The Uses of Encryption

Jun Li

lijun@cs.uoregon.edu

1

Learning Objectives

Learn:

- Public Key Encryption Characteristics
- Cryptographic hash functions
- Key exchange
- Digital signature
- Certificates

Key Proliferation Problem in a Symmetric Cryptosystem

- *n* users require $n*(n-1)/2$ keys
  - to allow every pair of users to communicate secretly

- In asymmetric cryptosystem, *n* users require $n*2$ keys
  - Every user can simply use the public key of the recipient to encrypt a message

Public Key Encryption Characteristics

(1) $P = D(k_{PRIV}, E(k_{PUB}, P))$

(2) $P = D(k_{PUB}, E(k_{PRIV}, P))$

Assume (k_{PUB}, k_{PRIV}) belongs to Alice

anyone can know k_{PUB} and do $E(k_{PUB}, *)$, but

only Alice should know k_{PRIV} and can do $E(k_{PRIV}, *)$.

From (1): Only Alice can decrypt $E(k_{PUB}, P)$ to get P.

From (2): Q: Is this a message from Alice?

A: Decrypt it with Alice's k_{PUB}.

Comparing Secret Key and Public Key Encryption

|	Secret Key (Symmetric)	Public Key (Asymmetric)
Number of keys	1	2
Protection of key	Must be kept secret	The private key must be secret; public key can go public
Best uses	Cryptographic workhorse	Key exchange; authentication
Key distribution	Out-of-band	Public key can be used
Speed	Fast (bit operations)	Typically 10,000 times slower (modular exponentiation using * /)

Four Applications of Encryption

- Encryption is not just for secrecy
  - How about integrity?

- Four brilliant uses
  - Cryptographic hash functions
  - Key exchange
  - Digital signature
  - Certificates
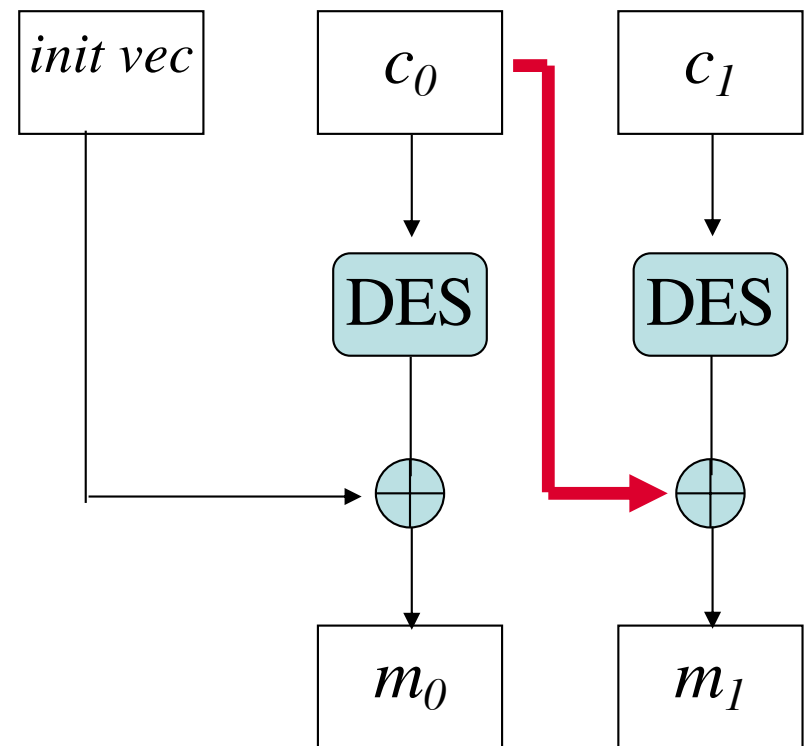
Cryptographic Hash Functions

- Hash, checksum, message digest
  - All synonyms
- One-way functions
  - Much easier to compute than their inverses
  - Any change to even a single bit will alter the result
- Cryptographic functions (DES, AES) can make good one-way functions
  - As outsiders do not know the key

Cipher Block Chaining (CBC)

Encipherment
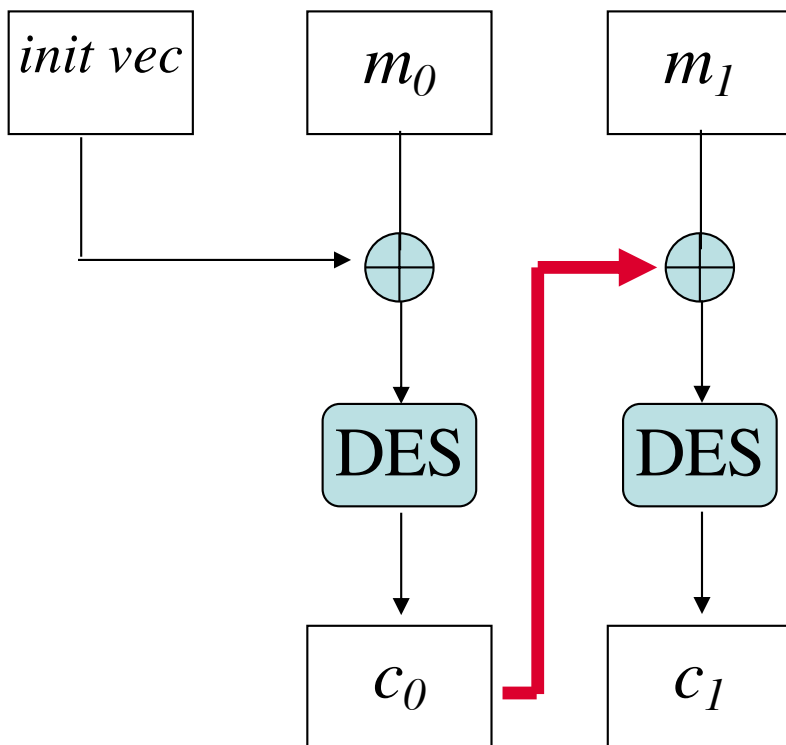
Decipherment

- Use the last cipher block (c_n) as hash
- Can be any block cipher

8

Popular Cryptographic Hash Functions

- **MD4, MD5** (message digest)
  - By Ron Rivest and RSA Laboratories
  - 128-bit digest

- **SHA/SHS** (Secure Hash Algorithm or Standard)
  - Digest size is between 128 and 512 bits, depending on the version of SHA

Key Exchange

- Problem: $S(k_{\text{PRIV-S}}, k_{\text{PUB-S}})$, $R(k_{\text{PRIV-R}}, k_{\text{PUB-R}}) \Rightarrow K?$
- Method 1: S picks K and sends $E(k_{\text{PRIV-S}}, K)$ to R
  - Does it work?
- Method 2: $E(k_{\text{PUB-R}}, K)$ to R?
  - Does it work?
- Method 3: $E(k_{\text{PUB-R}}, E(k_{\text{PRIV-S}}, K))$

Key Exchange

- Problem: S, R, field size n, starting number $g => K?$
- S and R each think up a secret number, s and r
- S sends R g^s, R sends to S g^r
- S computes $(g^r)^s$, R computes $(g^s)^r$
- Secret $K = (g^r)^s = (g^s)^r$

Is This Method Safe?

- Attacks can intercept g^s, g^r
- If he also knows g, he can get K as:

$s = \ln(g^s)/\ln(g)$ and $k = g^{rs}$

A Technical Detail: **mod n**

- Discrete Logarithm Problem: Find a value of x such that

$$Y = g^{\textcolor{red}{x}} \bmod n$$

for a given Y, g, and prime n.

- Difficulty increases exponentially as n increases
- So what S and R send to each other are $\underline{g^s \bmod n}$ and $\underline{g^r \bmod n}$, respectively.
- This is called **Diffie-Hellman key exchange protocol**

Digital Signature

- A **digital signature** is a construct that authenticates both the *origin* and *contents* of a message in a manner that is provable to a disinterested third party.

- Provides a service of nonrepudiation

Case Study 1: Is $\{m\}\,k$ a Signature?

- Alice sends Bob

$$m\ \{m\}\,k$$

k is the shared secret key between Alice and Bob;

$\{m\}\,k$ is another form of $\mathbf{E}(m, k)$.

- Bob deciphers $\{m\}\,k$ and compares with m
  - Thus verifying that the message is from Alice
  - And the contents are not modified
- But Bob cannot prove to a third party that $m\ \{m\}\,k$ is not created by Bob himself!
  - So $\{m\}\,k$ is not a signature of m

Case Study 2: Is $\{m\}\, d_{Alice}$ a Signature?

- Alice sends Bob

$$m \; \{m\}\, d_{Alice}$$

d_{Alice} is the private key of Alice

- Bob deciphers $\{m\}\, d_{Alice}$ and compares with m
  - Thus verifying that the message is from Alice
  - And the contents are not modified
- A judge can verify $\{m\}\, d_{Alice}$ is indeed signed by Alice if:

$$m = \{\{m\}\, d_{Alice}\}\, e_{Alice}$$

Public Key Signature

- Instead of using $\{m\}\, d_{Alice}$, Alice actually signs the message as

$$\{h(m)\}\, d_{Alice}$$

where h is a cryptographic hash function

- And sends Bob

$$m\; \{h(m)\}\, d_{Alice}$$

- Q: how does Bob verifies the signature?

Certificates

- A **certificate** is a token that binds an identity to a cryptographic key

$$C_{Alice} = \{e_{Alice} \mathbin{/\!/} Alice \mathbin{/\!/} T\} \, d_{Cathy}$$

$$C_{Alice} = e_{Alice} \mathbin{/\!/} Alice \mathbin{/\!/} T \mathbin{/\!/} signed \ by \ Cathy$$
$$\{h\{e_{Alice} \mathbin{/\!/} Alice \mathbin{/\!/} T\}\} \, d_{Cathy}$$

- A **certificate authority** (CA) issues certificates

20

Certificate Verification

- Suppose Bob knows Cathy's public key e_{Cathy}

- When Bob obtains C_{Alice},
  - Deciphers C_{Alice} using e_{Cathy}
  - Then knows that Cathy is vouching that e_{Alice} is Alice's public key, issued at time T
  - If Bob trusts what Cathy believes
  - Then Bob knows e_{Alice} is Alice's public key

But, Bob Has to Know e_{Cathy}

- Two solutions
  - Chain of certificates
    - There is another certificate for e_{Cathy}
  - Merkle's Tree Authentication Scheme
    - Eliminates Cathy's signature

Chain of Certificates

- *X.509: Centralized trust model*
  - Tree-like CA hierarchy employed
  - Every node has a local CA
  - A local CA has its CA, the parent
  - The parent CA has its parent
  - And there is a root CA
  - Together, a tree of CAs!
- *PGP: Decentralized trust model*
  - Anyone can sign a certificate to form a web of trust
  - Currently also supports X.509

Merkle's Tree Authentication Scheme

- All <id, public key> pairs are stored in a file
- A cryptographic hash function creates a digest of the file
  - The digest is known to the public
- If any pair is changed, it will be detected
  - Since the digest will be different

24

Y_1 | Cathy | 01389234789357
Y_2 | Bob | 89230378597823
Y_3 | Alice | 72384927894027
Y_4 | David | 32748902378240

digest

25

Digest Algorithm

- A tree-based algorithm

Y_1	Cathy	01389234789357
Y_2	Bob	89230378597823
Y_3	Alice	72384927894027
Y_4	David	32748902378240

h(1,4)

h(1,2)   *h(3,4)*

h(1,1)   *h(2,2)*   *h(3,3)*   *h(4,4)*

Y_1 Y_2 Y_3 Y_4

26
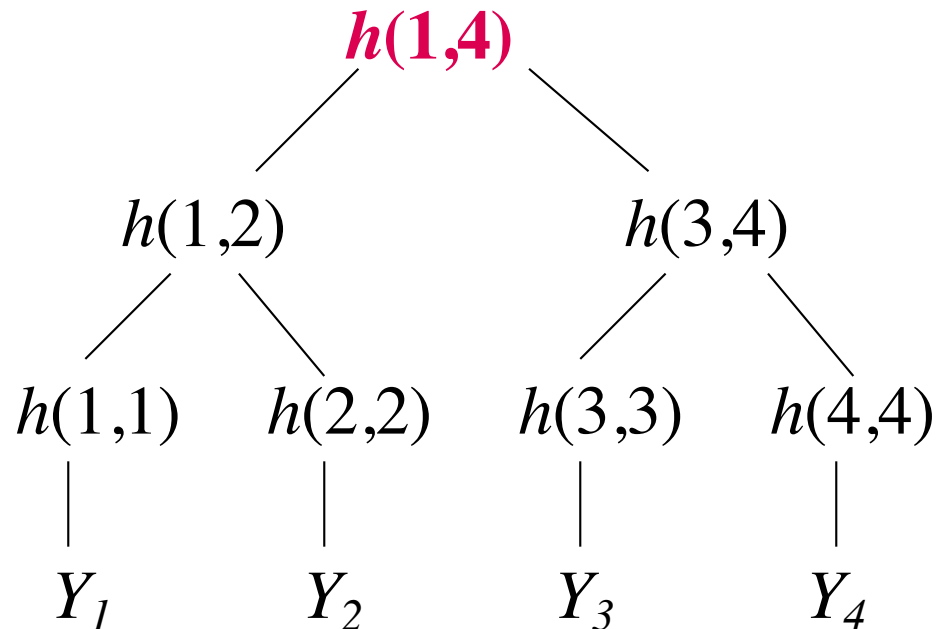
Public Key Verification

- How can Bob verify whether or not Alice's public key is 72384927894027.

- Bob will re-compute the digest, and compare that with the publicly known value of the digest
  - If Alice's public key is not 72384927894027, a discrepancy will be detected

Y_1	Cathy	01389234789357
Y_2	Bob	89230378597823
Y_3	Alice	72384927894027
Y_4	David	32748902378240

$h(1,4)$

$h(1,2)$ $h(3,4)$

$h(1,1)$ $h(2,2)$ $h(3,3)$ $h(4,4)$

Y_1 Y_2 Y_3 Y_4

Alice, 72384927894027

28
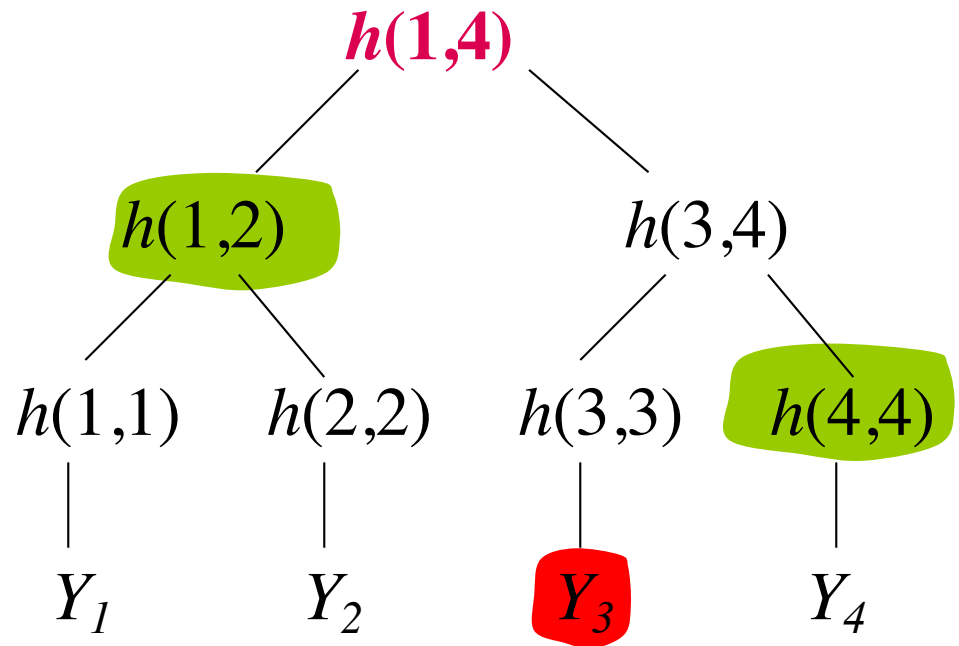
Authentication Path

- Bob knows Y_3
- Bob needs to know $h(4,4)$ and $h(1,2)$
- Y_3, $h(4,4)$ and $h(1,2)$ is the **authentication path** for Alice's public key
  - They can put together and used for certifying Alice's public key
- *This is a certificate without signature!*