CIS 505: Software Systems
Lecture Note : Cryptography

Insup Lee
Department of Computer and Information Science
University of Pennsylvania

CIS 505, Spring 2007

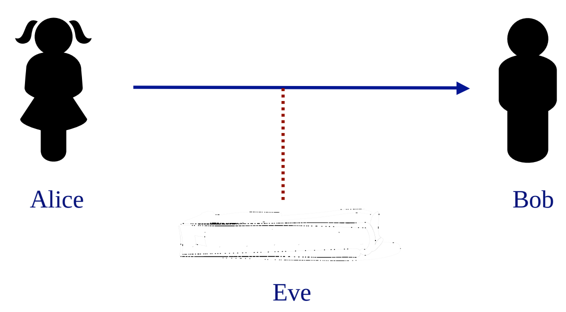
What's Crypto Good for?

- Data Confidentiality
  - Keep data and communication secret
  - Privacy of personal financial/health records, etc.
  - Military and commercial relevance
  - Encryption / decryption
- Data Integrity and Authenticity
  - Protect reliability of data against tampering
  - Can we be sure of the source and content of information?
  - Hashes
  - Also related: repudiation

What Crypto Isn't

- Cryptography isn't **the** solution to security
  - Buffer overflows, worms, viruses, etc.
  - SSL-enabled websites != security
- It's a tool, not a solution.
- Even when used, difficult to get right.
  - Choice of encryption algorithms
  - Choice of parameters
  - **Implementation**
  - **Hard to detect errors.**
    - **Even when crypto fails, the program may still work.**

Principals



Encryption and Decryption

The following identity must hold true:

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

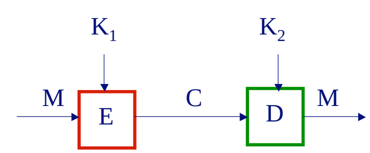
An aside...

- What is good ciphertext?
  - o Difficult to go from ciphertext -> plaintext
  - o Should appear random (each bit has equal probability of being 0/1)

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.

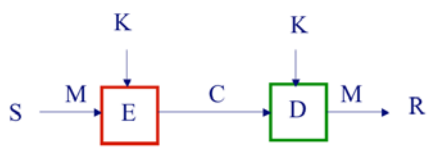
Key Based Encryption/Decryption



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

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

1. Secrete Key Cryptography

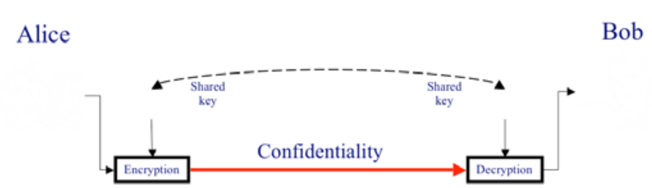


K is the secret key shared by both the sender (S) and receiver (R).

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.

Symmetric Encryption



Q: What problem are we ignoring?

Block Ciphers and Stream Ciphers

- Stream Ciphers
  - o Combine (e.g., XOR) pseudorandom stream of bits with plaintext
- Block Ciphers
  - o Fixed block size
  - o Encrypt block-sized portions of plaintext
  - o Combine encrypted blocks (more on this later)

Monoalphabetic Ciphers

- Substitution of letters (rot13; add 13 to each letter)
  - I ROT13 encrypt all my data twice for extra protection.
- Classical way of encoding text strings (Caesar Cipher)
- The key for decoding is the inverse substitution
- Encoding and decoding are efficient
- Theoretically sound: the number of substitutions of ASCII alphabet is VERY large (128!), and an adversary cannot possibly try out all possible substitutions to decipher
- Main problem: Any human language has distinct frequent letter (e.g. vowels) combos
  - E.g. e is the most common letter in English text, th is the most common sequence of adjacent symbols
  - Cryptogram puzzles

One-time Pads

- Choose one time pad to encrypt message.
  - Never reuse the pad. (Lookup "Venona")
  - Bob must also know pad to decrypt.
- Simple example
  - Want to encrypt *k bit message.*
  - *Using Random Number Generator (not pseudorandom), generate random stream of k bits*
  - *Encryption:*
    - *Bit i of ciphertext = bit i of plaintext XOR bit i of one-time pad*
  - *Decryption:*
    - *Bit i of plaintext = bit i of ciphertext XOR bit i of one-time pad*

Secret-Key Cryptography

- Sender and receiver share the secret key
- This is also called symmetric key cryptography
- A popular scheme for many years: DES (Data Encryption Standard) promoted by NSA
  - Key is 56 bits (extended to 64 bits using 8 parity bits)
  - Input data is processed in chunks of 64-bit blocks, by subjecting to a series of transformations using the key
- New standard: NIST's AES (Rijndael)
- Distribution of keys is a problem

Secret-Key Cryptography

- Substitution-Permutation Networks
  - S-Box
    - Input: *sequence of l bits*
    - *Output: new sequence of l bits*
    - *Mapping from one bit string to another*
  - *Permutation*
    - *Input: sequence of l bits*
    - *Output: permutation of the input*
- *Symmetric key encryption typically uses many rounds of S-Boxes and permutations, incorporating the key.*
- *Many modes:*
  - *ECB (Electronic Code Book)*
  - *CBC (Cipher Block Chaining)*
  - *Counter-mode*

DES (Data Encryption Standard)

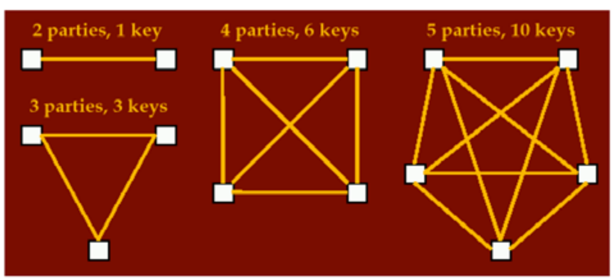
- In 1972, NIST (National Institute of Standards and Technology) decide to assist the development of a secure cryptographic method.
- In 1974, it settled on DES, which was submitted by IBM and is the Data Encryption Algorithm developed by Horst Feistel.
- NSA shortened the secrete key to 56 bits from 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 craed 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

Symmetric Key - Issues

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



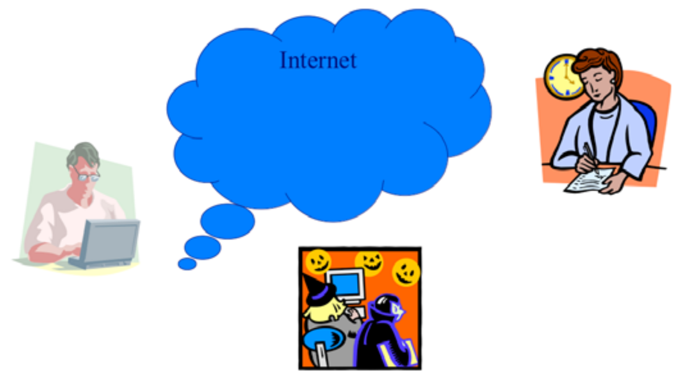
Secrete Key Assurances

- 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)

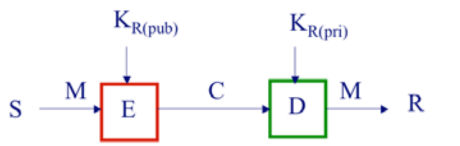
Example: non-repudiation

- 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?

Establishing Shared Secrete



2. Public Key Cryptography



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

Problem Statement

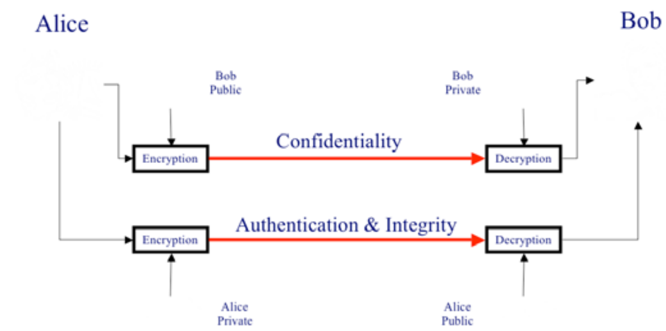
- Suppose Alice has an channel for communicating with Bob.
- Alice and Bob wish to use this channel to established a shared secret.
- However, Eve is able to learn everything sent over the channel.
- If Alice and Bob have no other channel to use, can they establish a shared secret that Eve does not know?

Asymmetric Encryption

- Two complementary keys
  - o Private key (kept secret)
  - o Public key (published)
- Private key VERY difficult to compute from public key
- Encryption with one key can only be reversed with the other key
- Used in PGP (Pretty Good Privacy) & PKI (Public Key Infrastructure)
- Best known RSA, DSA for signatures

RSA Rivest Shami Adleman, ECC – Eliptic Curve Cryptography, DSA – Digital Signature Algorithm

Asymmetric Encryption cont'd



Public-Key Cryptography

- All users pick a public key/private key pair
  - o publish the public key
  - o private key not published
- Public key is the encryption key
  - o To send a message to user Alice, encrypt the message with Alice's public key
- Private key is the decryption key
  - o Alice decrypts the ciphertext with its private key
- Popular schemes (1970s): Diffie-Hellman, RSA

Simplified Math Facts

- 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
  - o Message is 4
  - o To scramble it, use 4 X 3 mod 10 = 2
  - o To recover it, use 2 x 7 mod 10 = 4
- The security of public key systems depends on the difficulty of calculating inverses.

RSA Key Generation

- Pick large random primes p,q.
- Let p*q = n and ϕ=(p-1)(q-1).
- Choose a random number e such that: 1<e<ϕ and gcd(e, ϕ)=1. (relative primes)
- Calculate the unique number d such that 1<d<ϕ and d*e $\equiv$ 1 (mod ϕ). (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.

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} mod 3337 = 1570 = c_1
- For the entire message we have
  - 1570, 2756, 2091, 2276, 2423, 158
- To decrypt first block
  - 1570^{1019} mod 3337 = 688
- The rest of the message can be recovered in the same manner.

More on RSA

- Introduced by Rivest, Shamir, and Adleman in 1979
- Foundations in number theory and computational difficulty of factoring
- Not mathematically proven to be unbreakable, but has withstood attacks and analysis
  - Ideally, we would like to prove a theorem saying "if intruder does not know the key, then it cannot construct plaintext from the ciphertext by executing a polynomial-time algorithm"
- Public and private keys are derived from secretly chosen large prime numbers
- Plaintext is viewed as a large binary number and encryption is exponentiation in modulo arithmetic
- Intruder will have to factor large numbers (and there are no known polynomial-time algorithms for this)
  - 2002's major result: polynomial-time test to check if a number is prime

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 1,024-bit key.
- The RSA algorithm is patented in the US, but not in any other country.
- The US patent expires on September 20, 2000!

3. 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.

How can Alice *sign* a digital document ?

- Let S(A,M) be the message M tagged with Alice's signature
- No forgery possible: If Alice signs M then nobody else can generate S(A,M)
- Authenticity check: If you get the message S(A,M) you should be able to verify that this is really created by Alice
- No alteration: Once Alice sends S(A,M), nobody (including Alice) can tamper this message
- No reuse: Alice cannot duplicate S(A,M) at a later time

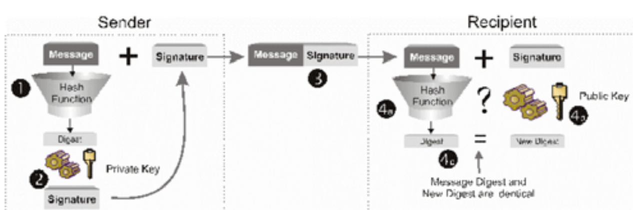
Digital Signatures: Symmetric Key

- 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



Digital Signatures with Public Keys

- Suppose K is public key and k is private key for Alice, and encryption/decryption is commutative:
  D(E(M,K), k) = E(D(M,k),K)=M
- To sign a message M, Alice simply sends D(M,k)
- Receiver uses Alice's public key to compute E(D(M,k),K), to retrieve M
  - Authenticity of signature because only Alice knows the private key k
- RSA encryption does satisfy the required commutativity
- To ensure "no reuse" and "no alteration" the message must include a timestamp
- The scheme is made more efficient by computing D(H(M),k), where H(M) is the *secure hash* of M
  - Hashing gives a constant size output
  - Hard to invert
  - We'll come to hashing in a bit...

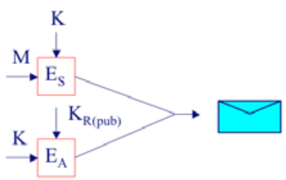
Symmetric vs. Asymmetric Encryption

- If public key crypto is so amazing, why use secret key?
  - o All of the known public key cryptography techniques are much slower than the known secret key techniques.
- So we use hybrid systems
  - o Get started with public key
  - o Switch to secret key

4. 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.

5. Message Digest

- How to assure integrity
  - o Alice makes a message digest from a plaintext message.
  - o Alice signs the message digest and sends the signed digest and plaintext to Bob
  - o Bob re-computes the message digest from the plaintext.
  - o Bob decrypts the signed digest with Alice's public key.
  - o 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

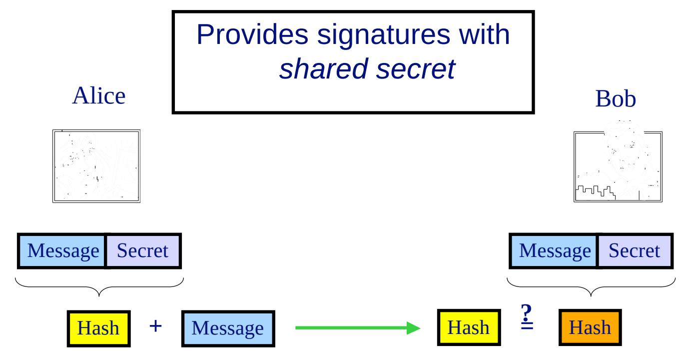
- Hash functions are used in creating “digital fingerprint” of a large message.
- Requirements of such hash functions are:
  - easy to compute (i.e., reduce a message of variable size to a small digest of fixed size)
  - one-way, that is, hard to invert
  - collision-free (the probability that a randomly 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.

Hash Functions

- Produce *hash values* for data access or security
- *Hash value*: Number generated from a string of text
- Hash is substantially smaller than the text itself
  - Hash is constant size
- Unlikely that other text produces the same hash value (collision resistance)
- Unidirectional (cannot calculate text from hash)
- Provides: Integrity & Authentication
- E.g.,: MD5, SHA-1, SHA-128, SHA-256
  - Collisions recently found in both

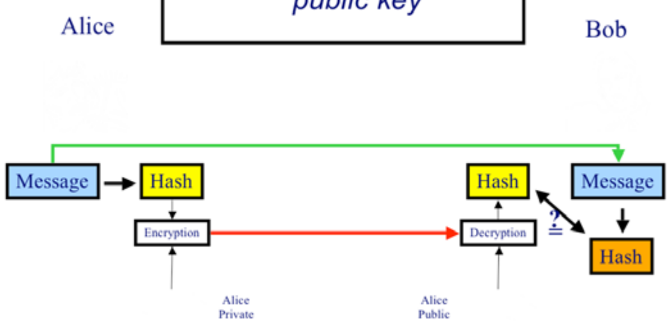
SHA – Secure Hash Algorithm, MD5 – Message Digest

Hash Functions cont’d



Hash Functions cont'd

Provides signatures with *public key*



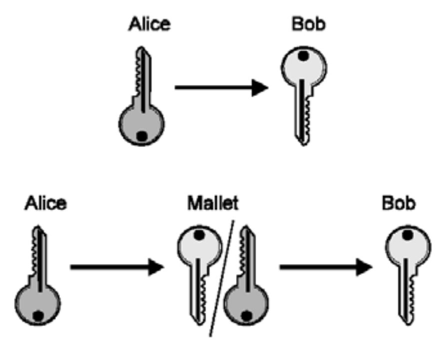
Uses for Hashing Algorithms

- 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

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?

Potential Problem



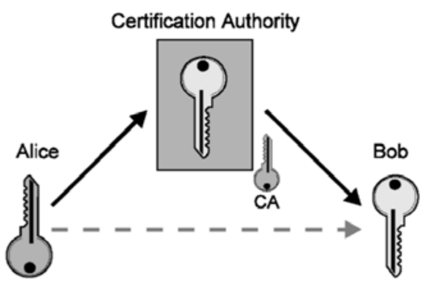
Matching keys to owners

- 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)



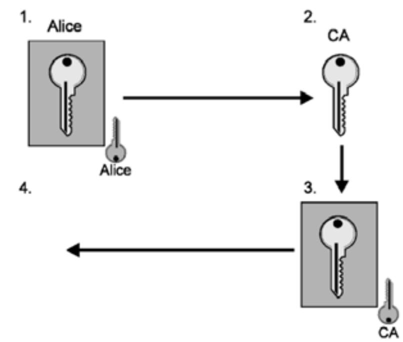
Certification Authorities (cont.)

- 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
  - o 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
  - o i.e., DNS name in server certificates
- ...

Ack and References

- Lecture materials are drawn from many sources, including Matt Blaze, Micah Sherr
- Cryptography: Theory and Practice. Douglas R. Stinson.
- Applied Cryptography. Bruce Schneier
- Handbook of Applied Cryptography. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. Available free online at ttp://www.cacr.math.uwaterloo.ca/hac/