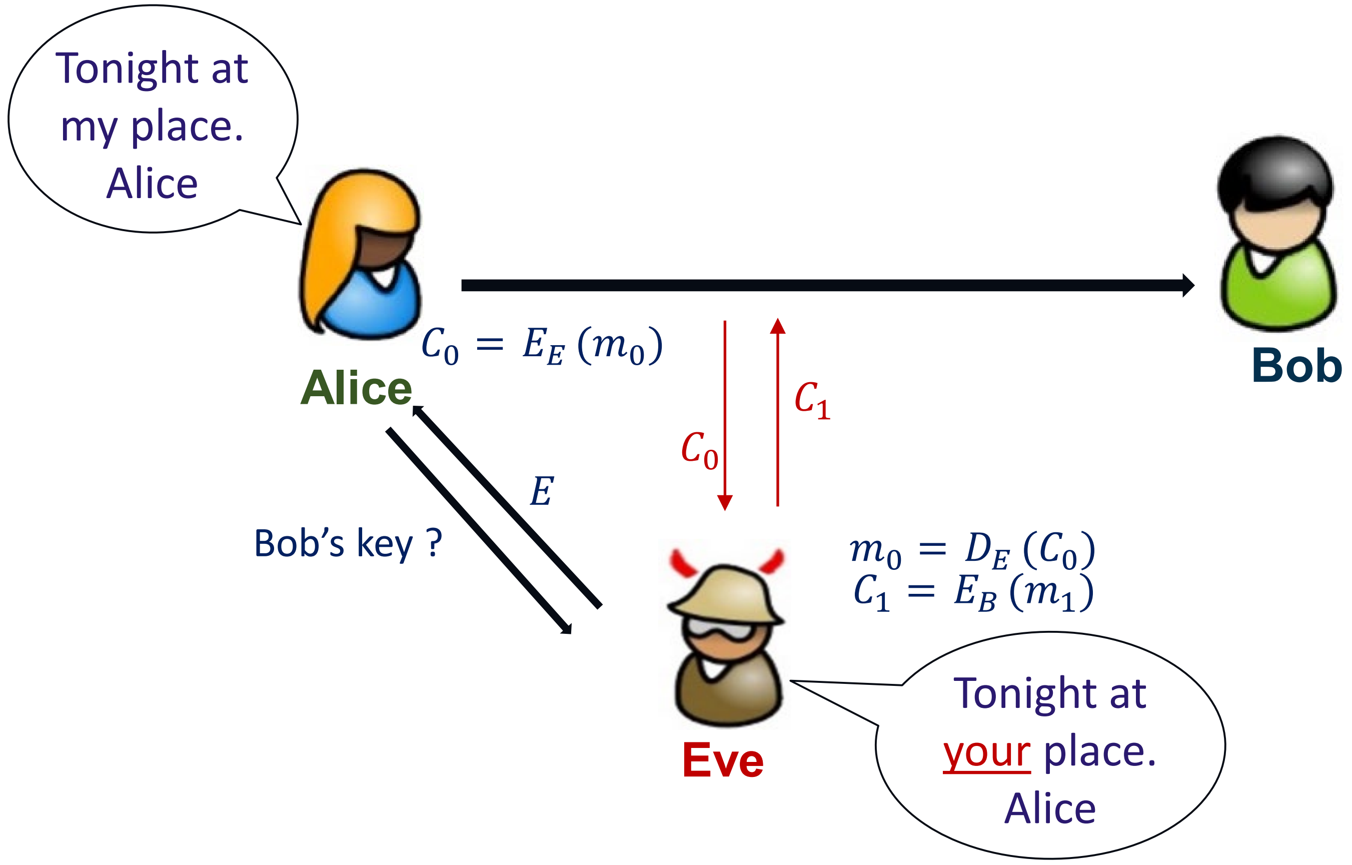
- Man-in-the-Middle Attack



Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

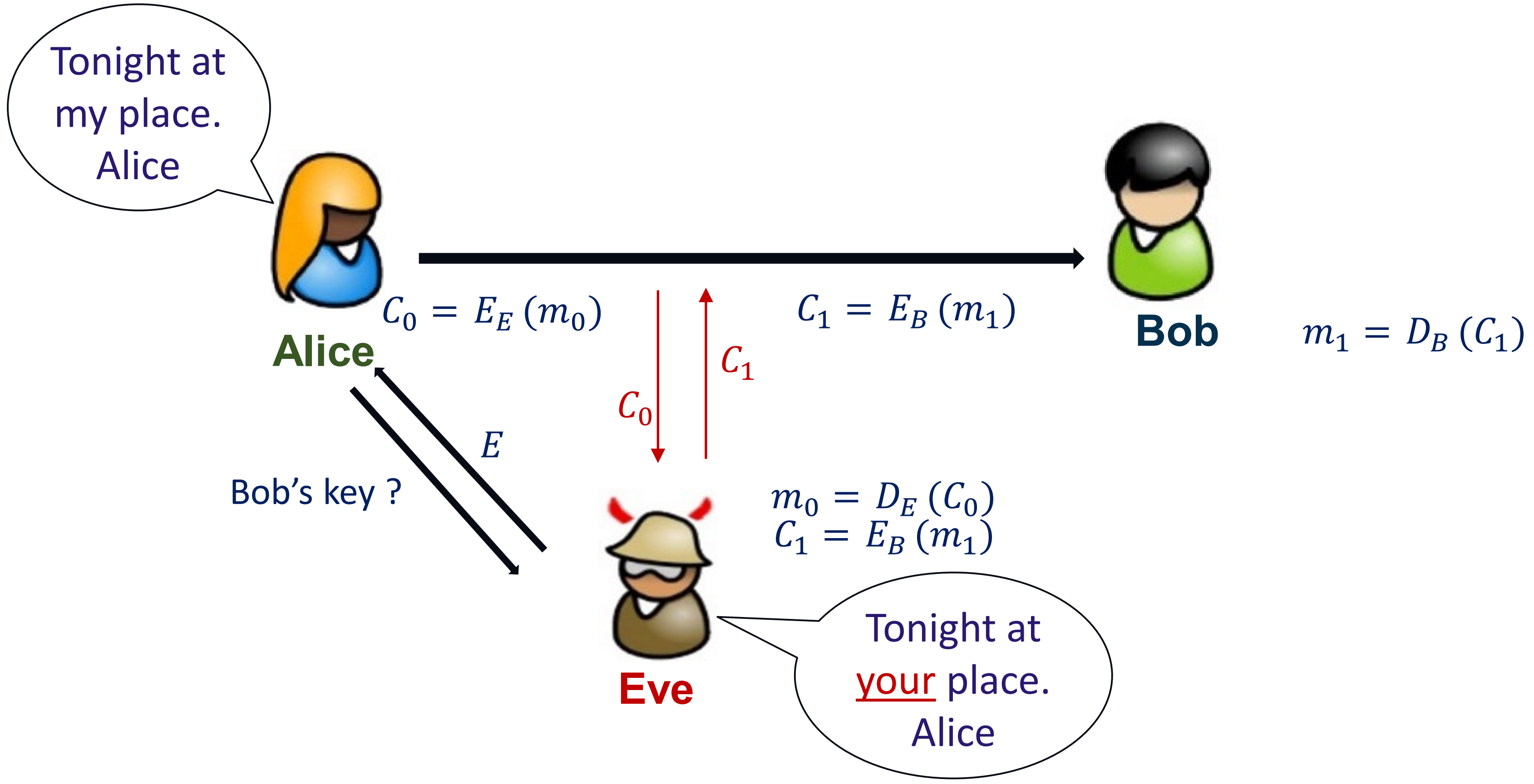
- Man-in-the-Middle Attack



Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

- Man-in-the-Middle Attack



Public Key Infrastructures (PKIs)

The Essence of Public-key Infrastructures (PKIs)

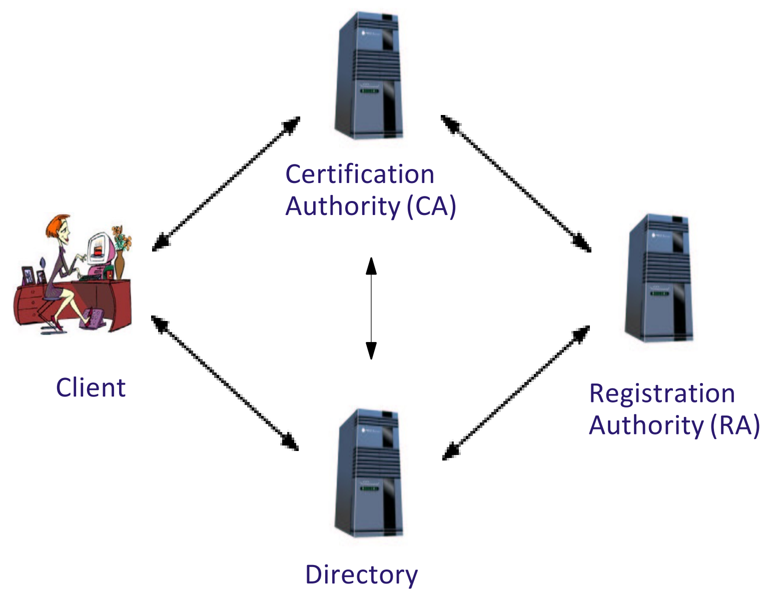
A **PKI** is an infrastructure that allows entities to recognize which public key belongs to whom (i.e., to bind public keys to principals)

- To join the PKI, Alice
  - generates her own public/private key pair
  - takes her public key K_A to a certification authority (CA) that everybody trusts and says "I am Alice and K_A is my public key"
- The CA verifies that Alice is who she says she is, and then signs a digital certificate that states

  "K_A is Alice's public key"
- Then
  - Any entity, e.g., Bob, can check the certificate to obtain Alice's public key K_A and accept it as valid
  - Alice can similarly obtain Bob's public key K_B
- Thus, the CA can help to establish the mutual trust

PKI Services and Components (1/2)

PKI components



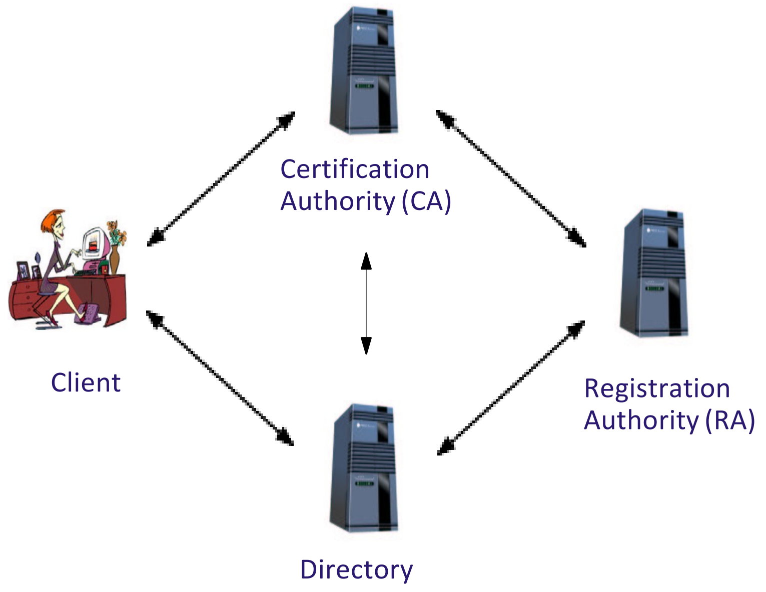
- Certification Authority (CA)
  - Creates certificates and publishes them in directory
  - Maintains a Certificate Revocation List (CRL) in directory
    CRL checked actively by single clients or by validation services
  - Backs-up *certain* keys (for key recovery or escrow)

- Directory
  - Makes user certificates and CRLs available
  - Must identify users uniquely (needs fresh/accurate user data)
  - Must be highly available

- Registration Authority (RA)
  - Manages process of registering users and issuing certificates
  - Ensures proper user identification

- Clients
  - Different uses of a PKI, e.g.
    - Authentication (one-way, two-way, or three-way)
    - Signed documents and transactions

PKI Services and Components (2/2)



PKI services

- Linking public keys to entities (certificates)
- Key life-cycle management (key revocation, recovery, updates)

Certificates

- A certificate is a token that binds an identity to a key

Example:

Issuer CA "Cathy" signs a hash of the identity of the principal to whom the certificate is issued (Alice), the corresponding public key (K_{Alice}), and information such as time of issue or expiration (T):

$$\{h\left(K_{\text{Alice}}, \text{Alice}, T\right)\}_{\{K^{-1}_{\text{Cathy}}\}}$$

- To validate the certificate, a principal entity, Bob, must obtain the issuer's public key and use it to decipher the hash and check the data in the certificate

- To illustrate certificates and certification, let us consider a concrete example: X.509

X.509

- A standard, part of the X.500 series of ITU-T recommendations that defines a framework for authentication services
- The certificate structure (certificate formats and certification validation) and authentication protocols defined in X.509 are used in a variety of contexts, e.g., in IPSEC, SSL/TLS, and S/MIME
- It is based on public-key cryptography (it recommends RSA), hashes, and digital signatures
- The heart of the X.509 scheme is the public-key certificate associated with each user, which is created by the CA and is placed in the directory by the CA or by the user

X.509 Certificate Components (1/2)

- Serial Number (SN) must be unique among the certificates issued by this issuer

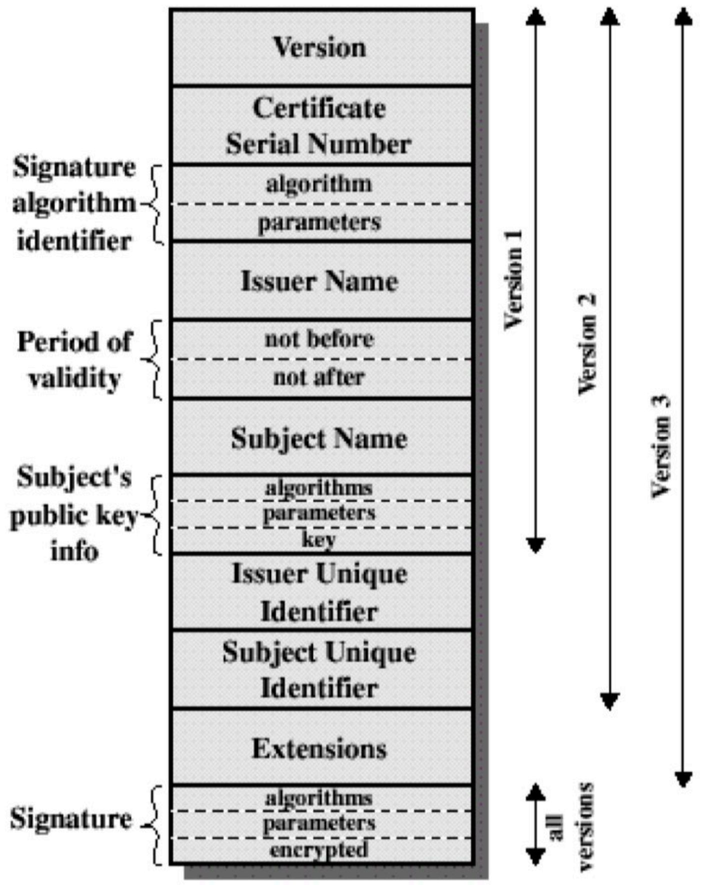
  i.e., pair *(issuer name, serial number)* must be unique

- Signature Algorithm Identifier (AI) of algorithm and any parameters used to sign the certificate

- Issuer name (CA) is X.500 name of CA that created and signed this certificate

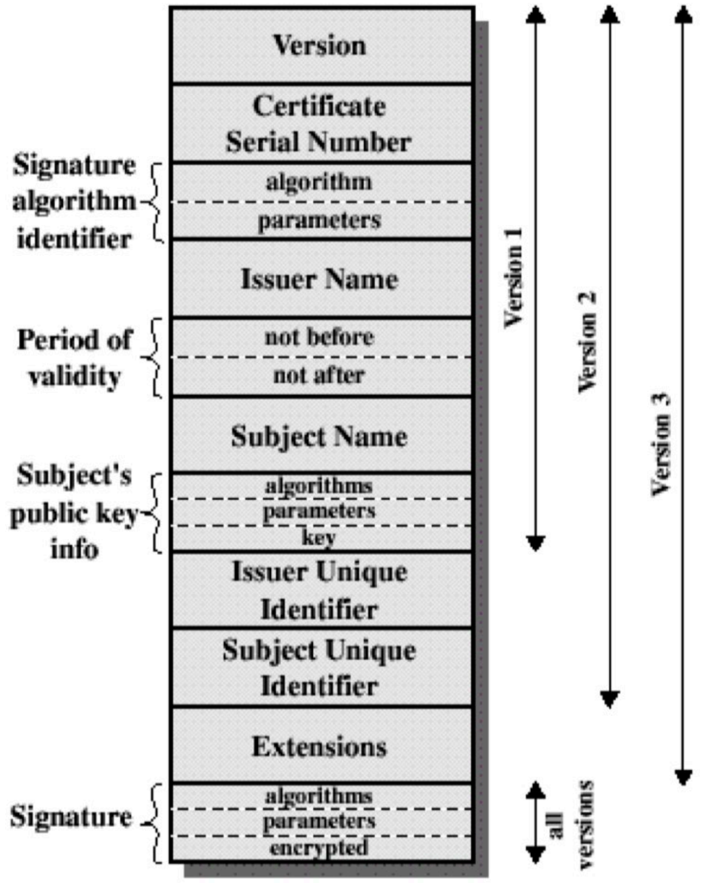
  Optional string issuer unique identifier in the event the

  X.500 name has been reused for different entities



X.509 Certificate Components (2/2)

- Period of validity (T_A)
- Subject name (A) is the name of the user to whom the certificate refers (i.e., the user whose public key is certified) Optional bit string subject unique identifier in the event the X.500 name has been reused for different entities
- Subject public-key info (A_p) identifies the algorithm, its parameters, and the subject's public key
- Signature contains the hash code of the other fields, encrypted with the CA's private key

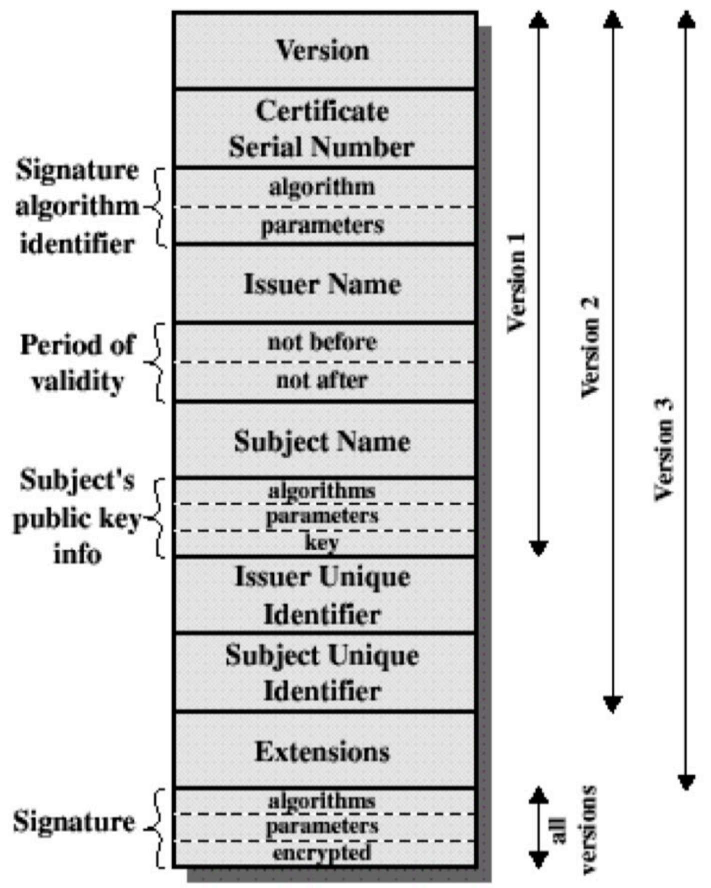


The X.509 Certificate

- The certificate of user A issued (and signed with K_{CA}^{-1}) by CA is

$$CA = \ll A \gg = (V, SN, AI, CA, T_A, A, Ap), \{h(V.SN, AI, CA, T_A, A, Ap)\}_{\{K_{CA}^{-1}\}}$$

- To validate $CA = \ll A \gg$ and verify the user public key that was generated, Bob obtains CA's public key for the particular signature algorithm and deciphers the signature
- Bob then uses the information in the signature field to recompute the hash value from the other fields
  - If it matches the deciphered signature, the signature is valid if the issuer's public key is correct
- Bob then checks the period of validity to ensure that the certificate is current

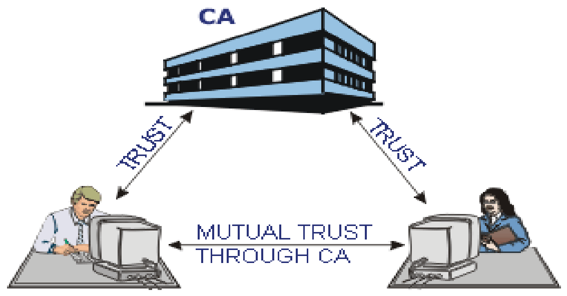


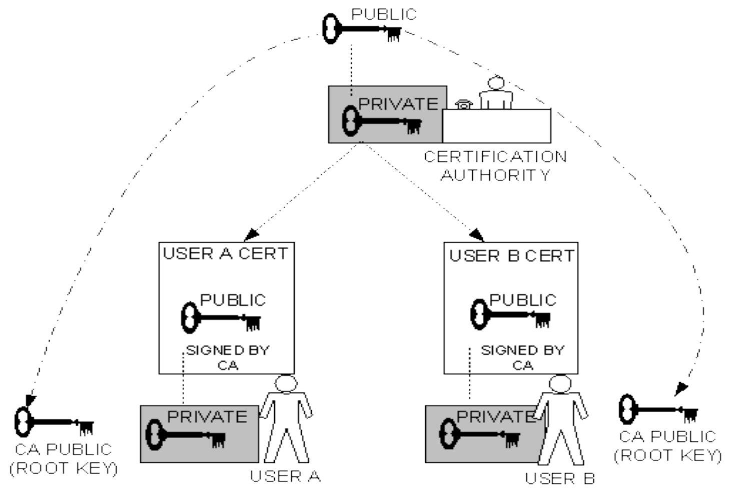
Trust Models

Direct Trust

- If all users subscribe to the same CA, then there is a common trust of that CA

- All user certificates can be placed in the same directory for access by all users



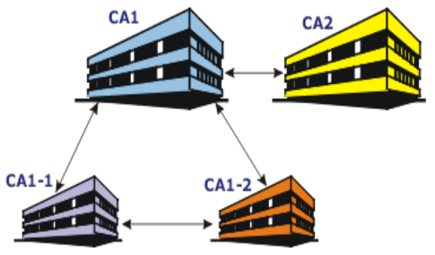


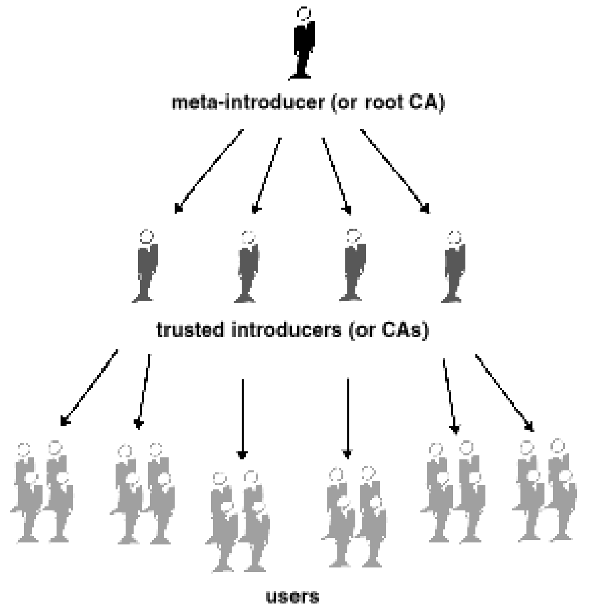
Trust Models

Hierarchical Trust

For a large community of users, it is more practical to have a number of CA's, each of which securely provides its public key to some fraction of the users

- Trust tree
    - Trust extends from a number of root certificates
    - These certificates may certify certificates themselves or they may certify certificates that certify still other certificates down some chain
    - The leaf certificate's validity is verified by tracing backward from its certifier to other certifiers, until a directly trusted root certificate is found





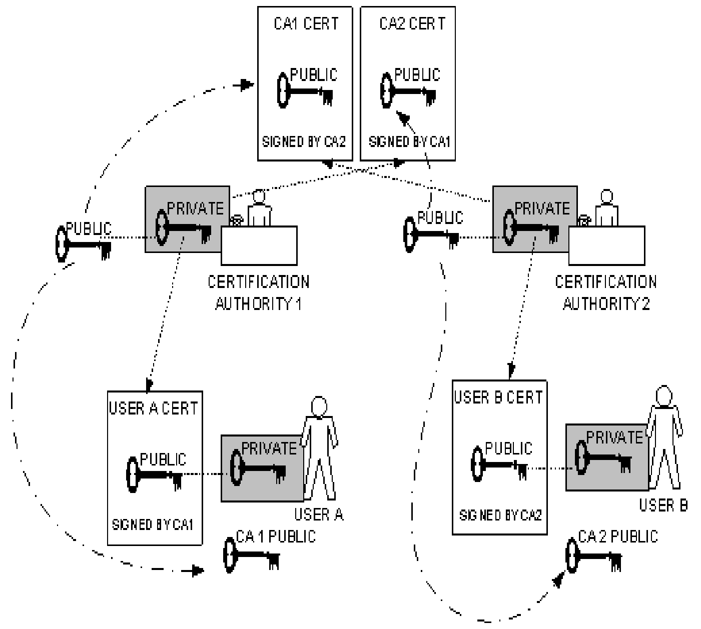
Hierarchical Models and Cross-Certification

- Suppose that A and B have obtained certificates from CAs X_1 and X_2, respectively

  If A does not securely know X_2's public key, then A cannot validated B's certificate

- Cross-certification

  If the CAs have exchanged their own public keys, then A can obtain B's public key by a chain of certificates
  - A obtains, from the directory, the certificate of X_2 signed by X_1
    A can thus get hold of X_2's public key (and verify it by means of X_1's signature on the certificate)
  - A then goes back to the directory and obtains the certificate of B signed by X_2, which A can now verify with the trusted copy of X_2's public key

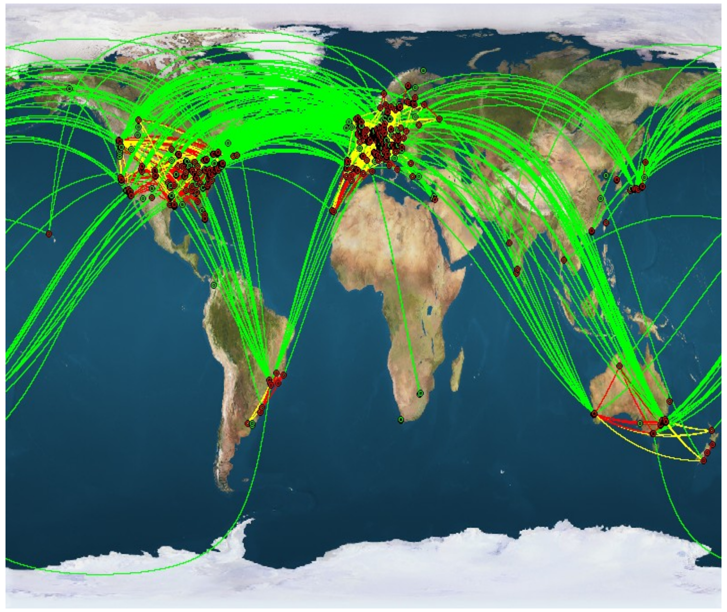


- X.509 suggests arranging CAs in a hierarchy

Trust Models
Web of Trust

- Web of trust
  - Encompasses direct and hierarchical trust
  - Adds the ideas that trust is in the eye of the beholder (which is the real-world view) and that more information is better

- A certificate is trusted directly or trusted in some chain going back to a directly trusted root certificate (the meta-introducer) or by some group of introducers



Web of Trust
PGP/GnuPG (1/3)

- Pretty Good Privacy (PGP), GNU Privacy Guard (GnuPG)

  An encipherment program widely used to provide privacy for e-mail and to sign files digitally

- It uses a certificate-based key management infrastructure for users' public keys

- PGP certificates (and key management) differ from X.509 certificates in several important ways, e.g.,
  - A PGP key may have multiple signatures (even "self-signing")

    Each user creates and signs certificates for the people he/she knows (hence, no need for a central infrastructure)
  - A notion of "trust" is embedded in each signature, and the signatures for a single key may have different levels of trust

    (and the users of a certificate act according to trust level)

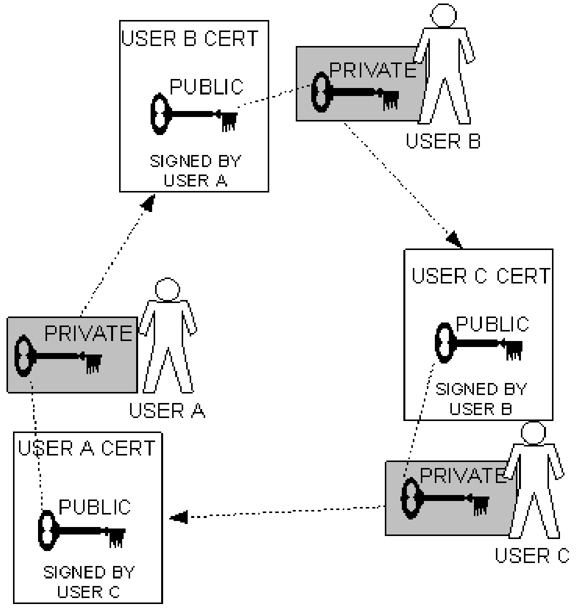
Web of Trust

PGP/GnuPG (2/3)

- In a PGP environment, any user can act as a certifying authority
  - Digital signatures as form of introduction

    When any user signs another's key, he or she becomes an introducer of that key
  - As this process goes on, it establishes a web of trust

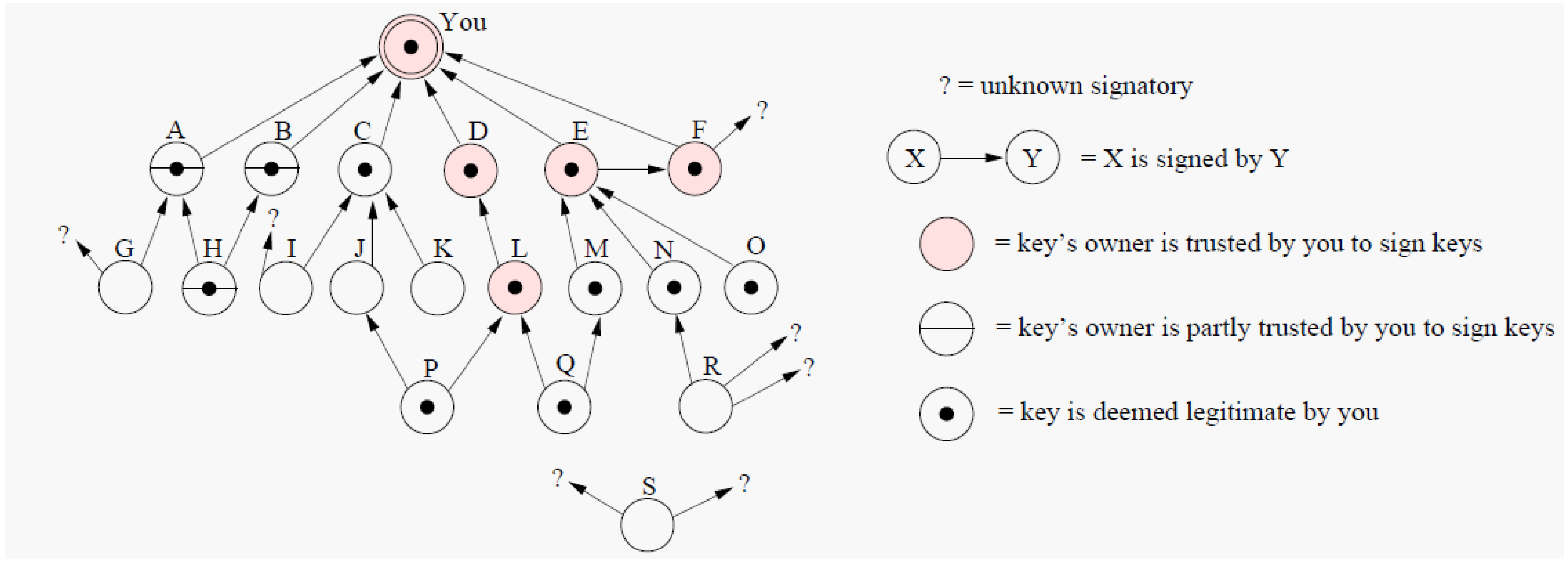
- Any PGP user can validate another user's public key certificate, but such a certificate is only valid to another user if he recognizes the validator as a trusted introducer
  - i.e., you trust my opinion that others' keys are valid only if you consider me to be a trusted introducer
  - Otherwise, my opinion on other keys' validity is unimportant



Web of Trust

PGP/GnuPG (3/3)

- Stored on each user's public keyring are indicators of
  - whether or not the user considers a particular key to be valid,
  - the level of trust the user places on the key that the key's owner can serve as certifier of others' keys
- You indicate on your copy of my key, whether you think my judgement counts
  - **A reputation system:** Certain people are reputed to give good signatures and people trust them to attest to other keys' validity


You
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
?
? = unknown signatory
X
Y
= X is signed by Y
= key's owner is trusted by you to sign keys
= key's owner is partly trusted by you to sign keys
= key is deemed legitimate by you

PKI – Key/Certificate Revocation

- Certificate Revocation List (CRL) signed and maintained by CA
    - Posted on the directory
    - Either clients check themselves actively (also with local caches), or use validation service that collects and checks CRLs centrally
    - Each CA maintains a list of all revoked but not expired certificates issued by that CA (both to users and to other CAs)

- Reasons for revocation
    - The user's private key is assumed to be compromised
    - The user is no longer certified by the CA
    - The CA's certificate is assumed to be compromised

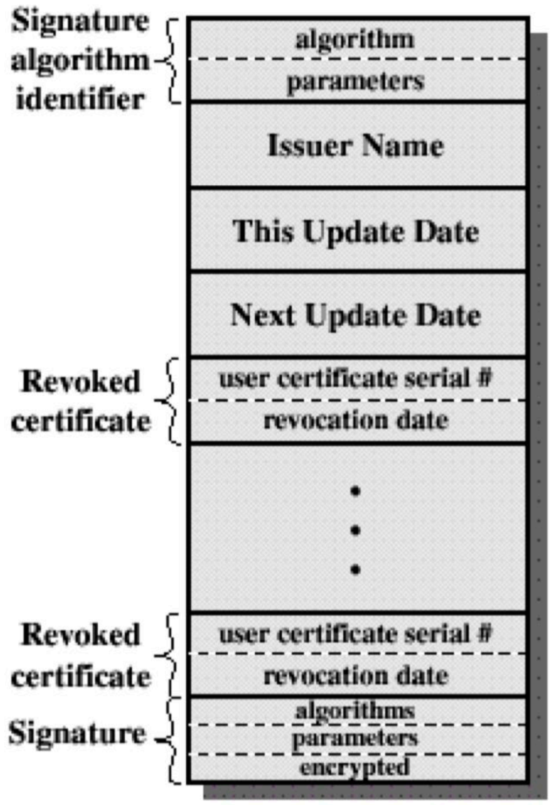
- X.509
    - Each certificate includes a period of validity
    - Typically, a new certificate is issued before the old one expires

X.509 Certificate Revocation

Each CRL includes

- The issuer's name
- The date the CRL was created
- The date the next CRL is scheduled to be issued
- An entry for each revoked certificate

CRL needs to be consulted for each certificate validation



PKI – Key Recovery

- How can one recover a key that is lost, or if the people who know it are unable or unwilling to reveal it?
    - Important, e.g., for keys belonging to roles

PKI – Key Recovery

- How can one recover a key that is lost, or if the people who know it are unable or unwilling to reveal it?
    - Important, e.g., for keys belonging to roles

- Three alternatives
    - The key is weak
    - The cryptosystem is weak
    - A copy of the key can be placed somewhere

- A key escrow system is a system in which a third party can recover a cryptographic key
    - For business (e.g. recovery of backup keys)
    - Law enforcement (key disclosure – recovery of keys used to encipher communications to which an authority requires access, such as enciphered letters or telephone messages)

Summary

Digital signature schemes

- Provide
  - Authentication
  - Non-repudiation

- Help to solve the “key distribution” problem

- Can be implemented using reversible public-key crypto systems (e.g., RSA)

Reading List

- Ross J. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001.
    - The complete book is available at: http://www.cl.cam.ac.uk/~rja14/book.html
- Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 5th edition, 2001.
    - The complete book is available at: http://cacr.uwaterloo.ca/hac/
- Bruce Schneier. Applied Cryptography. John Wiley & Sons, Inc., 2nd edition, 1996.
- Whitfield Diffie, The First Ten Years of Public-key Cryptography, Proceedings of the IEEE, 1988.
    - The paper is available at: https://www.cs.miami.edu/home/burt/manuscripts/crypto_for_intelligence/diffie.pdf

Thanks for your attention!

- Any questions or remarks?