Lecture 1 Computing Lab II CS 252A

Sandeep K. Shukla

Indian Institute of Technology Kanpur

Acknowledgements

- Dan Boneh (Stanford University)
- John C. Mitchell (Stanford University)
- Nicolai Zeldovich (MIT)
- Jungmin Park (Virginia Tech)
- Patrick Schaumont (Virginia Tech)
- C. Edward Chow
- Arun Hodigere
- Mike Freedman, Princeton University
- Scott Midkiff, Virginia Tech
- Insup Lee, University of Pennsylvania
- Web Resources

Acknowledgement

- Mike Freedman, Princeton University
- Scott Midkiff, Virginia Tech
- Insup Lee, University of Pennsylvania

- Material from COS 461 Course during Spring 2014 at Princeton University and from EMTM 553 in Spring 2001 at Upenn.

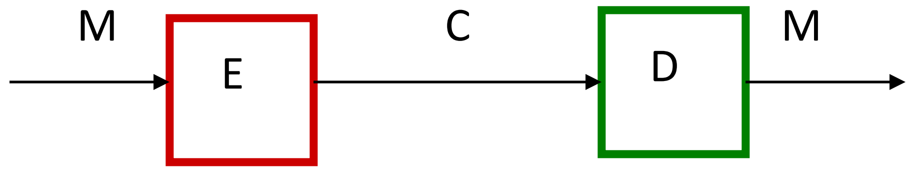
Outline

- How cryptography works
- Secrete key cryptography
- Public key cryptography
- Digital signature
- Message digest
- Distribution of public keys

Cryptography: Basic Terminology

- Plaintext (or cleartext)
  - The message.
  - Denoted by M or P.
- Encryption (encipher)
  - Encoding of message.
  - Denoted by E.
- Ciphertext
  - Encrypted message.
  - Denoted by C.
- Decryption (decipher)
  - decoding of ciphertext
  - denoted by D.

Encryption and Decryption



The following identity must hold true:

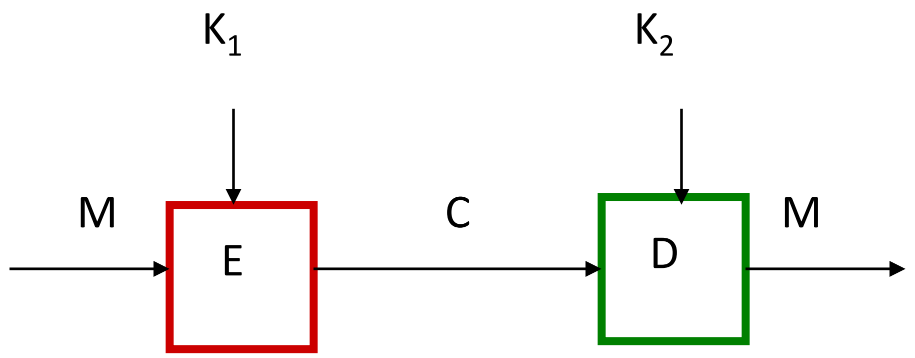
D(C) = M, where C = E(M)

M = D(E(M))

Cryptography: Algorithms and Keys

- A method of encryption and decryption is called a **cipher.**
- Generally there are two related functions: one for encryption and other for decryption.
- Some cryptographic methods rely on the secrecy of the algorithms.
- Such methods are mostly of historical interest these days.
- All modern algorithms use a **key** to control encryption and decryption.
- Encryption key may be different from decryption key.

Key Based Encryption/Decryption



Symmetric Case: both keys are the same or derivable from each other

$K_1 = K_2$.

Asymmetric Case: keys are different and not derivable from each other

$K_1 \; != K_2$

1. Secrete Key Cryptography

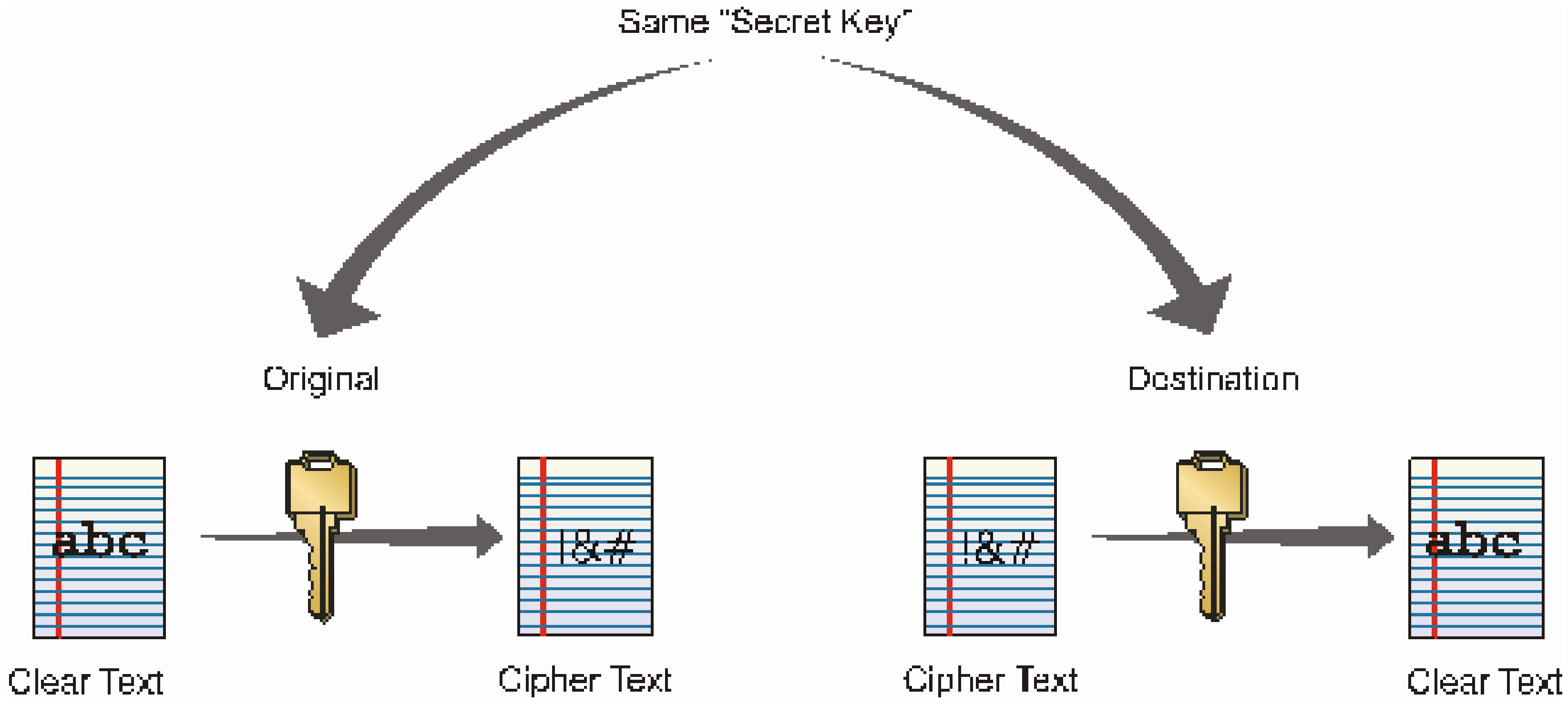
$$\begin{tikzcd}
 & K \arrow[d] & K \arrow[d] & \\
S \arrow[r, "M"] & \boxed{E} \arrow[r, "C"] & \boxed{D} \arrow[r, "M"] & R
\end{tikzcd}$$

K is the secret key shared by both the sender (S) and receiver (R).

Secrete Key Cryptography

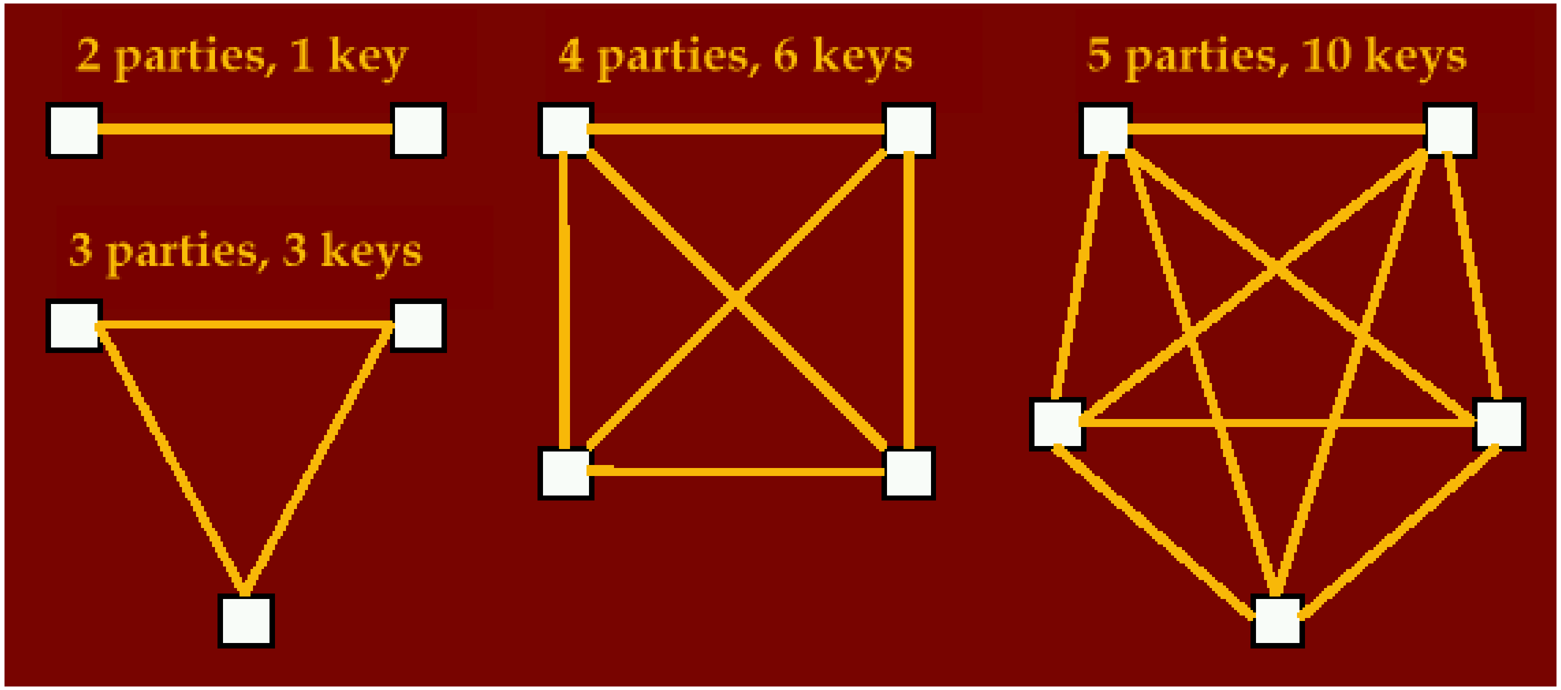
- Also called symmetric or single-key algorithms.
- The encryption and the decryption key are the same.
- Techniques based on a combination of substitution and permutation.
- Stream ciphers: operate on single bit or byte.
- Block ciphers: operate on blocks (typically 64 bits)
- Advantage: simple, fast.
- Disadvantage: ***key exchange, key management***.
- Examples: DES,RC4, IDEA, Blowfish, AES, etc.

Private Key Cryptosystem (Symmetric)



Symmetric Key - Issues

Key management, keys required = $(p*(p-1))/2$ or:



Secrete Key Assurances

- Confidentiality
  - is assurance that only owners of a shared secrete key can decrypt a message that has been encrypted with the shared secrete key
- Authentication
  - is assurance of the identify of the person at the other end of the line (use challenge and response protocols)
- Integrity
  - is assurance that a message has not been changed during transit and is also called message authentication (use message fingerprint)
- Non-repudiation
  - is assurance that the sender cannot deny a file was sent. This cannot be done with secrete key alone (need trusted third party or public key technology)

Example: non-repudiation

- Scenario 1:
  - Alice sends a stock buy request to Bob
  - Bob does not buy and claims that he never received the request
- Scenario 2:
  - Alice sends a stock buy request to Bob
  - Bob sends back an acknowledge message
  - Again, Bob does not buy and claims that he never received it
  - Alice presents the ack message as proof
- Can she prove that the ack message was created by him?

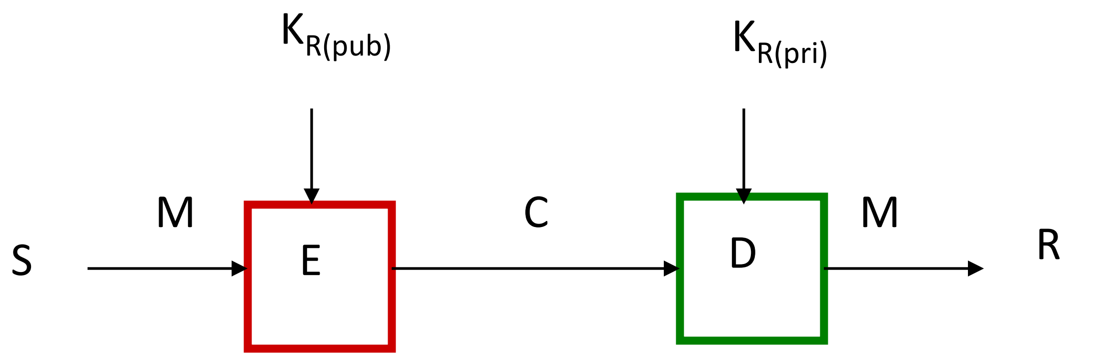
DES (Data Encryption Standard)

- In 1972, NIST (National Institute of Standards and Technology) decide to assist the development of a secure cryptographic method.
- In 1974, it settled on DES, which was submitted by IBM and is the Data Encryption Algorithm developed by Horst Feistel.
- NSA shortened the secrete key to 56 bits from 128 bits originally proposed by IBM.
- Initially intended for 10 years. DES reviewed in 1983, 1987, 1993.
- In 1997, NIST solicited candidates for a new secrete key encryption standard, Advanced Encryption Standard (AES).
- In Oct 2000, NIST selected Rijndael. (www.nist.gov/AES)

Cycling through DES keys

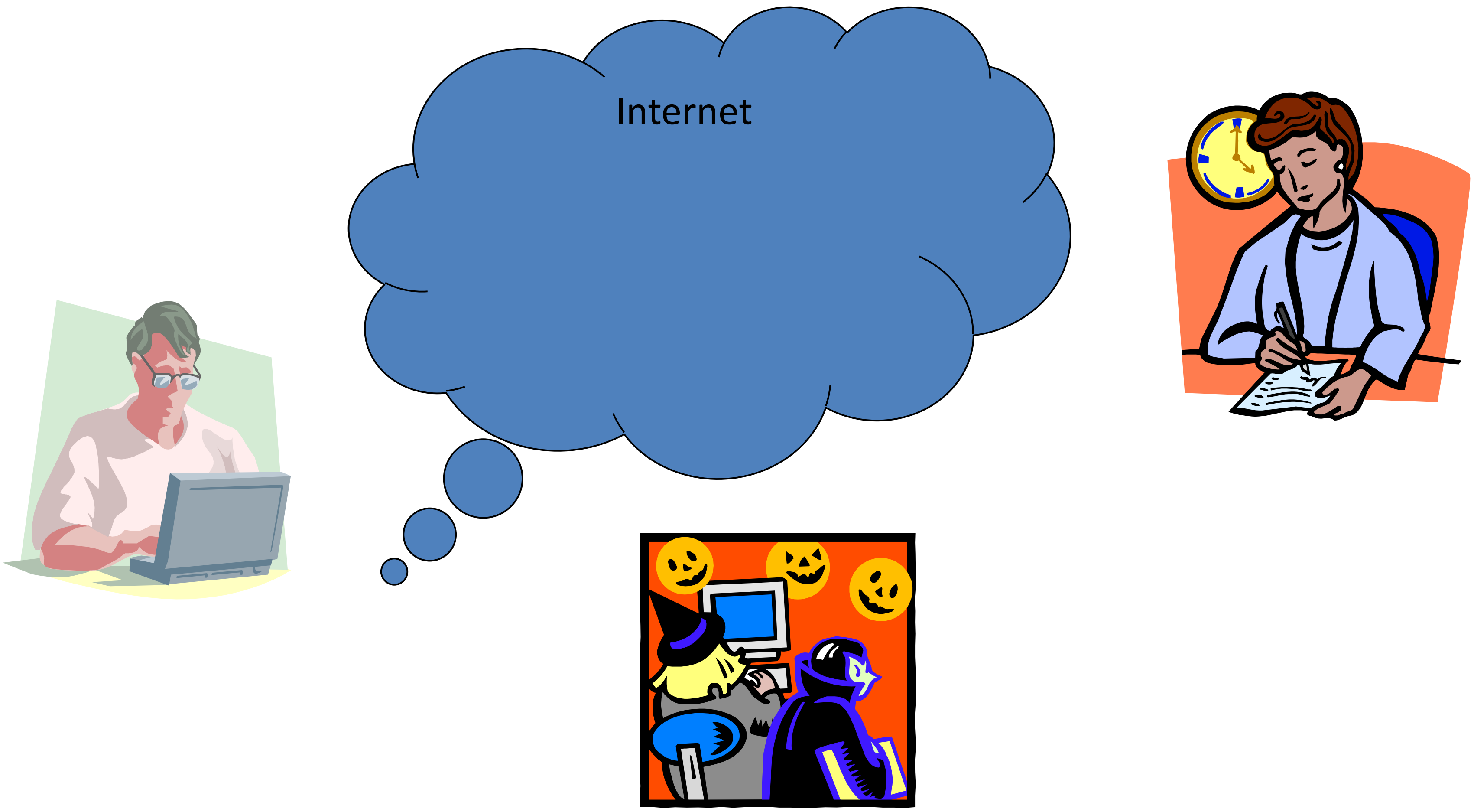
- In 1977, a 56-bit key was considered good enough.
  - Takes 1,000 years to try all keys with 56 1’s and 0’s at one million keys per second
- In Jan 1997, RSA Data Security Inc. issued “DES challenge”
  - DES cracked in 96 days
  - In Feb 1998, distributed.net cracked DES in 41 days
  - In July 1998, the Electroic Frontier Foundation (EFF) and distributed.net cracked in 56 hours using a $250K machine
  - In Jan 1999, the team did in less than 24 hours
- Double and Triple DES
  - Double DES only gives 2**57 = 2 x 2**56, instead of 2**112, due to *meet-in-the-middle* attack.
  - Triple DES recommended, but managing three keys more difficult

2. Public Key Cryptography


$$\begin{tikzcd}
 & K_{R(pub)} \arrow[d] & & K_{R(pri)} \arrow[d] & \\
S \arrow[r, "M"] & \boxed{E} \arrow[rr, "C"] & & \boxed{D} \arrow[r, "M"] & R
\end{tikzcd}$$

$K_{R(pub)}$ is Receiver's public key and $K_{R(pri)}$ is Receiver's private key.

Establishing Shared Secrete

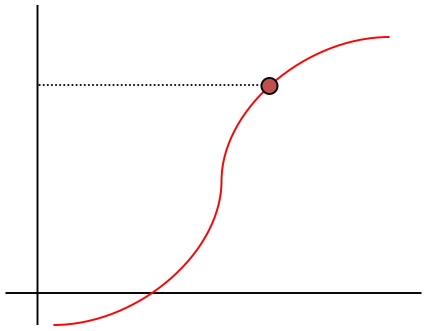


Problem Statement

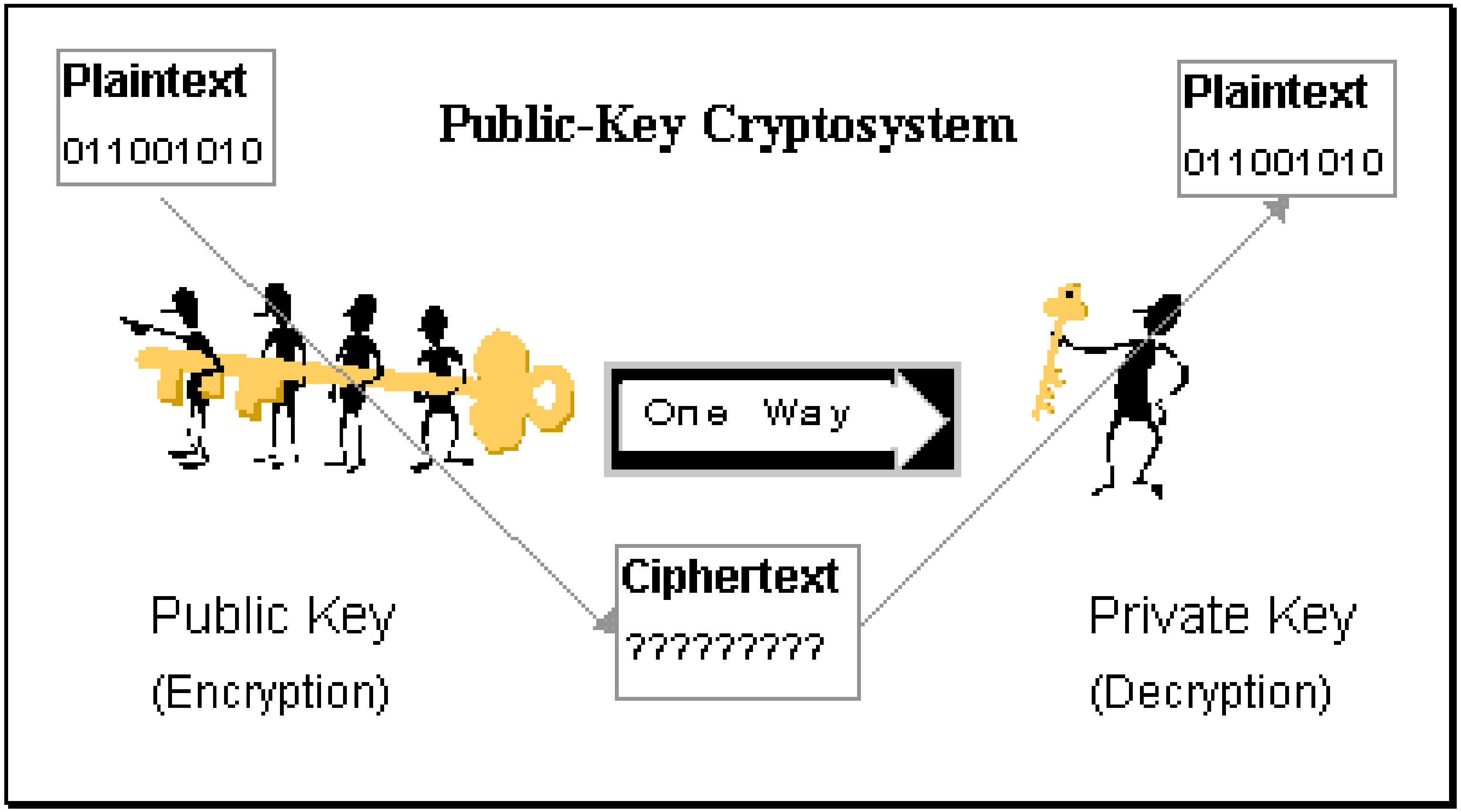
- Suppose Alice has an channel for communicating with Bob.
- Alice and Bob wish to use this channel to establish a shared secret.
- However, Eve is able to learn everything sent over the channel.
- If Alice and Bob have no other channel to use, can they establish a shared secret that Eve does not know?

Public Key Cryptographic Algorithms

Find a hard math problem, that is easy to compute in the forward direction, but is difficult to solve in the reverse direction, unless you have some special knowledge.



Public Key Cryptosystem



General Strategy

- A public key is used to encrypt a message that can be decrypted only by the matching private key.
- Bob can use Alice’s public key to encrypt messages. Only Alice can decrypt the message.
- Similarly, Alice can also use Bob’s public key.
- Alice and Bob exchange information, each keeping a secret to themselves.
- The secrets that they keep allow them to compute a shared secret.
- Since Eve lacks either of these secrets she is unable to compute the shared secret.

Simplified Math Tricks

- Public key cryptography is based on the mathematical concept of multiplicative inverse.
- Multiplicative inverses are two numbers that when multiplied equals one (e.g., 7 x 1/7 = 1)
- In modular mathematics, two whole numbers are inverses if they multiplies to 1 (e.g., 3 x 7 mod 10 = 1)
- Use modular inverse pairs to create public and private keys.
- Example
  - Message is 4
  - To scramble it, use 4 X 3 mod 10 = 2
  - To recover it, use 2 x 7 mod 10 = 4
- The security of public key systems depends on the difficulty of calculating inverses.

Asymmetric Algorithms

- Also called public-key algorithms.
- Encryption key is different from decryption key.
- Furthermore, one cannot be calculated from other.
- Encryption key is often called the **public key** and decryption key is often called the **private key**.
- Advantages: better key management.
- Disadvantages: slower, more complex.
- Both techniques are complementary.
- Examples: RSA, Diffie-Hellman, El Gamal, etc.

RSA Public Keys

- Named for Ron Rivest, Adi Shamir, and Len Adleman, published in 1978.
- Most widely known and used public key system.
- No shared secret is required.
- Based on some number-theoretic facts/results.
- Strength lies in the difficulty of determining the prime factors of a (large) number.
- Hardware improvements will not weaken RSA as long as appropriate key lengths are used.

RSA Key Generation

- Pick large random primes p,q.
- Let $p*q = n$ and $\phi=(p-1)(q-1)$.
- Choose a random number e such that: $1<e<\phi$ and $\gcd(e, \phi)=1$. (relative primes)
- Calculate the unique number d such that $1<d<\phi$ and $d*e \equiv 1 \pmod{\phi}$. (d is inverse of e)
- The public key is {e,n} and the private key is {d,n}.
- The factors p and q may be kept private or destroyed.

Encryption and Decryption

- Suppose Alice wants to send a message m to Bob.
- Alice computes $c = m^e \bmod n$, where $\{e,n\}$ is Bob's public key.
- She sends c to Bob.
- To decrypt, Bob computes $m = c^d \bmod n$, where $\{d,n\}$ is Bob's private key.
- The mathematical relationship between e and d ensures that Bob correctly recovers m.
- Since only Bob knows d, only he can decrypt.

RSA - Authentication

- Suppose Alice wants to send a message m to Bob and ensure him that the message is indeed from her.
- Alice computes signature $s = m^d \bmod n$, where {d,n} is Alice's private key.
- She sends m and s to Bob.
- To verify the signature, Bob computes using {e,n}
  $m = s^e \bmod n$ and checks that it is recovered.
- In practice, RSA is combined with a symmetric key cryptosystem (e.g., DES) to encrypt.
- RSA is usually combined with a hash function to sign a message.

Why Does it Work?

- It is secure because it is difficult to find ϕ or d using only e and n. Finding d is equivalent in difficulty to factoring n as p*q.
- It is feasible to encrypt and decrypt because:
  - It is possible to find large primes.
  - It is possible to find relative primes and their inverses.
  - Modular exponentiation is feasible.

RSA - Example

- Let p = 47 and q = 71
- then n = p*q = 3337
- (p-1)*(q-1) = 3220 = Φ_n
- Choose (at random) e = 79 [check using GCD (Greatest Common Divisor) that Φ_n and e are relatively prime.]
- Compute d = 79^{-1} mod 3220 = 1019
- Private key: {79, 3337}
- Public key: {1019, 3337}
- Let message m be 6882326879666683.
- To encrypt, first break it into blocks < n. [required condition]

RSA - Example (continued)

- Let message consists of the following blocks:
  - 688, 232, 687, 966, 668, 003
- For the first block
  - $688^{79} \bmod 3337 = 1570 = c_1$
- For the entire message we have
  - 1570, 2756, 2091, 2276, 2423, 158
- To decrypt first block
  - $1570^{1019} \bmod 3337 = 688$
- The rest of the message can be recovered in the same manner.

More on RSA

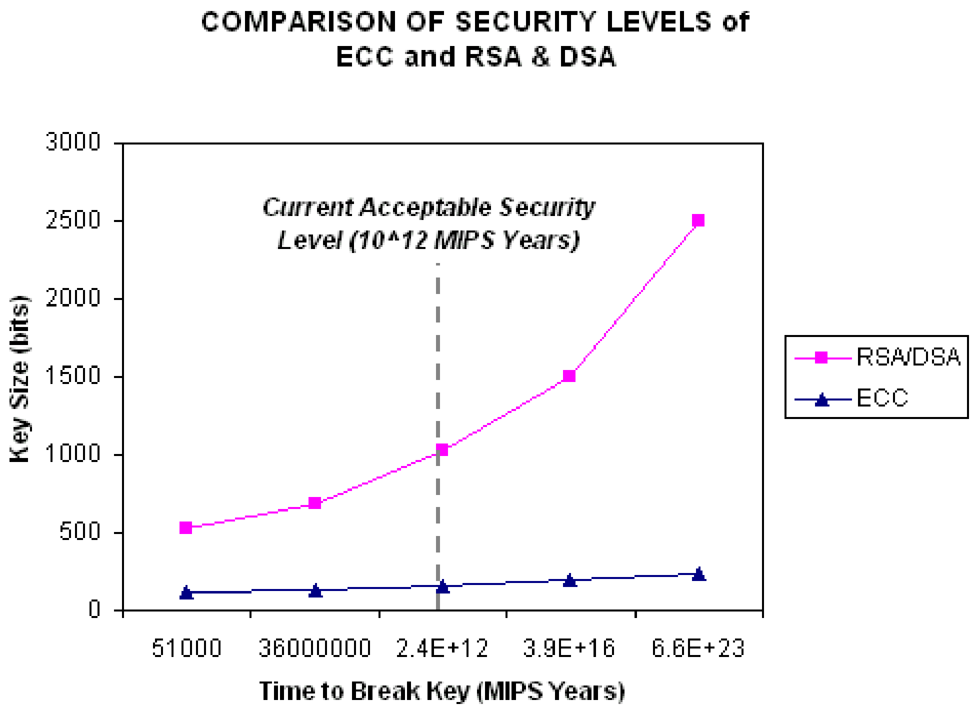
- RSA has been implemented in hardware.
- In hardware, RSA is about 1000 times slower than DES.
- In software, it is about 100 times slower.
- These numbers may change, but RSA can never approach the speed of symmetric algorithms.
- RSA encryption goes faster if e is chosen appropriately.
- Security of RSA depends on the problem of factoring large numbers. Though it has never been proven that one needs to factor n to calculate m from c and e.
- Most public key systems use at least 2048-bit key.
-

Key Lengths

- The longer the key, the longer it takes to do an exhaustive key search. The problem space is to find the private key.
- The longer the key, the greater the computational power required to perform cryptographic operations.
- This means a tradeoff between security and time/power.
- Time and power become important for portable devices (cell phones, smart cards, …).

Popular key lengths:

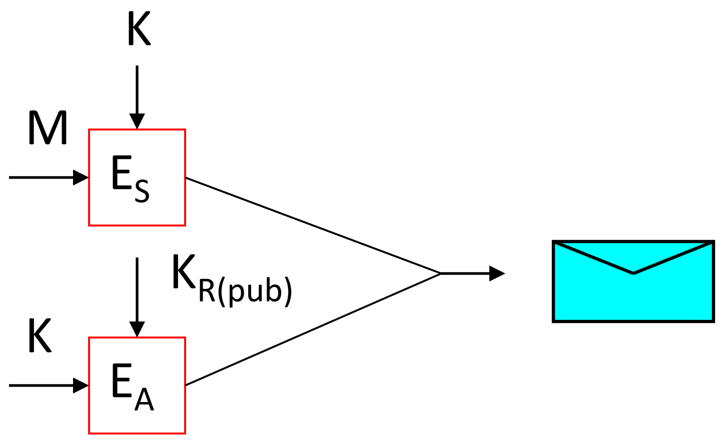
- DES = 56 bits
- 3-DES = 168 bits
- RSA = 2048 bits
- ECC < RSA for comparable cryptographic security.



3. Hybrid Cryptosystems

- In practice, public-key cryptography is used to secure and distribute **session keys**.
- These keys are used with symmetric algorithms for communication.
- Sender generates a random session key, encrypts it using receiver’s public key and sends it.
- Receiver decrypts the message to recover the session key.
- Both encrypt/decrypt their communications using the same key.
- Key is destroyed in the end.

Digital Envelope



K is a random session key and E_s is a symmetric encryption algorithm and E_A is an asymmetric encryption algorithm. The receiver recovers the secret key from the digital envelope using his/her private key. He/she then uses the secret key to decrypt the message.

4. Digital Signatures

- A digital signature is a protocol the produces the same effect as a real signature.
  - It is a mark that only sender can make
  - Other people can easily recognize it as belonging to the sender.
- Digital signatures must be:
  - Unforgeable: If P signs message M with signature S(P,M), it is impossible for someone else to produce the pair [M, S(P,M)].
  - Authentic: R receiving the pair [M, S(P,M)] can check that the signature is really from P.

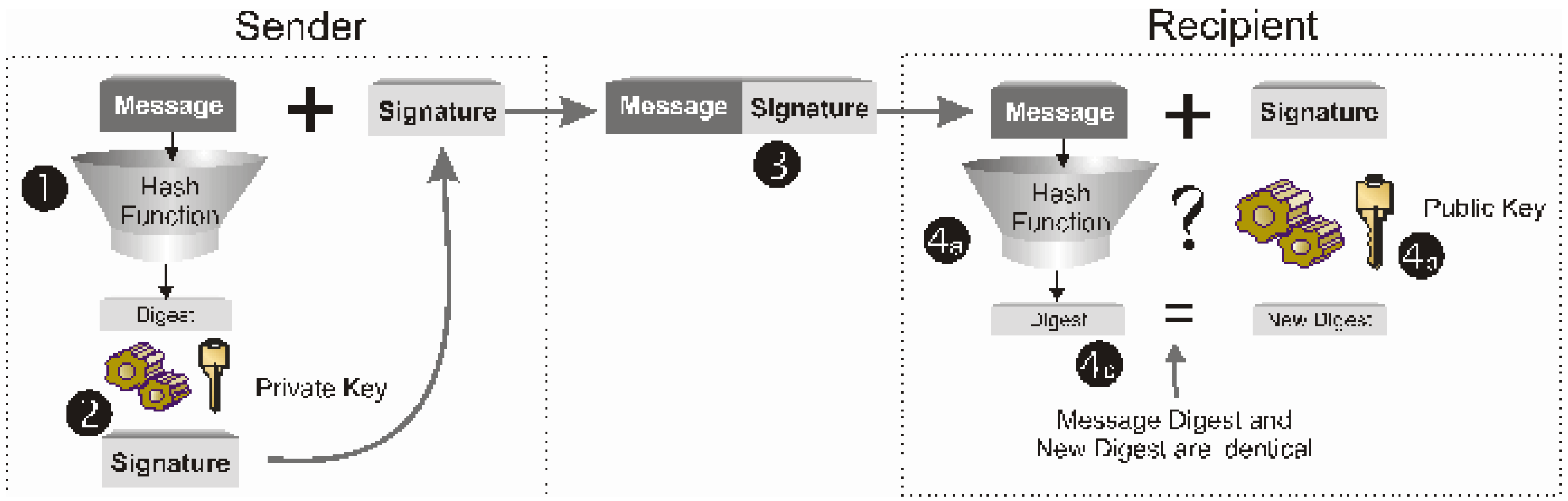
Digital Signatures: Symmetric Key

- Under private key encryption system, the secrecy of the key guarantees the authenticity of the message as well as its secrecy.
- It does not prevent forgery, however.
- There is no protection against repudiation (denial of sending a message).
- An arbitrator (a trusted third party) is needed to prevent forgery.

Digital Signatures - Public Key

- Public key encryption systems are ideally suited to digital signatures.
- Reverse of public key encryption/decryption.
- To sign a message, use your private key to encrypt the message.
- Send this signature together with the message.
- The receiver can verify the signature using your public key.
- Only you could have signed the message since your private key belongs to you and only you.
- The receiver saves the message and signature and anyone else can verify should you claim forgery.

Digital Signature Process



5. Message Digest

- How to assure integrity
  - Alice makes a message digest from a plaintext message.
  - Alice signs the message digest and sends the signed digest and plaintext to Bob
  - Bob re-computes the message digest from the plaintext.
  - Bob decrypts the signed digest with Alice’s public key.
  - Bob verifies that message is authentic if the message digest he computed is identical to the decrypted digest signed by Alice.

Possible Scenarios

- Message
  - Plaintext, can be altered
- Message, E(Message-digest, pub-key)
  - Plaintext, encrypted msg digest
- E(message,sym-key), E(message-digest,pub-key)
  - Cipher-text, encrypted msg digest

Cryptographic Hash Functions

- Hash functions are used in creating “digital fingerprint” of a large message.
- Requirements of such hash functions are:
  - easy to compute (i.e., reduce a message of variable size to a small digest of fixed size)
  - one-way, that is, hard to invert
  - collision-free (the probability that a randomly chosen message maps to an n-bit hash should ideally be ½ **n)
- To sign a message, first apply a hash function to create a message digest, encrypt the digest using private key and send it along with the message.

Uses for Hashing Algorithms

- Hash functions without secret keys are used:
  - To condense a message for digital signature.
  - To check the integrity of an input if the hash has been previously recorded.
- Such functions are called Modification Detection Codes (MDC's).
- Hash functions that use secret keys are called Message Authentication Codes (MAC's).
  - They are used for data origin authentication.
- MD5, SHA, SHA-2, SHA-3, SHA-256 etc.

6. Public Key Distribution

- Every user has his/her own public key and private key.
- Public keys are all published in a database.
- Sender and receiver agree on a cryptosystem.
- Sender gets receiver’s public key from the db.
- Sender encrypts the message and sends it.
- Receiver decrypts it using his/her private key.
- What can be a problem?

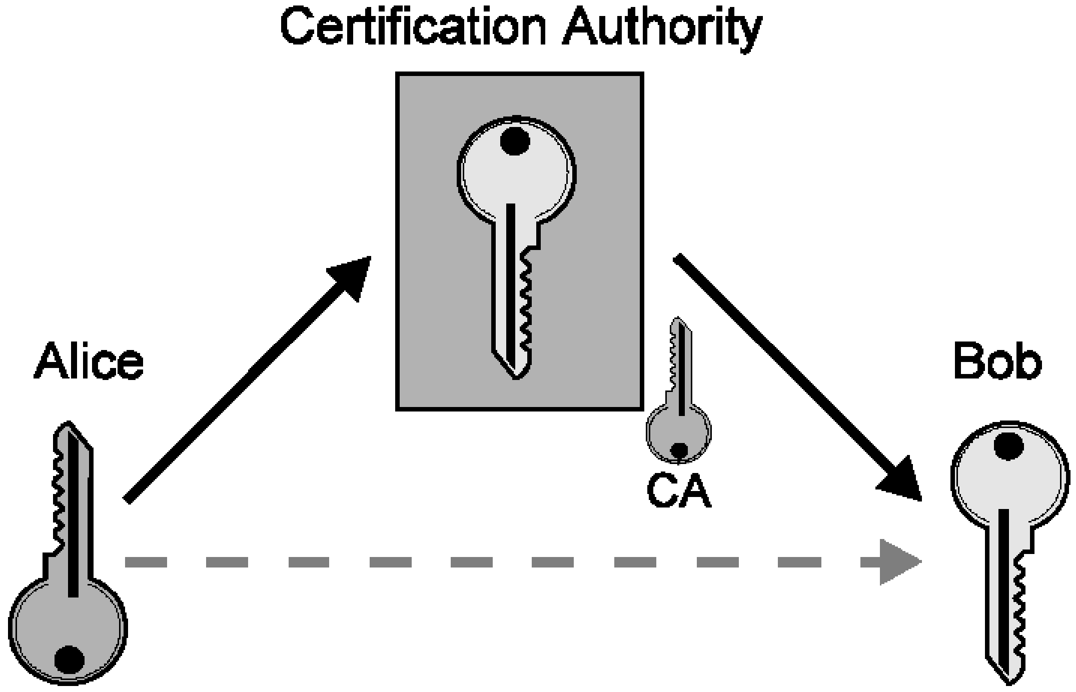
Matching keys to owners

- Insecurity of TCP/IP
  - No authentication
  - No privacy/confidentiality
  - Repudiation possible
- Public key cryptography not enough
- Need to match keys to owners
- Need *infrastructure* and *certificate authorities*

Public Key Infrastructure (PKI)

- As defined by Netscape:
  - *"Public-key infrastructure (PKI) is the combination of software, encryption technologies, and services that enables enterprises to protect the security of their communications and business transactions on the Internet."*
  - Integrates digital certificates, public key cryptography, and certification authorities
- Two major frameworks
  - X.509
  - PGP (Pretty Good Privacy)

Certification Authorities (CAs)



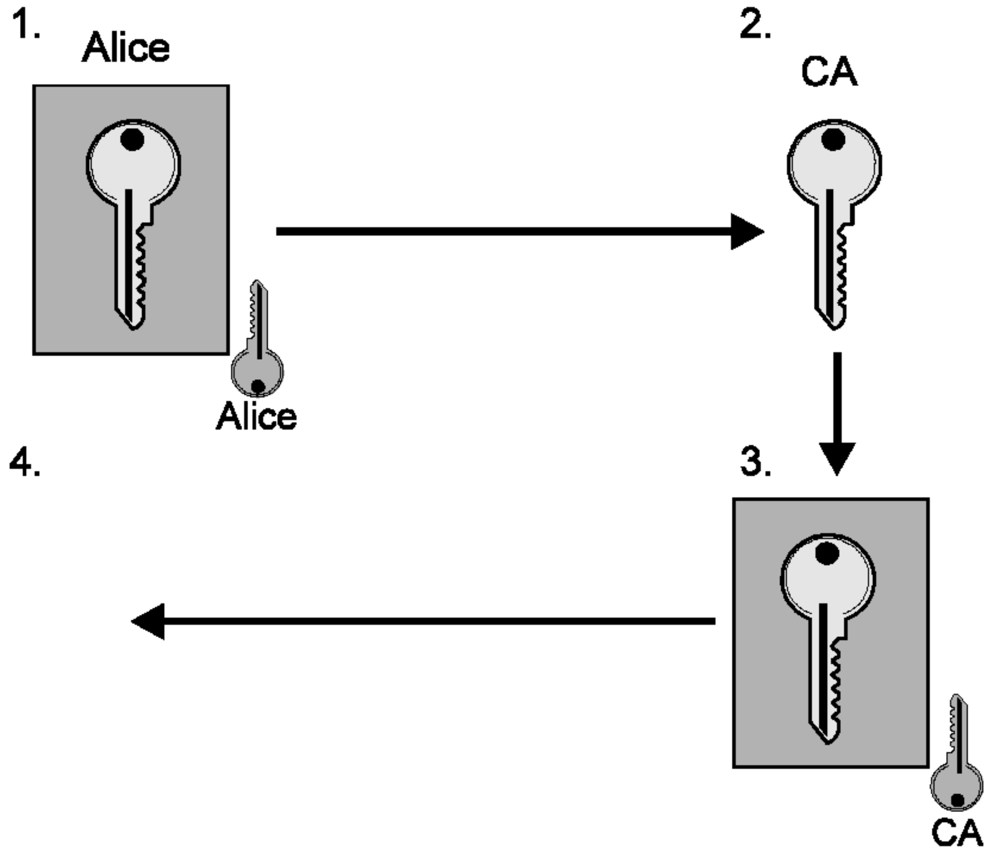
Certification Authorities (cont.)

- Guarantee connection between public key and end entity
  - Man-In-Middle no longer works undetected
  - Guarantee authentication and non-repudiation
  - Privacy/confidentiality not an issue here
    - Only concerned with linking key to owner
- Distribute responsibility
  - Hierarchical structure

Digital Certificates

- Introduced by IEEE-X.509 standard (1988)
- Originally intended for accessing IEEE-X.500 directories
  - Concerns over misuse and privacy violation gave rise to need for access control mechanisms
  - X.509 certificates addressed this need
- From X.500 comes the Distinguished Name (DN) standard
  - Common Name (CN)
  - Organizational Unit (OU)
  - Organization (O)
  - Country (C)
- Supposedly enough to give every entity on Earth a unique name

Obtaining Certificates



Obtaining Certificates

- 1\. Alice generates A_{priv}, A_{pub} and A_{ID}; Signs {A_{pub}, A_{ID}} with A_{priv}
  - Proves Alice holds corresponding A_{priv}
  - Protects {A_{pub}, A_{ID}} en route to CA
- 2\. CA verifies signature on {A_{pub}, A_{ID}}
  - Verifies A_{ID} offline (optional)
- 3\. CA signs {A_{pub}, A_{ID}} with CA_{priv}
  - Creates certificate
  - Certifies binding between A_{pub} and A_{ID}
  - Protects {A_{pub}, A_{ID}} en route to Alice
- 4\. Alice verifies {A_{pub}, A_{ID}} and CA signature
  - Ensures CA didn't alter {A_{pub}, A_{ID}}
- 5\. Alice and/or CA publishes certificate

PKI: Benefits

- Provides authentication
- Verifies integrity
- Ensures privacy
- Authorizes access
- Authorizes transactions
- Supports non-repudiation

PKI: Risks

- Certificates only as trustworthy as their CAs
  - Root CA is a single point of failure
- PKI only as secure as private signing keys
- DNS not necessarily unique
- Server certificates authenticate DNS addresses, not site contents
- CA may not be authority on certificate contents
  - i.e., DNS name in server certificates
- ...