

UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Introduction to Cryptography
ECE 597XX/697XX

Part 6

Introduction to
Public-Key Cryptography

Israel Koren

ECE597/697 Koren Part.6 .1

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

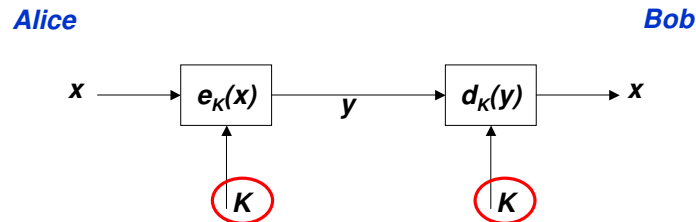
Content of this part

- ◆ **Symmetric Cryptography Revisited**
- ◆ **Principles of Asymmetric Cryptography**
- ◆ **Practical Aspects of Public-Key Cryptography**
- ◆ **Important Public-Key Algorithms**
- ◆ **Essential Number Theory for Public-Key Algorithms**

ECE597/697 Koren Part.6 .2

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Symmetric Cryptography revisited



Two properties of symmetric (secret-key) crypto-systems:

- ♦ The **same secret key K** is used for encryption and decryption
- ♦ Encryption and Decryption are very similar (or even identical) functions

ECE597/697 Koren Part.6 .3

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Symmetric Cryptography: Analogy



Safe with a strong lock, only Alice and Bob have a copy of the key

- Alice encrypts $\rightarrow$ locks message in the safe with her key
- Bob decrypts $\rightarrow$ uses his copy of the key to open the safe

ECE597/697 Koren Part.6 .4

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

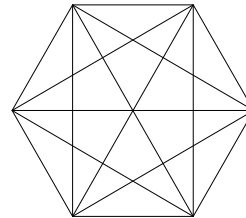
Symmetric Cryptography: Shortcomings

- ♦ Symmetric algorithms, e.g., AES or 3DES, are very secure, fast & widespread **but**:
- ♦ Key distribution problem: The secret key must be **transported securely**
- ♦ Number of keys: In a network, each pair of users requires an individual key
 $\rightarrow n$ users in the network require $\frac{n \cdot (n-1)}{2}$ keys, each user stores $(n-1)$ keys

Example:

6 users (nodes)

$$\frac{6 \cdot 5}{2} = 15 \text{ keys (edges)}$$



- ♦ Alice or Bob can cheat each other, because they have identical keys.
Example: Alice can claim that she never ordered a TV online from Bob (he could have fabricated her order). To prevent this: „non-repudiation“

ECE597/697 Koren Part.6 .5

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Idea behind Asymmetric Cryptography



New Idea:

Use the „good old mailbox“ principle:

Everyone can drop a letter

But: Only the owner has the correct key to open the box



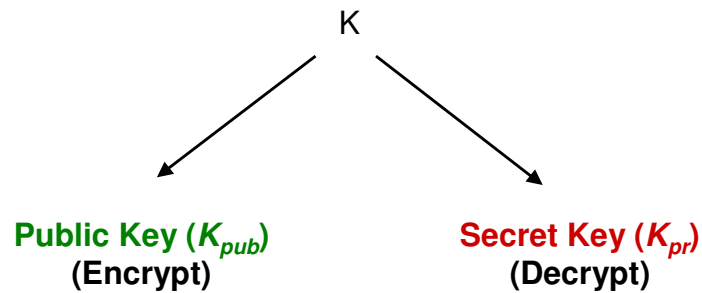
1976: first publication of such an algorithm by Diffie and Hellman, and also by Merkle.

ECE597/697 Koren Part.6 .6

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Asymmetric (Public-Key) Cryptography

Principle: "Split up" the key



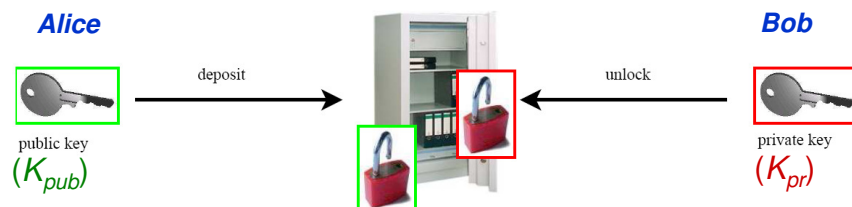
During the key generation, a key pair K_{pub} and K_{pr} is computed

ECE597/697 Koren Part.6 .7

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Asymmetric Cryptography: Analogy

Safe with public lock and private lock:



- Alice deposits (encrypts) a message with the - *not secret* - public key K_{pub}
- Only Bob has the - *secret* - private key K_{pr} to retrieve (decrypt) the message

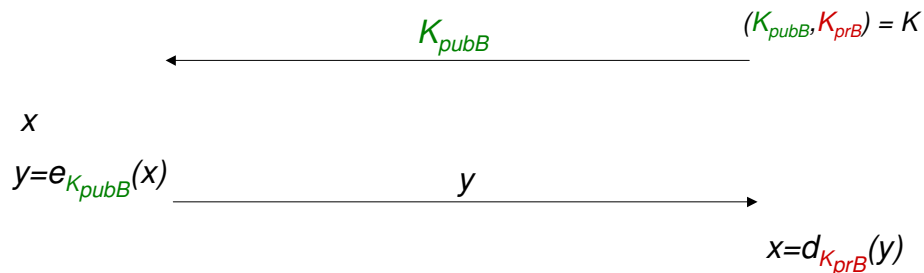
ECE597/697 Koren Part.6 .8

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Basic Protocol for Public-Key Encryption

Alice

Bob



→ Key Distribution Problem solved *

*) at least for now; public keys need to be authenticated, cf. Chap. 13 of *Understanding Cryptography*

ECE597/697 Koren Part.6 .9

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Security Mechanisms of Public-Key Cryptography

Here are main mechanisms that can be realized with asymmetric cryptography:

- ♦ **Key Distribution** (e.g., Diffie-Hellman key exchange, RSA) without a pre-shared secret (key)
- ♦ **Nonrepudiation and Digital Signatures** (e.g., RSA, DSA or ECDSA) to provide message integrity
- ♦ **Identification** using challenge-response protocols with digital signatures
- ♦ **Encryption** (e.g., RSA / ElGamal)
- ♦ **Disadvantage:** Computationally very intensive (1000 times slower than symmetric Algorithms)

ECE597/697 Koren Part.6 .10

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Basic Key Transport Protocol 1/2

In practice: **Hybrid systems**, incorporating asymmetric and symmetric algorithms

1. **Key exchange** (for symmetric schemes) and digital signatures are performed with (slow) asymmetric algorithms
2. **Encryption** of data is done using (fast) symmetric ciphers, e.g., block ciphers or stream ciphers

ECE597/697 Koren Part.6 .11

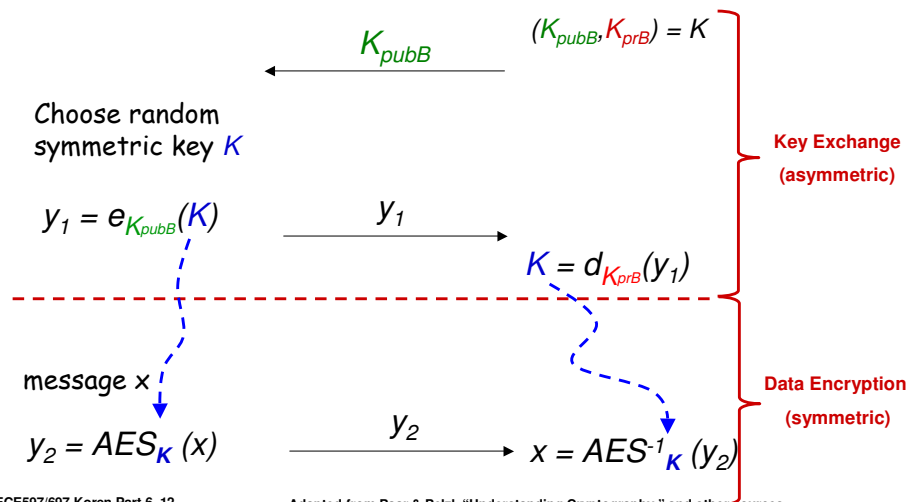
Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Basic Key Transport Protocol 2/2

Example: Hybrid protocol with AES as the symmetric cipher

Alice

Bob



ECE597/697 Koren Part.6 .12

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

How to build Public-Key Algorithms

Asymmetric schemes are based on a „one-way function“ $f()$:

- ♦ Computing $y = f(x)$ is computationally easy
- ♦ Computing $x = f^{-1}(y)$ is computationally very hard

One way functions are based on mathematically hard problems.
Three main families:

- **Factoring integers** (RSA, ...):
Given a composite integer n , find its prime factors
(Multiply two primes: easy)
- **Discrete Logarithm** (Diffie-Hellman, ElGamal, DSA, ...):
Given a , y and m , find x such that $a^x = y \bmod m$
(Exponentiation a^x : easy)
- **Elliptic Curves (EC)** (ECDH, ECDSA): Generalization of discrete logarithm

Note: The problems are considered mathematically hard, but no proof exists (so far).

ECE597/697 Koren Part.6 .13

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Key Lengths and Security Levels

<i>Symmetric</i>	<i>ECC</i>	<i>RSA, DL</i>	<i>Remark</i>
64 Bit	128 Bit	≈ 700 Bit	Only short term security (a few hours or days)
80 Bit	160 Bit	≈ 1024 Bit	Medium security (except attacks from big governmental institutions)
128 Bit	256 Bit	≈ 3072 Bit	Long term security (without quantum computers)

- The exact complexity of RSA (factoring) and DL (Index-Calculus) is difficult to estimate
- The development of quantum computers would probably be the end for ECC, RSA & DL

ECE597/697 Koren Part.6 .14

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Euclidean Algorithm 1/2

- ♦ Compute the **greatest common divisor** $\gcd(r_0, r_1)$ of two integers r_0 and r_1
- ♦ $\gcd$ is easy for small numbers:
 1. factor r_0 and r_1
 2. $\gcd$ = highest common factor
- ♦ Example:

$$r_0 = 84 = \textcircled{2} \cdot 2 \cdot \textcircled{3} \cdot 7$$

$$r_1 = 30 = \textcircled{2} \cdot \textcircled{3} \cdot 5$$

→ The $\gcd$ is the product of all common prime factors:
 $2 \cdot 3 = 6 = \gcd(30, 84)$
- ♦ **But:** Factoring is very complicated for large numbers

ECE597/697 Koren Part.6 .15

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Euclidean Algorithm 2/2

- ♦ Observation: $\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1)$
- Core idea:
 - Reduce the problem of finding the $\gcd$ of two given numbers to that of the $\gcd$ of two smaller numbers
 - Repeat process recursively
 - The final $\gcd(r_i, 0) = r_i$ is the answer to the original problem

Example: $\gcd(r_0, r_1)$ for $r_0 = 27$ and $r_1 = 21$

21			6
6	6	6	3
3	3		

$$\gcd(27, 21) = \gcd(1 \cdot 21 + 6, 21) = \gcd(21, 6)$$

$$\gcd(21, 6) = \gcd(3 \cdot 6 + 3, 6) = \gcd(6, 3)$$

$$\gcd(6, 3) = \gcd(2 \cdot 3 + 0, 3) = \gcd(3, 0) = 3$$

Very efficient method even for long numbers: complexity grows *linearly* with number of bits

$973 = 3 \cdot 301 + 70$	$\gcd(973, 301) = \gcd(301, 70)$
$301 = 4 \cdot 70 + 21$	$\gcd(301, 70) = \gcd(70, 21)$
$70 = 3 \cdot 21 + 7$	$\gcd(70, 21) = \gcd(21, 7)$
$21 = 3 \cdot 7 + 0$	$\gcd(21, 7) = \gcd(7, 0) = 7$

ECE597/697 Koren Part.6 .16

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Extended Euclidean Algorithm (EEA)

- ♦ Extend the Euclidean algorithm to find **modular inverse** of $r_1 \bmod r_0$
- ♦ EEA computes s, t , and the gcd: $\gcd(r_0, r_1) = s \cdot r_0 + t \cdot r_1$
- ♦ $\gcd(r_0, r_1) = 1$ in order for the inverse to exist
 $s \cdot r_0 + t \cdot r_1 = 1 = \gcd(r_0, r_1)$
- ♦ Reduce the equation mod r_0 : $s \cdot 0 + t \cdot r_1 \equiv 1 \bmod r_0$
 $r_1 \cdot t \equiv 1 \bmod r_0$
- ♦ t is the inverse of $r_1 \bmod r_0$
- ♦ EEA uses recursive formulae to calculate s and t in each step
 - Express current remainder r_i as $r_i = s_i r_0 + t_i r_1$.
 - Last iteration:
 $r_1 = \gcd(r_0, r_1) = s_1 r_0 + t_1 r_1 = s_0 + t_1$

ECE597/697 Koren Part.6 .17

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Extended Euclidean Algorithm - Example

$\gcd(973, 301) = 7$
 $r_0 = 973; r_1 = 301$

$s = 13; t = -42$

$973 = 3 \cdot 301 + 70$	$70 = [1]r_0 + [-3]r_1$
$301 = 4 \cdot 70 + 21$	$21 = 301 - 4 \cdot 70$ $= r_1 - 4(1r_0 - 3r_1)$ $= [-4]r_0 + [13]r_1$
$70 = 3 \cdot 21 + 7$	$7 = 70 - 3 \cdot 21$ $= (1r_0 - 3r_1) - 3(-4r_0 + 13r_1)$ $= [13]r_0 + [-42]r_1$
$21 = 3 \cdot 7 + 0$	

$$\gcd(973, 301) = 7 = [13]973 + [-42]301 = 12649 - 12642$$

EEA can be expressed
 using recursive
 formulae for s_i, t_i

$$\begin{array}{ll} s_0 = 1 & t_0 = 0 \\ s_1 = 0 & t_1 = 1 \end{array}$$

```

1 DO
1.1  i = i + 1
1.2  r_i = r_{i-2} mod r_{i-1}
1.3  q_{i-1} = (r_{i-2} - r_i) / r_{i-1}
1.4  s_i = s_{i-2} - q_{i-1} \cdot s_{i-1}
1.5  t_i = t_{i-2} - q_{i-1} \cdot t_{i-1}
  WHILE r_i \neq 0
  
```

ECE597/697 Koren Part.6 .18

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

EEA - calculating a modular inverse

Example: Calculate the modular inverse of 12 mod 67:

♦ Using EEA we obtain $-5 \cdot 67 + 28 \cdot 12 = 1$

♦ Hence 28 is the inverse of 12 mod 67.

$\gcd(67, 12) = \gcd(12, 7) = \gcd(5, 2) = \gcd(2, 1)$
 $67 = 12 \cdot 5 + 7 \Rightarrow 7 = (1)67 + (-5)12$
 $12 = 7 + 5 \Rightarrow 5 = 12 - 7 = (-1)67 + (6)12$
 $7 = 5 + 2 \Rightarrow 2 = 7 - 5 = (2)67 + (-11)12$
 $5 = 2 \cdot 2 + 1 \Rightarrow 1 = 5 - 2 \cdot 2 = (-5)67 + (28)12$

i	q_{i-1}	r_i	s_i	t_i
2	5	7	1	-5
3	1	5	-1	6
4	1	2	2	-11
5	2	1	-5	28

$s_0 = 1 \quad t_0 = 0$
 $s_1 = 0 \quad t_1 = 1$

```

1 DO
1.1  i = i + 1
1.2  r_i = r_{i-2} mod r_{i-1}
1.3  q_{i-1} = (r_{i-2} - r_i) / r_{i-1}
1.4  s_i = s_{i-2} - q_{i-1} * s_{i-1}
1.5  t_i = t_{i-2} - q_{i-1} * t_{i-1}
WHILE r_i ≠ 0

r_1 mod r_0 = 12 mod 67
            
```

♦ **Check:** $28 \cdot 12 = 336 \equiv 1 \pmod{67}$ ✓

ECE597/697 Koren Part.6 .19

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Euler's Phi Function 1/2

- ♦ *Important for public-key systems, e.g., RSA:*
 Given the set of the m integers $\{0, 1, 2, \dots, m-1\}$,
 How many numbers in the set are **relatively prime to m** ?
- ♦ **Answer:** Euler's Phi function $\phi(m)$ (totient function)
- ♦ Example: sets $\{0, 1, \dots, 5\}$ ($m=6$), and $\{0, 1, \dots, 4\}$ ($m=5$)

$\gcd(0, 6) = 6$	$\gcd(0, 5) = 5$
$\gcd(1, 6) = 1$ ←	$\gcd(1, 5) = 1$ ←
$\gcd(2, 6) = 2$	$\gcd(2, 5) = 1$ ←
$\gcd(3, 6) = 3$	$\gcd(3, 5) = 1$ ←
$\gcd(4, 6) = 2$	$\gcd(4, 5) = 1$ ←
$\gcd(5, 6) = 1$ ←	

→ $\phi(5) = 4$; $\phi(6) = 2$

- ♦ Testing one gcd per number in the set is extremely slow for large m .

ECE597/697 Koren Part.6 .20

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Euler's Phi Function 2/2

- ♦ If canonical factorization of m known: $m = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_n^{e_n}$ (where p_i primes and e_i positive integers)
- ♦ then calculate Phi according to the relation

$$\Phi(m) = \prod_{i=1}^n (p_i^{e_i} - p_i^{e_i-1})$$

- ♦ Phi especially easy for $e_i = 1$, e.g., $m = p \cdot q \Rightarrow \Phi(m) = (p-1) \cdot (q-1)$
- ♦ Examples: $m=6=3 \cdot 2 \Rightarrow \Phi(6) = (3-1)(2-1)=2$
 $m = 899 = 29 \cdot 31$:
 $\Phi(899) = (29-1) \cdot (31-1) = 28 \cdot 30 = 840$
- ♦ Note: Finding $\Phi(m)$ is computationally easy if factorization of m is known (otherwise the calculation of $\Phi(m)$ is computationally very hard for large numbers)

ECE597/697 Koren Part.6 .21

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Fermat's Little Theorem

- ♦ Given a prime p and an integer a : $a^p \equiv a \pmod{p}$
- ♦ Can be rewritten as $a^{p-1} \equiv 1 \pmod{p}$
 $a^{a^{p-2}} \equiv 1 \pmod{p}$
- ♦ Use: Find modular inverse, if p is prime.
- ♦ Comparing with definition of the modular inverse
 $a \cdot a^{-1} \equiv 1 \pmod{m}$
 the modular inverse modulo a prime p is
 $a^{-1} \equiv a^{p-2} \pmod{p}$
- Example: $a = 2, p = 7$ $a^{p-2} = 2^5 = 32 \equiv 4 \pmod{7}$
 verify: $2 \cdot 4 \equiv 1 \pmod{7}$ ✓
- ♦ Fermat's Little Theorem works only modulo a prime p

ECE597/697 Koren Part.6 .22

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Euler's Theorem

- ♦ Generalization of Fermat's little theorem to any integer modulus

- ♦ Given two relatively prime integers a and m :

- ♦ Example: $m=18, a=5$ $a^{\Phi(m)} \equiv 1 \pmod{m}$

1. Calculate Euler's Phi Function

$$\Phi(18) = \Phi(3^2 \cdot 2) = (3^2 - 3^1)(2^1 - 2^0) = 6$$

2. Verify Euler's Theorem

$$5^{\Phi(18)} = 5^6 = 25^3 = 7^3 \pmod{18} \Rightarrow 7^3 = 343 = 18 \cdot 19 + 1 = 1 \pmod{18}$$

- ♦ Fermat's little theorem = special case of Euler's Theorem

- ♦ for a prime p : $\Phi(p) = (p^1 - p^0) = p - 1$

→ Fermat: $a^{\Phi(p)} = a^{p-1} \equiv 1 \pmod{p}$

ECE597/697 Koren Part.6 .23

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Lessons Learned

- ♦ Public-key algorithms have capabilities that symmetric ciphers don't have, in particular digital signature and key establishment functions.
- ♦ Public-key algorithms are **computationally intensive** (a nice way of saying that they are *slow*), and hence are poorly suited for bulk data encryption.
- ♦ Only three families of public-key schemes are widely used. This is considerably fewer than in the case of symmetric algorithms.
- ♦ The **extended Euclidean algorithm** allows us to compute modular inverses quickly, which is important for almost all public-key schemes.
- ♦ **Euler's phi function** gives us the number of elements smaller than an integer n that are relatively prime to n . This is important for the RSA.

ECE597/697 Koren Part.6 .24

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources