

Information Security

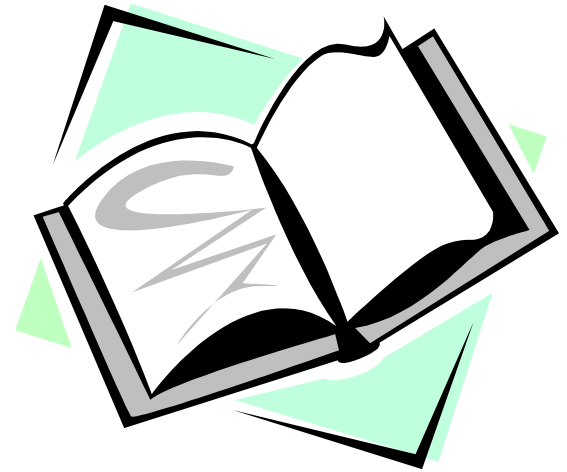
CS 526



Topic 5: Public Key Encryption and Digital Signatures

Readings for This Lecture

- Required: On Wikipedia
 - [Public key cryptography](#)
 - [RSA](#)
 - [Diffie–Hellman key exchange](#)
 - [ElGamal encryption](#)
- Required:
 - Diffie & Hellman: “New Directions in Cryptography” IEEE Transactions on Information Theory, Nov 1976.



Review of Secret Key (Symmetric) Cryptography

- Confidentiality
 - stream ciphers (uses PRNG)
 - block ciphers with encryption modes (CBC or CTR)
- 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

Concept of Public Key Encryption

- 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 $D_{K^{-1}}[E_K[M]] = M$
- 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. Secure Public Key encryption is 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

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

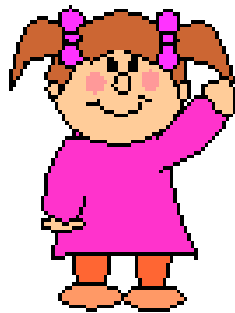
Public Key Encryption Algorithms

- Most public-key encryption algorithms use either modular arithmetic number theory, or elliptic curves
- RSA
 - based on the hardness of factoring large numbers
- El Gamal
 - Based on the hardness of solving discrete logarithm
 - Use the same idea as Diffie-Hellman key agreement

Diffie-Hellman Key Agreement Protocol

Not a Public Key Encryption system, but can allow A and B to agree on a shared secret in a public channel (against passive, i.e., eavesdropping only adversaries)

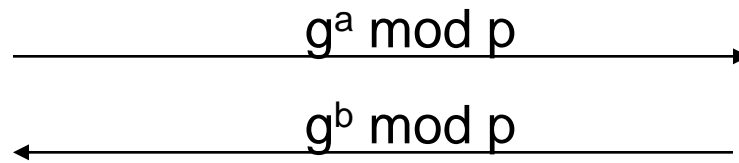
Setup: p prime and g generator of Z_p^* , p and g public.



Pick random, secret a

Compute and send $g^a \bmod p$

$$K = (g^b \bmod p)^a = g^{ab} \bmod p$$



Pick random, secret b

Compute and send $g^b \bmod p$

$$K = (g^a \bmod p)^b = g^{ab} \bmod p$$

Diffie-Hellman

- Example: Let $p=11$, $g=2$, then

a	1	2	3	4	5	6	7	8	9	10	11
g^a	2	4	8	16	32	64	128	256	512	1024	2048
$g^a \bmod p$	2	4	8	5	10	9	7	3	6	1	2

A chooses 4, B chooses 3, then shared secret is

$$(2^3)^4 = (2^4)^3 = 2^{12} = 4 \pmod{11}$$

Adversaries sees $2^3=8$ and $2^4=5$, needs to solve one of $2^x=8$ and $2^y=5$ to figure out the shared secret.

Security of DH is based on Three Hard Problems

- **Discrete Log (DLG) Problem:** Given $\langle g, h, p \rangle$, computes a such that $g^a = h \pmod p$.
- **Computational Diffie Hellman (CDH) Problem:** Given $\langle g, g^a \pmod p, g^b \pmod p \rangle$ (without a, b) compute $g^{ab} \pmod p$.
- **Decision Diffie Hellman (DDH) Problem:** distinguish (g^a, g^b, g^{ab}) from (g^a, g^b, g^c) , where a, b, c are randomly and independently chosen
- If one can solve the DL problem, one can solve the CDH problem. If one can solve CDH, one can solve DDH.

Assumptions

- DDH Assumption: DDH is hard to solve.
- CDH Assumption: CDH is hard to solve.
- DLG Assumption: DLG is hard to solve

- DDH assumed difficult to solve for large p (e.g., at least 1024 bits).
- Warning:
 - New progress in 2013 can solve discrete log for p values with some properties. No immediate attack against practical setting yet.
 - Look out when you need to use/implement public key crypto
 - May want to consider Elliptic Curve-based algorithms, require shorter keys than discrete log (for 128-bit security require 256 bit key size)

ElGamal Encryption

- Public key $\langle g, p, h=g^a \bmod p \rangle$
- Private key is a
- To encrypt: chooses random b , computes $C=[g^b \bmod p, g^{ab} * M \bmod p]$.
 - Idea: for each M , sender and receiver establish a shared secret g^{ab} via the DH protocol. The value g^{ab} hides the message M by multiplying it.
- To decrypt $C=[c_1, c_2]$, computes M where
 - $((c_1^a \bmod p) * M) \bmod p = c_2$.
 - To find M for $x * M \bmod p = c_2$, compute z s.t. $x*z \bmod p = 1$, and then $M = C_2 * z \bmod p$
- CDH assumption ensures M cannot be fully recovered.
- IND-CPA security requires DDH.

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. $ed \equiv 1 \pmod{\Phi(n)}$
Knowing $\Phi(n)$, d easy to compute.

Public key: (e, n)

Private key: d

RSA Description (cont.)

Encryption

Given a message M , $0 < M < n$ $M \in \mathbb{Z}_n - \{0\}$

use public key (e, n)

compute $C = M^e \bmod n$ $C \in \mathbb{Z}_n - \{0\}$

Decryption

Given a ciphertext C , use private key (d)

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

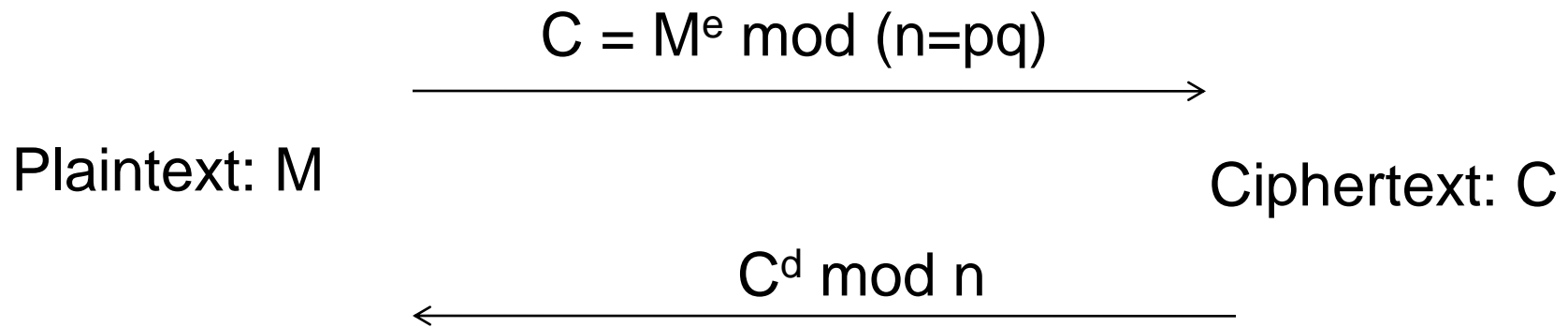
RSA Example

- $p = 11, q = 7, n = 77, \Phi(n) = 60$
- $d = 13, e = 37$ ($ed = 481; ed \bmod 60 = 1$)
- 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:
 - $p = 3, q = 5, n = pq = 15$
 - $\Phi(n) = ?$
- Let $e = 3$, what is d ?
- Given $M=2$, what is C ?
- How to decrypt?

Hard Problems on Which RSA Security Depends



1. **Factoring Problem:** Given $n=pq$, compute p,q
2. **Finding RSA Private Key:** Given (n,e) , compute d s.t. $ed = 1 \pmod{\Phi(n)}$.
 - Given (d,e) such that $ed = 1 \pmod{\Phi(n)}$, there is a clever randomized algorithm to factor n efficiently.
 - **Implication: cannot share the modulus n among multiple users**
3. **RSA Problem:** From (n,e) and C , compute M s.t. $C = M^e$
 - Aka computing the e 'th root of C .
 - Can be solved if n can be factored

RSA Security and Factoring

- Security depends on the difficulty of factoring n
 - Factor $n \Rightarrow$ compute $\Phi(n) \Rightarrow$ compute d from (e, n)
 - Knowing e, d such that $ed = 1 \pmod{\Phi(n)} \Rightarrow$ factor n
- The length of $n=pq$ reflects the strength
 - 700-bit n factored in 2007
 - 768 bit n factored in 2009
- RSA encryption/decryption speed is quadratic in key length
- 1024 bit for minimal level of security today (roughly 80 bit security)
 - likely to be breakable in near future
- Minimal 2048 bits recommended for current usage (roughly 112 bit security)
- NIST suggests 15360-bit RSA keys are equivalent in strength to 256-bit
- Factoring is easy to break with quantum computers

RSA Encryption & IND-CPA Security

- The RSA assumption, which assumes that the RSA problem is hard to solve, ensures that the plaintext cannot be fully recovered.
- Plain RSA does not provide IND-CPA security.
 - For Public Key systems, the adversary has the public key, hence the initial training phase is unnecessary, as the adversary can encrypt any message he wants to.
 - How to break IND-CPA security?
 - How to use it more securely?

Real World Usage of Public Key Encryption

- Often used to encrypt a symmetric key
 - To encrypt a message M under an RSA public key (n,e) , generate a new AES key K , compute $[K^e \bmod n, \text{AES-CBC}_K(M)]$
- Alternatively, one can use random padding.
 - E.g., compute $(M \parallel r)^e \bmod n$ to encrypt a message M with a random value r
 - More generally, uses a function $F(M,r)$, and encrypts as $F(M,r)^e \bmod n$
 - From $F(M,r)$, one should be able to recover M
 - This provides randomized encryption

Digital Signatures: The Problem

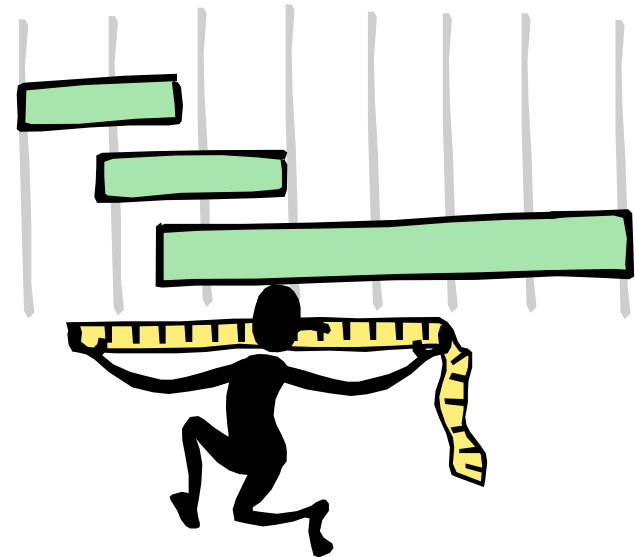
- Consider the real-life example where a person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
- Contracts are valid if they are signed.
- Signatures provide **non-repudiation**.
 - ensuring that a party in a dispute 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?**

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) 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:
 - 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. $ed \equiv 1 \pmod{\Phi}$

Public key: (e, n)

used for verification

Private key: d ,

used for generation

RSA Signatures with Hash (cont.)

Signing message M

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

Verifying signature S

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

Non-repudiation

- Nonrepudiation is the assurance that someone cannot deny something. Typically, nonrepudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.
- Can one deny a digital signature one has made?
- Does email provide non-repudiation?

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.

Coming Attractions ...

- User authentication

