COMPSCI 4CR3 - Applied Cryptography

Jake Doliskani

McMaster University

Introduction to public-key cryptography

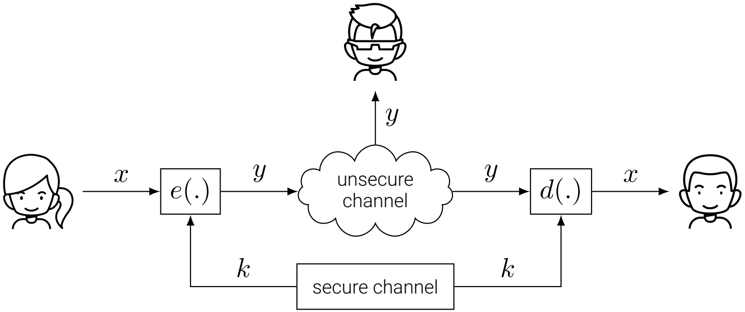
This lecture

- Symmetric cryptography revisited
- Principles of asymmetric cryptography
- Practical aspects of public-key cryptography
- Well-known public-key algorithms
- Some number theory

Symmetric vs. Asymmetric

- Symmetric-key and private-key are the same; Asymmetric-key and public-key are the same. We will use them interchangeably.
- Private-key cryptography has been used for thousands of years. Public-key cryptography was introduced by Diffie, Hellman and Merkle in 1976.
- In private-key cryptography, parties use the same key, while in public-key cryptography, parties use different types of keys.

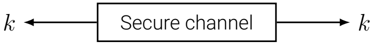
Symmetric-key Cryptography (Revisited)



- The same secret key is used for encryption and decryption
- The encryption and decryption function are very similar

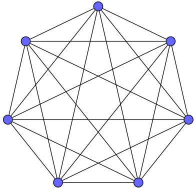
Symmetric-key Cryptography (shortcomings)

- Key distribution:
  There must be a secure channel to transport the key!

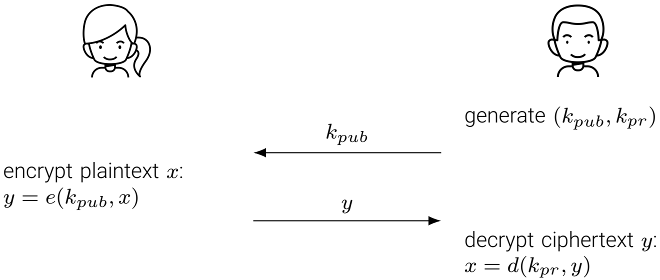


- No protection against a dishonest Alice or Bob: Nonrepudiation

- Number of keys:
  For a network of n users, the number of key pairs can be as large as $n(n-1)/2$



Public-key cryptography



- Bob's key is a pair (k_{pub}, k_{pr}) of public and private keys
- k_{pr} is only known to Bob, while everyone knows k_{pub}
- Alice uses the public key to encrypt, Bob uses the private key to decrypt

A simple key transport

generate (k_{pub}, k_{pr})

$\xleftarrow{\quad k_{pub} \quad}$

choose a random h
$y = e(k_{pub}, h)$

$\xrightarrow{\quad y \quad}$

decrypt ciphertext y:
$h = d(k_{pr}, y)$

encrypt message x
$z = \mathrm{AES}(h, x)$

$\xrightarrow{\quad z \quad}$

decrypt ciphertext z:
$x = \mathrm{AES}^{-1}(h, z)$

How to build public-key systems

One-way function

A function f is one-way if it is easy to compute but hard to invert.

- Easy means computationally feasible: there is a polynomial time algorithm for evaluating f
- Hard means computationally infeasible: there is no polynomial time algorithm for inverting f

Do one-way functions exist?

Security Mechanisms

We can do many things with public-key cryptography:

- Encryption
  - basic functionality
- Key exchange
  - e.g., Diffie–Hellman key exchange
- Nonrepudiation
  - using digital signature algorithms
- Identification
  - using challenge-and-response protocols together with digital signatures

Problem: authentication

The man-in-the-middle attack:

It is possible if public keys are not authenticated.

A solution: certificates

- Certificates bind the identity of a user to their public key

Underlying hard problems

Current/Old

- Integer-Factorization
  - RSA
- Discrete Logarithm
  - Diffie–Hellman key exchange
  - Digital Signature Algorithms
  - Elgamal encryption
- Elliptic Curves DL
  - Diffie–Hellman key exchange (ECDH)
  - Digital Signature Algorithm (ECDSA)

New

- Lattice-based
  - LWE and RLWE constructions,
  - Many others
- Hash-based
  - Digital signatures, e.g., SPHINCS+
- Isogeny-based
  - Digital signatures
  - Encryption
- Code-based
  - Encryption, e.g., Classic McEliece, HQC

Key lengths and security levels

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024	3072	7680	15360
Discrete logarithm	DH, DSA, Elgamal	1024	3072	7680	15360
Elliptic curves	ECDH, ECDSA	160	256	384	512
Symmetric-key	AES, 3DES	80	128	192	256

The Euclidean Algorithm

Question: how do we compute $\gcd(r_0, r_1)$?

Answer 1: factor r_0, r_1, and collect the greatest common divisor

Example: $\gcd(130, 52) = 26$, since $130 = 2 \cdot 5 \cdot 13$ and $52 = 2^2 \cdot 13$.

- This only works for small numbers because factoring is hard!
- In cryptography, we usually deal with large numbers.

Answer 2: use the Euclidean algorithm

Trick: suppose $r_0 \geq r_1$, then $\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$.

The Euclidean Algorithm

- The identity $\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$ reduces the input size.
- We can repeat until one of the inputs is zero

Example 1: $r_0 = 130, r_1 = 52$
$\gcd(130, 52) = \gcd(52, 26)$
$\gcd(52, 26) = \gcd(26, 0) = 26$

Example 2: $r_0 = 27, r_1 = 21$
$\gcd(27, 21) = \gcd(21, 6)$
$\gcd(21, 6) = \gcd(6, 3)$
$\gcd(6, 3) = \gcd(3, 0) = 3$

The Euclidean Algorithm

Input: positive integers r_0, r_1 such that $r_0 \geq r_1$
Output: $\gcd(r_0, r_1)$

$i = 1$

do

$i = i + 1$

$r_i = r_{i-2} \bmod r_{i-1}$

while $r_i \neq 0$

return r_{i-1}

- There are unique numbers s, t such that $\gcd(r_0, r_1) = sr_0 + tr_1$
- With a little change to the above algorithm, we can compute s, t as well

The Extended Euclidean Algorithm

Input: positive integers r_0, r_1 such that $r_0 \geq r_1$
Output: $\gcd(r_0, r_1)$, and integers s, t such that $\gcd(r_0, r_1) = sr_0 + tr_1$

$s_0 = 1, t_0 = 0$

$s_1 = 0, t_1 = 1$

$i = 1$

do

$i = i + 1$

$r_i = r_{i-2} \bmod r_{i-1}$

$q_{i-1} = (r_{i-2} - r_i)/r_{i-1}$

$s_i = s_{i-2} - q_{i-1}s_{i-1}$

$t_i = t_{i-2} - q_{i-1}t_{i-1}$

while $r_i \neq 0$

return $\gcd(r_0, r_1) = r_{i-1}, s = s_{i-1}, t = t_{i-1}$

Euler's phi function

$\varphi(m)$: the number of integers smaller than m that are relatively prime to m

Example:
$\varphi(20) = 8$; the coprime numbers are $1, 3, 7, 9, 11, 13, 17, 19$.
$\varphi(8) = 4$; the coprime numbers are $1, 3, 5, 7$.

Theorem

Suppose m has the prime factorization $m = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$, where the p_i are distinct primes and the e_i are positive integers. Then

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

Euler's theorem

Theorem

Let $a, m > 0$ be integers such that $\gcd(a, m) = 1$. Then

$$a^{\varphi(m)} \equiv 1 \bmod m.$$

Example 1:
$a = 5, m = 21$
$\varphi(21) = (3-1)(7-1) = 12$, $5^{12} \equiv 1 \bmod 12$

Example 2:
$a = 12, m = 25$
$\varphi(25) = 5^2 - 5 = 20$, $12^{20} \equiv 1 \bmod 25$