IS53012B/A Computer Security

Dr Ida Pu

Room 10, 29 St James
Goldsmiths, University of London

2018-19 (since 2007)

Part I

Public Key cryptosystems

Primes

prime A positive integer (whole number) that has exactly TWO factors, namely 1 and itself. e.g. 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, $\cdots$

composite A whole number greater than 1 that is not a prime. e.g. 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, $\ldots$

A composite number can be written as a product of its prime factors, e.g.

$$15 = 3 \times 5, \quad 21 = 3 \times 7, \cdots$$

Fermat's Little Theorem

If p is a prime number and a is any number between 1 and $p-1$ inclusive, then

$$a^{p-1} \mod p = 1$$

Example

Let $a = 2, p = 7$, we have $2^6 \mod 7 = 64 \mod 7 = 1$
and
Let $a = 3, p = 7$, $3^6 \mod 7 = 729 \mod 7 = 1$

This is *not* true in general.

Fermat's Little Theorem

Example

Let $a = 7, p = 5$, we have $7^4 \mod 5 = 2401 \mod 5 = 1$ $(a > p - 1)$
Let $a = 5, p = 5$, we have $5^4 \mod 5 = 0 \neq 1$ $(a > p - 1)$
Let $a = 5, p = 6$, we have $5^5 \mod 6 = 5 \neq 1$ ($a < p - 1$, but p is not a prime)

This can be used to decide if a given number n is composite or probably a prime.

Factorisation

factorisation Given an integer n, there is an efficient algorithm to determine whether n is composite or prime.

factorisation problem Determining the factors of a large composite number is hard. This becomes the basis of the RSA cryptosystem.

The basis of the RSA cryptosystem

Easy

- Determining whether a large number is a prime or composite is easy
- Multiplying 2 large numbers together is easy

Hard

- Factorising a large number which is the product of 2 large primes (i.e. retrieval of the original prime factors) is very difficult.

To find the factors of a composite number n which is the product of 2 large primes, and has about 640 binary bits (approximately 200 decimal digits) is an impossible task using current computer power.

RSA

Everyone x has two keys: $K_{private}(x)$, and $K_{public}(x)$, for x only and for public respectively

$\boxed{Alice}$ Encryption:

Alice wants to send Bob message m. Alice

1. looks up Bob's $K_{public}(Bob) = (e, n)$
2. computes $c = m^e \mod n$ and sends the value of c to Bob

$\boxed{Bob}$ Decryption:

Bob has two keys:
$K_{private}(Bob) = d$
$K_{public}(Bob) = (e, n)$

Upon receipt of c from Alice, Bob

1. uses his $K_{private}(Bob) = d$
2. computes $m = c^d \mod n$

RSA Example

Example

Alice:
$m = 5$
$c = 5^7 \mod 11 = 3$

Bob:
$K_{public}(Bob) = (e, n) = (7, 11)$
$K_{private}(Bob) = d = 3$
$3^3 \mod 11 = 27 \mod 11 = 5$

Example

Alice:
$m = 5$
$c = 5^{13} \mod 77 = 26$

Bob:
$K_{public}(Bob) = (e, n) = (13, 77)$
$K_{private}(Bob) = d = 37$
$26^{37} \mod 77 = 5 \mod 77 = 5$

Does this work? How?

Two issues

1. Decryption: Consider $c = m^e \mod p$. If c, e and p are given, can we determine m easily?
2. Key generation: How are the K_{public} and $K_{private}$ chosen?

Solving a problem

Given

- a prime number p
- a number $m \in [1, p-1]$ (between 1 and $p-1$ inclusive)
- another number e, also $\in [1, p-1]$

We compute $c = m^e \mod p$.
If c, e and p are given, can we determine m easily?
Yes if we take the following steps:

1. Find a number d such that $e * d \mod p - 1 = 1$
2. Compute $c^d \mod p = m$.

Why it works

We find d such that $e * d \mod p - 1 = 1$, which means that, for some value of q,

$$e * d = k(p-1) + 1 \qquad \text{(definition of mod)}$$

We compute

$$\begin{aligned}
c^d \mod p &= (m^e)^d \mod p \\
&= m^{ed} \mod p \\
&= m^{k(p-1)+1} \mod p \\
&= [m^{(p-1)}]^k * m^1 \mod p && \text{(Fermat's)} \\
&= 1^k * m \mod p \\
&= m \mod p \\
&= m && (0 < m < p)
\end{aligned}$$

Why it works

- This works because of Fermat's Little Theorem.
- Since p is a prime we have
  1. $a^{p-1} \mod p = 1$ for any $a \in (0, p)$ and so
  2. $c^{k(p-1)} = 1 \mod p$ leaves us with the answer m.
- BUT if the modulus is not a prime number then the method does not work.
  In general $a^{n-1} \mod n \neq 1$ if n is a composite (not prime),
  e.g. $5^5 \mod 6 = 5 \neq 1$
- HOWEVER, we could make the method for finding m work if we knew the number r such that $a^r \mod n = 1$
- If a and n are co-prime then there will be such a number r and there is a way to find it.
  Two numbers are *co-prime* if they have no common factors,
  e.g. 6 and 35, where $6 = 1 \times 6, 2 \times 3; 35 = 1 \times 35, 5 \times 7$

Finding r

- In order to find r such that $a^r \mod n = 1$, we have to factorise n and find all of its prime factors.
- If $n = p * q$ where p and q are primes then we have

$$r = (p-1) * (q-1)$$

Important to note

- It is easy to determine whether a large number is prime or composite.
- It is easy to compute the product of two large primes $n = p * q$.
- Setting $r = (p-1) * (q-1)$ we have
  $m^r \mod n = 1$, for all m that co-prime with n (i.e. having no factor in common with n).
- Given e (co-prime with r), it is easy to determine d such that
  $(e * d) \mod r = 1$
- It is easy to compute $m^e \mod n$
- If $c = m^e \mod n$ then $m = c^d \mod n$ and it is easy to compute $c^d \mod n$ if we know d.
- We can only find d if we can find r; we can only find r if we can factorise n. But factorising n is hard.

Basis of the RSA

- The holder of the public key knows p and q, can find r, then d; and can compute $cd \mod n$ to find m.
- No-one else knows p and q, so they cannot find r nor d, and so they cannot decrypt c to get m.
- There is no known way to recover m which is not equivalent to factorising n.

Key Generation

Bob

1. generates two large primes p and q (each with approximately 100 decimal digits).
2. He computes $n = p * q$
3. He computes $r = (p-1) * (q-1)$
4. He chooses a large random number e which is between 1 and r which has no factor in common with r.
5. He computes the private key d by solving the equation $(e * d) \mod r = 1$.
6. He can now carefully dispose the values of p, q and r.
7. Bob keeps d private but publishes the value of the pair (e, n). This is his public key.
   i.e. $K_{private}(Bob) = d, K_{public}(Bob) = (e, n)$.

RSA Encryption and Decryption

Alice wants to send Bob message m. She

1. looks up Bob's public key pair (e, n) .
2. computes $c = m^e \mod n$ and sends the value of c to Bob

Upon receipt of c from Alice, Bob

1. uses his private key d
2. computes $m = c^d \mod n$

Note:

- The message m must be smaller than n. Alice breaks her message up into blocks each with a value less than n and encrypts each of these blocks individually.
- The public key can be used by anyone wishing to send Bob a message. Bob does not need a separate key pair for each correspondent.

Part II

El Gamal

The Discrete Log Problem I

This public key cryptosystem is based on the difficulty of solving the **Discrete Logarithm Problem** (DLP).

Definition

DLP for the prime p *Given a prime p and values g and y, find x such that $y = g^x \mod p$.*

For a small value of p, it is easy to solve a DLP by trial and error or exhaustive search.

The Discrete Log Problem II

Example

Given $p = 11$, $g = 2$ and $y = 9$, we can try different values of x until we reach the correct solution for $2^x \mod 11 = 9$

However, for a large value of p, e.g. 100 or so decimal digits, it is impossible to solve a DLP using current technologies.
If we can solve the DLP then we can crack El Gamal public key cryptosystems.

El Gamal Key Generation

To generate public and private keys, Bob

1. chooses a large random prime p
2. finds a generator $g \mod p$
3. chooses a random number $d \in (1, p-1)$
4. computes $y = g^d \mod p$
5. So, Bob's public key is (p, g, y) and the private key is d.

El Gamal Encryption

If Alice wants to send Bob a message, she looks up Bob's public key (p, g, y) and breaks the message up into blocks with each block less than p. Then for each message block m
Alice

1. generates a random number k between 1 and $p-1$.
2. computes
    1. $r = g^k \mod p$,
    2. $x = y^k \mod p$, and
    3. $c = (m * x) \mod p$
3. sends Bob the values (r, c), and carefully discards (k, x).

El Gamal Decryption

Upon receipt of the ciphertext (r, c) from Alice, Bob follows:

1. computes $r^d \mod p = x$
2. solves $c = (m * x) \mod p$ to find the value of m.
    1. finds x^{-1}, the inverse of $x \mod p$
    2. $m = (c * x^{-1}) \mod p$

Note: Only Bob can do this because only Bob knows the value of the private key d.

Example

Alice wishes to send Bob message $m = 13$ so she

1. looks up Bob's public keys
2. generates a random number $k = 8$ and computes $r = g^k \mod p = 2^8 \mod 19 = 9$, $x = y^k \mod p = 15^8 \mod 19 = 5$, and $c = (m * x) \mod p = (13 * 5) \mod 19 = 8$.
3. sends $(r, c) = (9, 8)$

Bob (Nelson's example, p74)

1. chooses $p = 19$, find $g = 2$ (exponent 18), generates a random number $d = 11$ and determines $y = g^d = 2^{11} = 15$, so he generates his public key $(p, g, y) = (19, 2, 15)$, and private key $d = 11$
2. Upon receipt of (r, c), Bob computes
    1. $x = r^d \mod p = 9^{11} \mod 19 = 5$
    2. solves equation $c = (x * m) \mod p$, i.e. $8 = (5 * m) \mod 19$), and finds $m = 13$.

Comparison between RSA and El Gamal

RSA

- Security based on the difficulty of the *factorisation problem*.
- The ciphertext is just one value c which is roughly the same size as the message m.
- The encryption and decryption algorithms are the same (modular exponentiation).
- RSA is a patented algorithm.

El Gamal

- Security based on the difficulty of the *discrete log problem*.
- The ciphertext is two values c and r and so is a double size of the message m.
- The encryption and decryption algorithms are different (although both take about the same time to perform).
- El Gamal has no patent. This gives it a financial advantage over RSA.