

# Cryptography CS 555

## Lecture 17



Notions of Security for Public-Key Encryption Schemes, and Digital Signatures

# Review

- Discrete Log
  - assumed to be hard in many settings
- The Diffie Hellman key establishment protocol
  - The CDH problem and the DDH problem
- ElGamal Encryption
  - public key:  $(p, g, \beta = g^a)$
  - private key:  $a$
  - encryption:  $\langle \gamma = g^k \bmod p, \delta = m \beta^k \rangle$ 
    - $k$  should be different for each encryption, **why?**

# Lecture Outline

- Security notions for public-key encryption



# Notions of Security for Public Key Encryption

- Information theoretically secure
  - no information is leaked in the ciphertext
  - not achievable in public key encryption
- Attack objectives
  - Key recovery (KR)
  - Plaintext recovery (PR)
  - Indistinguishability (IND)
    - no “usable” information is leaked in the ciphertext
  - Non-Malleability (NM)

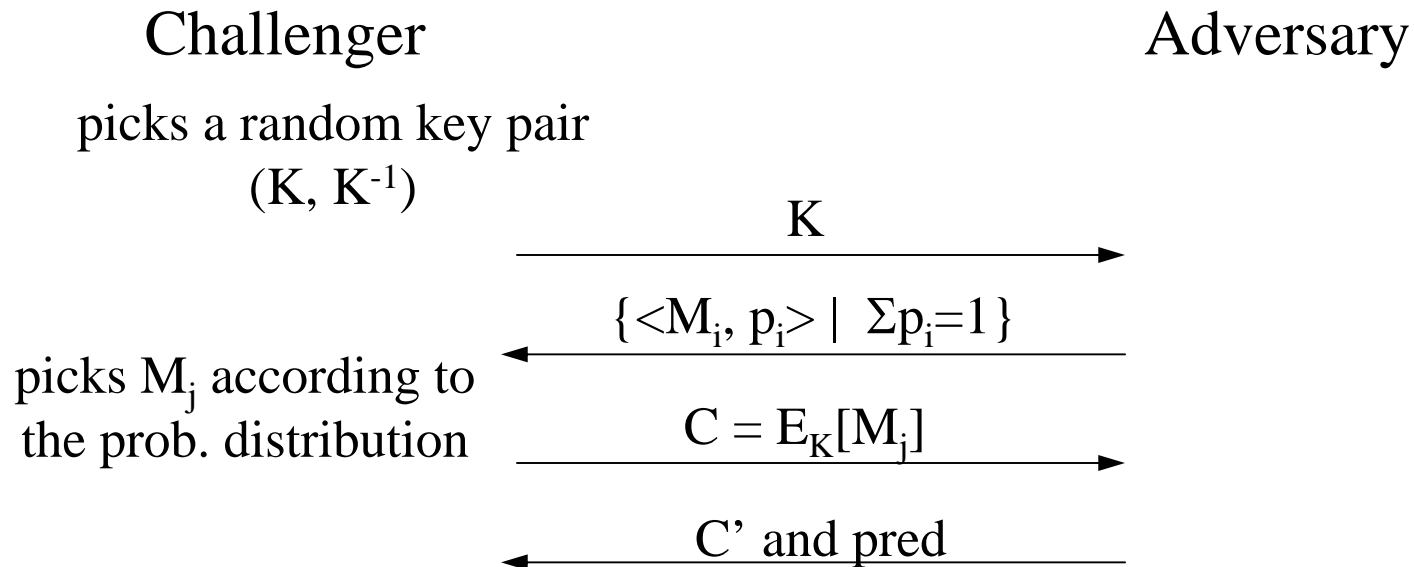
# Why Indistinguishability?

- Computational version of perfect secrecy
- Plaintext recovery is not good enough
  - If the space of messages is small, then adversary can try to encrypt each of them, to determine which message is encrypted in a ciphertext
  - Even if plaintext recovery is difficult, there may be bits that can be easily computed.
    - E.g., in the case of RSA, maybe among the 1024 bits, half of them can be computed, the encryption is still secure under the plaintext recovery attacks

# Why Non-Malleable Cryptography?

- What is non-malleable: Require that it is difficult to modify a ciphertext so that the corresponding plaintext changes in a meaningful manner
- Consider an example
  - CS wants to build a new building and invites bids from two contractors; bids are encrypted in CS's public key; if the encryption scheme is malleable, then one may be able to always bid 1 higher than the other one's bid

# Precise Definition of the NM



Attacker wins game if  
 $\text{pred}(D_K[C'], M_j)$  holds

# RSA is Malleable

- An encryption scheme is NM-secure if the adversary's probability of success does not increase in a non-negligible way
- RSA is malleable
  - Given  $M$ , and  $C = M^e \pmod n$
  - Compute  $C' = 2^e C \pmod n$ , then  $D_K[C'] = 2M_j \pmod n$
- Similarly, Rabin is malleable

# Training of the Adversary

- No training (-CPA)
- Chosen-ciphertext training before committing to the attack (-CCA)
  - chooses a ciphertext, and the challenger decrypts
- Chosen-ciphertext training after committing to the attack (-CCA2)
  - in the IND case, the ciphertexts must be different from the one the adversary needs to attack
  - also known as adaptive chosen ciphertext attacks

# Insecurity Against CCA2 attackers

- RSA is insecure against PR-CCA2 attackers
  - the adversary is given a ciphertext  $C$ , it computes  $C' = 2^e C \bmod n$ , and ask the Challenger to decrypt  $C'$ ; the adversary is then able to decrypt  $C$

# Relationships Between the Security Notions

- Let ATK be CPA, CCA, or CCA2
- $\text{NM-ATK} \geq \text{IND-ATK} > \text{PR-ATK} > \text{KR-ATK}$ 
  - $\text{NM-CCA2} = \text{IND-CCA2}$
  - $\text{NM-CCA} > \text{IND-CCA}$
  - $\text{NM-CPA} > \text{IND-CPA}$

# 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.
- Can we have a similar service in the electronic world?

# Digital Signatures

- Digital Signature: a data string which associates a message with some originating entity.
- Digital Signature Scheme: for each key, there is a **SECRET signature generation algorithm** and a **PUBLIC verification algorithm**
- Services provided:
  - Authentication
  - Data integrity
  - Non-Repudiation (MAC does not p

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

# Adversarial Goals

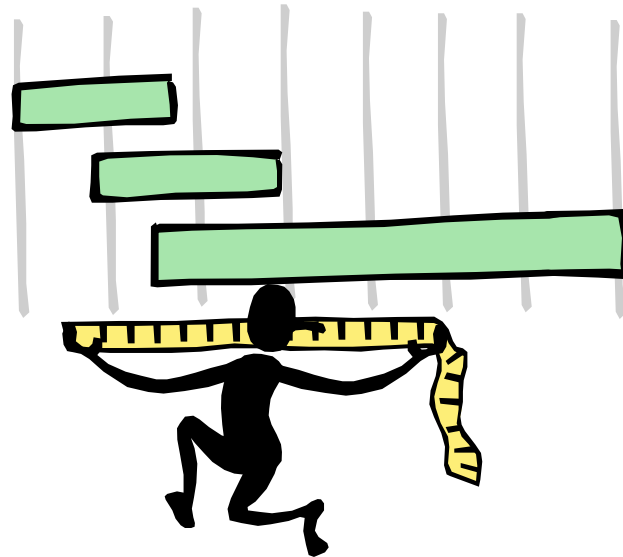
- **Total break**: adversary is able to find the secret for signing, so he can forge then any signature on any message.
- **Selective forgery**: adversary is able to create valid signatures on a message chosen by someone else, with a significant probability.
- **Existential forgery**: adversary can create a pair (message, signature), s.t. the signature of the message is valid.
- A signature scheme can not be perfectly secure; it can only be computationally secure.
- Given enough time and adversary can always forge Alice's signature on any message.

# Attack Models for Digital Signatures

- **Key-only attack:** Adversary knows only the verification function (which is supposed to be public).
- **Known message attack:** Adversary knows a list of messages previously signed by Alice.
- **Chosen message attack:** Adversary can choose what messages wants Alice to sign, and he knows both the messages and the corresponding signatures.

# 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 Signature

## 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:**  $(e, n)$

**Secret key:**  $d, p$  and  $q$  must also remain secret

# RSA Signature (cont.)

## Signing message $M$

- $M$  must verify  $0 < M < n$
- Use private key ( $d$ )
- compute  $S = M^d \bmod n$

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

## 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.

# RSA Signature (cont.)

## Example of forging

- Attack based on the multiplicative property of property of RSA.

$$y_1 = \text{sig}_K(x_1)$$

$$y_2 = \text{sig}_K(x_2), \text{ then}$$

$$\text{ver}_K(x_1x_2 \bmod n, y_1y_2 \bmod n) = \text{true}$$

- So adversary can create the valid signature  $y_1y_2 \bmod n$  on the message  $x_1x_2 \bmod n$
- This is an existential forgery using a known message attack.

# El Gamal Signature

## Key Generation (as in ElGamal encryption)

- Generate a large random prime  $p$  such that DLP is infeasible in  $Z_p$  and a generator  $g$  of the multiplicative group  $Z_p$  of the integers modulo  $p$
- Select a random integer  $a$ ,  $1 \leq a \leq p-2$ , and compute

$$y = g^a \text{ mod } p$$

- Public key is  $(p; g ; \beta = g^a)$
- Private key is  $a$ .
- Recommended sizes: 1024 bits for  $p$  and 160 bits for  $a$ .

# ElGamal Signature (cont.)

## Signing message M

- Select random  $k$ ,  $1 \leq k \leq p-1$ ,  $k \in \mathbb{Z}_{p-1}^*$

- Compute

$$r = g^k \bmod p$$

$$s = k^{-1}(h(M) - ar) \bmod (p-1)$$

- Signature is:  $(r,s)$
- Size of signature is double size of  $p$

NOTE:  $h$  is a hash function



# ElGamal Signature (cont.)

Signature is:  $(r, s)$

$$r = g^k \text{ mod } p$$

$$s = k^{-1}(h(M) - ar) \text{ mod } (p-1)$$

## Verification

- Verify that  $r$  is in  $Z_{p-1}^*$  :  $1 \leq r \leq p-1$
- Compute
$$v_1 = \beta^r r^s \text{ mod } p$$
$$v_2 = g^{h(M)} \text{ mod } p$$
- Accept iff  $v_1 = v_2$

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

# ElGamal Signature (cont.)

## Security of ElGamal signature

- Weaker than DLP
- $k$  must be unique for each message signed
- Hash function  $h$  must be used, otherwise easy for an existential forgery attack
- $0 < r < p$  must be checked, otherwise easy to forge a signature on any message if an valid signature is available.

# Digital Signature Algorithm (DSA)

Specified as FIPS 186

## Key generation

- Select a prime  $q$  of 160-bits
- Choose  $0 \leq t \leq 8$
- Select  $2^{511+64t} < p < 2^{512+64t}$  with  $q \mid p-1$
- Let  $\alpha$  be a generator of  $Z_p^*$ , and set  
 $g = \alpha^{(p-1)/q} \bmod p$
- Select  $1 \leq a \leq q-1$
- Compute  $\beta = g^a \bmod p$

Public key:  $(p, q, g, \beta)$

Private key:  $a$

# DSA

## Signing message M:

- Select a random integer  $k$ ,  $0 < k < q$
- Compute
$$k^{-1} \bmod q$$
$$r = (g^k \bmod p) \bmod q$$
$$s = k^{-1} (h(M) + ar) \bmod q$$
- Signature:  $(r, s)$

Note: FIPS recommends  
the use of SHA-1 as hash function.



# DSA

Signature: (r, s)

$$r = (g^k \bmod p) \bmod q$$

$$s = k^{-1} (h(M) + ar) \bmod q$$

## Verification

- Verify  $0 < r < q$  and  $0 < s < q$ , if not, invalid
- Compute
$$w = s^{-1} \bmod q$$
$$u_1 = w \bullet h(m) \bmod q,$$
$$u_2 = r \bullet w \bmod q$$
$$v = (g^{u_1} \beta^{u_2} \bmod p) \bmod q$$
- Valid iff  $v = r$

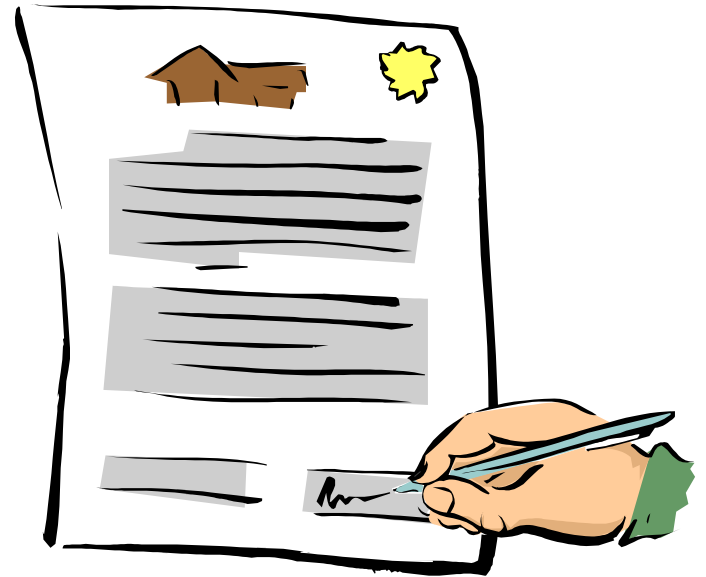
# One-Time Digital Signatures

- One-time digital signatures: digital schemes used to sign, at most one message; otherwise signature can be forged.
- A new public key is required for each signed message.
- Advantage: signature generation and verification are very efficient and is useful for chipcards, where low computation complexity is required.

# Lamport One-time Signature

## To sign one bit:

- Choose as secret keys  $x_1, x_2$ 
  - $x_1$  represents '0'
  - $x_2$  represents '1'
- public key:
  - $y_1 = h(x_1)$ ,
  - $y_2 = h(x_2)$ .
- Signature is  $h(x_1)$  if the message is  $x_1$  or  $h(x_2)$  for  $x_2$



# Merkle One-Time Signature

## Key generation: to sign a $n$ -bit message

- Select  $t = n + 1 + \lfloor \lg n \rfloor$  and random secret strings  $k_1, k_2, \dots, k_t$  each of bitlength  $l$

- Compute

$$v_i = h(k_i), \quad 1 \leq i \leq t,$$

$h$  is a pre-image resistant hash:

$$\{0,1\}^* \rightarrow \{0,1\}^l$$

- Public key:  $(v_1, v_2, \dots, v_t)$
- Private key:  $(k_1, k_2, \dots, k_t)$

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

# Merkle One-Time Signature

## Sign message $m$ of bit-length $n$

- Compute  $c$ , the binary representation of number of 0's in  $m$ ;

$$w = m \parallel c = (a_1 a_2 \dots a_t)$$

- Determine the coordinate positions

$$i_1 < i_2 \dots < i_u \text{ in } w, \text{ s.t. } a_{i_j} = 1, 1 \leq j \leq u$$

- Let  $s_j = k_{i_j}$ ,  $1 \leq j \leq u$
- Signature is:  $(s_1, \dots, s_u)$

# Merkle One-Time Signature

## Verification

- Compute  $c$ , the binary representation of number of 0's in  $m$ ;  
 $w = m || c = (a_1 a_2 \dots a_t)$
- Determine the coordinate positions  $i_1 < i_2 \dots < i_u$  in  $w$ , s.t.  $a_{i_j} = 1$
- Let  $s_j = k_{i_j}$ ,  $1 \leq j \leq u$
- Valid iff  $v_{i_j} = h(s_j)$  for all  $j$ ,  $1 \leq j \leq u$

# Summary

- Notions of security for public key encryption
- Digital Signatures
  - Notions of security
  - RSA
  - ElGamal
  - DSA
  - One-time signatures
    - Lamport
    - Merkle



# Next ...

- Authentication & Identification Protocols

