MA080G Cryptography Summary Block 2

Viktor Rosvall

April 23, 2019

Public-key cryptography

One of the problems public-key cryptography solves is the **key distribution problem** by using a distributed *public* key for encryption and a *private* key for decryption.

This works because encryption is done using a *One-way function.*

One-way function: a function $f()$ is a one-way function if:

1. $y = f(x)$ is computaitonally easy
2. $x = f^{-1}(y)$ is computaitonally impossible

This means that even if the public key used to encrypt a message is know, it can't be decrypted without the private key. [1]

Key Distribution Example [1]

Let's say Alice wants to send x to Bob. Both Alice and Bob have a public and private key-pair: $k = (k_{\text{pup}}, k_{\text{priv}})$.

Alice encrypts x using Bob's public key b_{pup}, as:

$$y = e_{b_{\text{pup}}}(x)$$

where e is a one-way function. Now Bob can decrypt the received message y using his private key b_{priv} and retrieve x, as:

$$x = d_{b_{\text{priv}}}(y)$$

We can send any data securely using this method. It's common to send key's for symmetric ciphers such as AES, since it's computationally heavy to use these computations.

With Digital Signatures [2]

A digital signature is a must when using public-key cryptography. Because encryption keys are public, Oscar is able to do a **man-in-the-middle attack**, i.e., encrypt a message using *Bob's public* key, while faking it's origin as Alice.

This can be fixed using **digital signatures**. The sender creates a digital signature by using one's *private* key to encrypt (instead of decrypt) a plaintext. Then the sender encrypts the already encrypted plaintext using the receiver's public key, with a message like "Hey, it's bob".

The receiver decrypts the first layer using his private key and sees the message "Hey, it's bob". At this point the receiver expects the message to be sent from Bob. Thus the receiver will then decrypts the second layer using the sender's (presumably Bob's) public key.

RSA [3]

RSA works by using public and private key-pairs. The private key (in it's basic form) is two very large prime numbers p and q. The public key is the product of p and q.

RSA is only secure because we don't have good algorithms for factorizing large integers. [4]

Fermat's Little Theorem

Fermat's Little Theorem is useful in primality testing and in public-key cryptography. It can also be used for find the inverse of an integer a modulo a prime. [5]

Theorem: let a be an integer and p be a prime, then:

$$a^p \equiv a \pmod{p}$$

This can also be rewritten as:

$$a^{p-1} \equiv 1 \pmod{p}$$

If p is a prime then the inverse of a can be calculated as:

$$a^{-1} \equiv a^{p-2} \pmod{p}$$

Proof using modular arithmetic [6]

Let's assume a is a positive integer, not divisible by prime p. If we write down the sequence of numbers in modulo p

$$a, 2a, 3a, ..., (p-1)a$$

and after reducing each integer modulo p, we get the resulting sequence of numbers

$$1, 2, 3, ..., p-1.$$

Which means the two sequences are congruent modulo p

$$a, 2a, 3a, ..., (p-1) \equiv 1, 2, 3, ..., p-1 \pmod{p}$$

Which is the same as

$$a^{p-1}(p-1)! \equiv (p-1)! \pmod{p}.$$

After canceling out the sequence of both sides we get

$$a^{p-1} \equiv 1 \pmod{p}$$

Example

Let $a = 2$ and $p = 7$. The sequence of numbers thus is

$$2, 4, 6, 8, 10, 12$$

and after reducing each integer modulo p, we get

$$2, 4, 6, 1, 3, 5$$

reordered as

$$1, 2, 3, 4, 5, 6.$$

The two sequences are also congruent

$$2, 4, 6, 1, 3, 5 \equiv 1, 2, 3, 4, 5, 6 \pmod{p}$$
$$2^6 6! \equiv 6! \pmod{p}$$
$$2^6 \equiv 1 \pmod{p}$$

Euler's generalization [5]

Euler's generalization of Fermat's Little Theorem allows any integer modulo m, instead of just modulo prime.

‘ **Euler's Theorem:** let a and m be co-prime integers, i.e., $\gcd(a, m) = 1$, then:

$$a^{\Phi(m)} \equiv 1 \pmod{m}$$

Example

Let $a = 3$ and $m = 8$. The $\gcd(3, 8) = 1$.
First we need to calculate $\Phi(8)$.

$$\Phi(8) = \Phi(2^3) = 2^3 - 2^2 = 4.$$

Now we can use Euler's theorem:

$$3^{\Phi(8)} = 3^4 = 81 \equiv 1 \pmod{8}$$

Compute the order of elements in Zn [7]

To compute the order or *cardinality* of Z_n, the group needs to be finite. Z_n is defined as the set of integers $\{0, 1, 2, ..., n-1\}$. The order of Z_n is denoted as $|Z_n| = n$.

Z_n^* is defined as the set of positive integers, co-prime to n. The order of Z_n^* is denoted as $Z_n^* = \Phi(n)$

The order of an element a, of a group G, is denoted as: ord(a). The element a is called the *generator* or a *primitive element* of the group G, if ord(a) = $|G|$.

Example

Determine the order of $a = 2$ in Z_7. We calculate this by doing modular arithmetic on 2^n, until $2^n \equiv 1 \pmod{7}$

$$\begin{aligned}
2^1 &= 2 &\equiv 2 \pmod{7} \\
2^2 &= 4 &\equiv 4 \pmod{7} \\
2^{\mathbf{3}} &= 8 &\equiv 1 \pmod{7}
\end{aligned}$$

So from the last line, we can see $2^{\mathbf{3}} = 8 \equiv 1 \pmod{7}$, thus ord(2) = 3 in Z_7. This also means that $a = 2$ in Z_7 is not a primitive element.

Carmichael's lambda-function

Miller-Rabin probabilistic primality test

Pollard's p-1 factorization method

References

[1] C. Paar, J. Pelzl, *Understanding Cryptography*. 2010 ed. Springer., Chapter 6.1

[2] P. J. Cameron, *Notes on cryptography*.
http://www.maths.qmul.ac.uk/ pjc/notes/crypt.pdf Chapter 4.4

[3] C. Paar, J. Pelzl, *Understanding Cryptography*. 2010 ed. Springer., Chapter 7.2-7.3

[4] C. Paar, J. Pelzl, *Understanding Cryptography*. 2010 ed. Springer., Chapter 6.2.3

[5] C. Paar, J. Pelzl, *Understanding Cryptography*. 2010 ed. Springer., Chapter 6.3.4

[6] Wikipedia, ”Proofs of Fermat’s little theorem”,
https://en.wikipedia.org/wiki/Proofs_of_Fermat%27s_little_theorem
18-04-2019

[7] C. Paar, J. Pelzl, *Understanding Cryptography*. 2010 ed. Springer., Chapter 8.2.2