CS 426 (Fall 2010)

Public Key Encryption and Digital Signatures

Review of Secret Key (Symmetric) Cryptography

- Confidentiality
  - stream ciphers (uses PRNG)
  - block ciphers with encryption modes
- Integrity
  - Cryptographic hash functions
  - Message authentication code (keyed hash functions)
- Limitation: sender and receiver must share the same key
  - Needs secure channel for key distribution
  - Impossible for two parties having no prior relationship
  - Needs many keys for n parties to communicate

Public Key Encryption Overview

- Each party has a PAIR (K, K^{-1}) of keys:
  - K is the **public** key, and used for encryption
  - K^{-1} is the **private** key, and used for decryption
  - Satisfies $\mathbf{D}_{K^{-1}}[\mathbf{E}_K[M]] = M$
- Knowing the public-key K, it is computationally infeasible to compute the private key K^{-1}
  - How to check (K,K^{-1}) is a pair?
  - Offers only computational security. PK Encryption impossible when P=NP, as deriving K^{-1} from K is in NP.
- The public-key K may be made publicly available, e.g., in a publicly available directory
  - Many can encrypt, only one can decrypt
- Public-key systems aka *asymmetric* crypto systems

Public Key Cryptography Early History

- The concept is proposed in Diffie and Hellman (1976) “New Directions in Cryptography”
  - public-key encryption schemes
  - public key distribution systems
    - Diffie-Hellman key agreement protocol
  - digital signature
- Public-key encryption was proposed in 1970 by James Ellis
  - in a classified paper made public in 1997 by the British Governmental Communications Headquarters
- Concept of digital signature is still originally due to Diffie & Hellman

Public Key Encryption Algorithms

- Almost all public-key encryption algorithms use either number theory and modular arithmetic, or elliptic curves
- RSA
  – based on the hardness of factoring large numbers
- El Gamal
  – Based on the hardness of solving discrete logarithm
  – Basic idea: public key g^x, private key x, to encrypt: $[g^y, g^{xy} M]$.

RSA Algorithm

- Invented in **1978** by Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman
  - Published as R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
- Security relies on the difficulty of factoring large composite numbers
- Essentially the same algorithm was discovered in 1973 by Clifford Cocks, who works for the British intelligence

RSA Public Key Crypto System

Key generation:

1. Select 2 large prime numbers of about the same size, p and q

   Typically each p, q has between 512 and 2048 bits

2. Compute $n = pq$, and $\Phi(n) = (q-1)(p-1)$

3. Select e, $1<e< \Phi(n)$, s.t. $\gcd(e, \Phi(n)) = 1$

   Typically e=3 or e=65537

4. Compute d, $1< d< \Phi(n)$ s.t. $ed \equiv 1 \bmod \Phi(n)$

   Knowing $\Phi(n)$, d easy to compute.

Public key: (e, n)

Private key: d

RSA Description (cont.)

Encryption

Given a message M, $0 < M < n$ $\quad M \in Z_n - \{0\}$

use public key (e, n)

compute $C = M^e \bmod n$ $\quad C \in Z_n - \{0\}$

Decryption

Given a ciphertext C, use private key (d)

Compute $C^d \bmod n = (M^e \bmod n)^d \bmod n = M^{ed} \bmod n = M$

$C = M^e \bmod (n=pq)$

$\longrightarrow$

Plaintext: M

Ciphertext: C

$C^d \bmod n$

$\longleftarrow$

From n, difficult to figure out p,q

From (n,e), difficult to figure d.

From (n,e) and C, difficult to figure out M s.t. $C = M^e$

RSA Example

- $p = 11, q = 7, n = 77, \Phi(n) = 60$
- $d = 13, e = 37$ ($ed = 481$; $ed \bmod 60 = 1$)
- Let $M = 15$. Then $C \equiv M^e \bmod n$
  - $C \equiv 15^{37} \pmod{77} = 71$
- $M \equiv C^d \bmod n$
  - $M \equiv 71^{13} \pmod{77} = 15$

RSA Example 2

- Parameters:
  - p = 3, q = 5, q= pq = 15
  - $\Phi(n)$ = ?
- Let e = 3, what is d?
- Given M=2, what is C?
- How to decrypt?

RSA Security

- Security depends on the difficulty of factoring n
  - Factor n => $\Phi(n)$ => compute d from (e, $\Phi(n)$)
- The length of n=pq reflects the strength
  - 700-bit n factored in 2007
  - 768 bit factored in 2009
- 1024 bit for minimal level of security today
  - likely to be breakable in near future
- Minimal 2048 bits recommended for current usage
- NIST suggests 15360-bit RSA keys are equivalent in strength to 256-bit
- RSA speed is quadratic in key length

Real World Usage of Public Key Encryption

- Often used to encrypt a symmetric key
  - To encrypt a message M under a public key (n,e), generate a new AES key K, compute [RSA(n,e,K), AES(K,M)]
- Plain RSA does not satisfy IND requirement.
  - How to break it?
- One often needs padding, e.g., Optimal Asymmetric Encryption Padding (OAEP)
  - Roughly, to encrypt M, chooses random r, encode M as
    $$M' = [X = M \oplus H_1(r) \quad , Y = r \oplus H_2(X)\ ]$$
    where H_1 and H_2 are cryptographic hash functions, then encrypt it as $(M')^e \bmod n$
  - Note that given $M'=[X,Y]$, $r = Y \oplus H_2(X)$, and $M = X \oplus H_1(r)$

Digital Signatures: The Problem

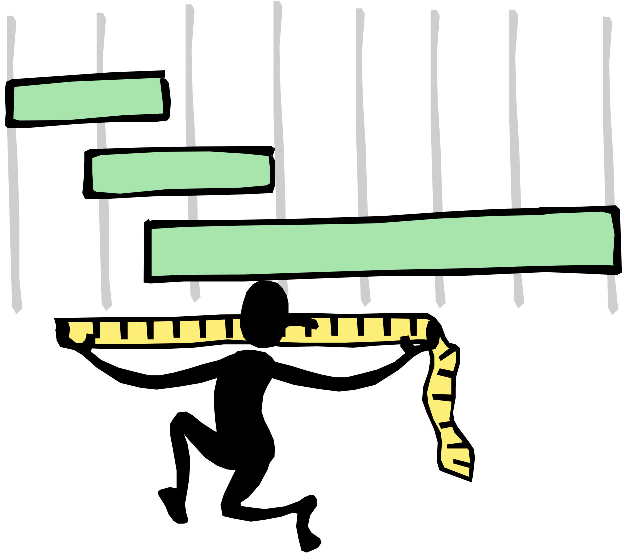
- Consider the real-life example where a person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
- Contracts, they are valid if they are signed.
- Signatures provide non-repudiation.
  - ensuring that a party in a dispute cannot repudiate, or refute the validity of a statement or contract.
- Can we have a similar service in the electronic world?
  - Does Message Authentication Code provide non-repudiation? Why?

Digital Signatures

- MAC: One party generates MAC, one party verifies integrity.
- Digital signatures: One party generates signature, many parties can verify.
- Digital Signature: a data string which associates a message with some originating entity.
- Digital Signature Scheme:
  - a signing algorithm: takes a message and a (private) signing key, outputs a signature
  - a verification algorithm: takes a (public) key verification key, a message, and a signature
- Provides:
  - Authentication, Data integrity, Non-Repudiation

Digital Signatures and Hash

- Very often digital signatures are used with hash functions, hash of a message is signed, instead of the message.
- Hash function must be:
  - Pre-image resistant
  - Weak collision resistant
  - Strong collision resistant



RSA Signatures

Key generation (as in RSA encryption):

- Select 2 large prime numbers of about the same size, p and q
- Compute $n = pq$, and $\Phi = (q - 1)(p - 1)$
- Select a random integer e, $1 < e < \Phi$, s.t. $\gcd(e, \Phi) = 1$
- Compute d, $1 < d < \Phi$ s.t. $ed \equiv 1 \bmod \Phi$

Public key: (e, n) **used for verification**

Secret key: d, **used for generation**

RSA Signatures (cont.)

Signing message M

- Verify $0 < M < n$
- Compute $S = M^d \bmod n$

Verifying signature S

- Use public key (e, n)
- Compute $S^e \bmod n = (M^d \bmod n)^e \bmod n = M$

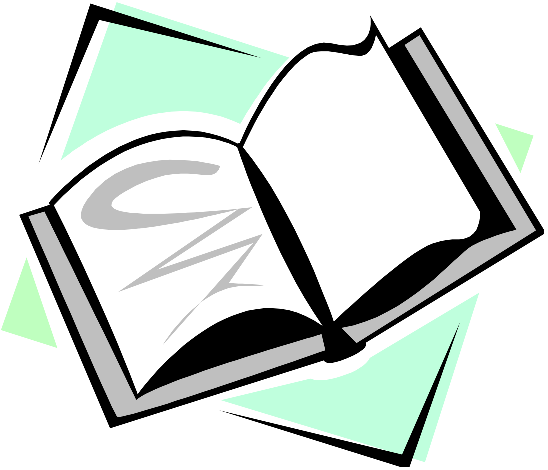
Note: in practice, a hash of the message is signed and not the message itself.

The Big Picture

	Secret Key Setting	Public Key Setting
Secrecy / Confidentiality	Stream ciphers Block ciphers + encryption modes	Public key encryption: RSA, El Gamal, etc.
Authenticity / Integrity	Message Authentication Code	Digital Signatures: RSA, DSA, etc.

Readings for This Lecture

- Differ & Hellman:
  - New Directions in Cryptography



Coming Attractions …

- Key management and certificates