

Coding and Cryptography Notes

Daniel da Silva

November 2016

1 Theorem and Algorithms

1.1 Euclids's Algorithm

1.1.1 Statement

To find the GCD of two numbers, say a and b ($b < a$), do the following:

$$\begin{aligned}a &= q_0b + r_0 \\ b &= q_1r_0 + r_1 \\ r_0 &= q_2r_1 + r_2 \\ r_1 &= q_3r_2 + r_3 \\ &\vdots \\ &\vdots \\ &\vdots\end{aligned}$$

Since the remainder is decreasing with every step but cannot be negative, there will eventually be a remainder r_N which is equal to zero, at which point the algorithm stops. The remainder r_{N-1} is the GCD of a and b . If ($a < b$) swap a and b before beginning the algorithm.

1.1.2 Euclid's Algorithm for finding modular inverses

Assume we have two integers a and b which are co-prime. We want to find the inverse of $a \pmod{b}$, denoted \bar{a} . We know from the definition of the modular inverse that $a\bar{a} \equiv 1 \pmod{b}$. Hence our task is to solve the equation $ax + by = 1$ for x and y which will give us $x = \bar{a}$.

We do this by using Euclid's Algorithm and stopping when the remainder is equal to 1:

$$\begin{aligned}
 a &= q_0 b + r_0 \\
 b &= q_1 r_0 + r_1 \\
 r_0 &= q_2 r_1 + r_2 \\
 r_1 &= q_3 r_2 + r_3 \\
 &\vdots \\
 &\vdots \\
 &\vdots \\
 r_{N-3} &= q_{N-1} r_{N-2} + r_{N-1}
 \end{aligned}$$

We know that $r_{N-1} = 1$ because a and b are co-prime and hence their GCD is 1. We now re-write the equations and substitute back until we end with an equation of the form $1 = ax + by$ and we will have $\bar{a} \equiv x \pmod{b}$.

1.2 Chinese Remainder Theorem

1.2.1 Statement

The Chinese Remainder Theorem states that the system of congruences,

$$\begin{aligned}
 x &\equiv a_1 \pmod{m_1} \\
 x &\equiv a_2 \pmod{m_2} \\
 &\vdots \\
 &\vdots \\
 &\vdots \\
 x &\equiv a_r \pmod{m_r}
 \end{aligned}$$

where the modulo m_1, m_2, \dots, m_r are all pairwise coprime has a unique solution modulo M given by

$$x \equiv a_1(M_1 \overline{M_1}) + a_2(M_2 \overline{M_2}) + \dots + a_r(M_r \overline{M_r}) \pmod{M}$$

where

- $M = m_1 m_2 \dots m_r$
- $M_i = \frac{M}{m_i}$
- $M_i \overline{M_i} \equiv 1 \pmod{m_i}$

1.2.2 Example 1

Let

$$\begin{aligned}
 x &\equiv 0 \pmod{3} \\
 x &\equiv 3 \pmod{4} \\
 x &\equiv 4 \pmod{5}
 \end{aligned}$$

Find $x \pmod{M}$

$$\bullet M = m_1 m_2 m_3 = (3)(4)(5) = 60$$

- $$\begin{aligned}
M_1 &= \frac{M}{m_1} = \frac{60}{3} = 20 \\
M_2 &= \frac{M}{m_2} = \frac{60}{4} = 15 \\
M_3 &= \frac{M}{m_3} = \frac{60}{5} = 12
\end{aligned}$$

- $$\begin{aligned}
M_1 \overline{M_1} &\equiv 1 \pmod{m_i} \\
\Rightarrow 20 \overline{M_1} &\equiv 1 \pmod{3}
\end{aligned}$$

Using Euclid's Algorithm :

Solving $1 = 20\overline{M_1} - 3y$

$$20 = 3(6) + 2$$

$$3 = 2(1) + 1$$

Substituting back :

$$3 = (20 - 3(6))(1) + 1$$

$$3 = 20(1) - 3(6) + 1$$

$$1 = 3(7) + 20(-1)$$

Hence
$$\begin{aligned}
\overline{M_1} &\equiv -1 \pmod{3} \\
\Rightarrow \overline{M_1} &\equiv 2 \pmod{3}
\end{aligned}$$

1.3 Modular Exponentiation Algorithm

1.3.1 Statement

To find the least postive residue of $a^n \pmod{m}$ where n is a large number do the following:

- Express n in binary to get $n = (b_0)2^0 + (b_1)2^1 + (b_2)2^2 + \dots + (b_k)2^k$ ($b_i \in [0, 1]$)
- Calculate the least positive residues modulo n for $a^{2^0}, a^{2^1}, \dots, a^{2^k}$
- Multiply the least positive residues whose corresponding bit is 1 in the first step.
- Find the least positive residue of the result above.

1.3.2 Example

What is the least positive residue of $3^{231} \pmod{49}$?

- 231 in binary:

$$231 = (1)2^7 + (1)2^6 + (1)2^5 + (0)2^4 + (0)2^3 + (1)2^2 + (1)2^1 + (1)2^0$$
Hence $231 = (11100111)_2$
- Least positive residues modulo 49 for $3^{2^0}, 3^{2^1}, \dots, 3^{2^7}$:

$$\begin{aligned}
b_0 = 1 \quad 3^{2^0} &\equiv 3^1 \equiv 3 \pmod{49} \\
b_1 = 1 \quad 3^{2^1} &\equiv 3^2 \equiv 9 \pmod{49} \\
b_2 = 1 \quad 3^{2^2} &\equiv 3^4 \equiv 9^2 \equiv 32 \pmod{49}
\end{aligned}$$

$$\begin{array}{ll}
b_3 = 0 & 3^{2^3} \equiv 3^8 \equiv 32^2 \equiv 44 \pmod{49} \\
b_4 = 0 & 3^{2^4} \equiv 3^{16} \equiv 44^2 \equiv 25 \pmod{49} \\
b_5 = 1 & 3^{2^5} \equiv 3^{32} \equiv 25^2 \equiv 37 \pmod{49} \\
b_6 = 1 & 3^{2^6} \equiv 3^{64} \equiv 37^2 \equiv 46 \pmod{49} \\
b_7 = 1 & 3^{2^7} \equiv 3^{128} \equiv 46^2 \equiv 9 \pmod{49}
\end{array}$$

- Multiply results where $b_i = 1$:
 $(3)(9)(32)(37)(46)(9) = 13234752$
- Find least positive residue of above result:
 $3^{2^{31}} \equiv (3^{2^7})(3^{2^6})(3^{2^5})(3^{2^2})(3^{2^1})(3^{2^0}) \equiv 13234752 \equiv 48 \pmod{49}$

1.3.3 Explanation of Algorithm

We know from the definition of the mod function that:

$$\begin{aligned}
a &\equiv r_1 \pmod{m} \\
b &\equiv r_2 \pmod{m} \\
\Rightarrow ab &\equiv r_1 r_2 \pmod{m}
\end{aligned}$$

This is what allows us to complete step 2 in the algorithm.

Furthermore, we know that $a^x a^y = a^{x+y}$ which is what allows us to split up the work in the algorithm and then recombine the results.

1.3.4 Fast-tracking the algorithm

Before applying the Modular Exponentiation Algorithm, we can eliminate a lot of the work by applying Euler's Theorem:

For any modulus n and integer a which are co-prime (ie $\text{GCD}(a,n)=1$) we have that:

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

Hence, to aid in computing the least positive residue we can express the exponent in the original problem as $n = q\varphi(m) + r$. This helps us in that we can now write $a^{q\varphi(m)+r}$ and since we know that $a^{\varphi} \equiv 1 \pmod{m}$ the problem is reduced to finding the least positive residue of $a^r \pmod{m}$. This is because $a^n \equiv a^{q\varphi(m)+r} \equiv a^{q\varphi(m)} a^r \equiv a^r \pmod{m}$.

1.4 Fermat's Factorization

1.4.1 Statement

Fermat's Factorization allows us to factorize an *odd* integer using the difference of two squares. Given an odd integer n , and two factors c and d s.t. $n = cd$ we know that:

$$n = \left(\frac{c+d}{2}\right)^2 - \left(\frac{c-d}{2}\right)^2$$

The method is as follows:

- We start by choosing $a = \lceil \sqrt{n} \rceil$
- We then calculate $n - a^2 = z$
- If z is a perfect square, we have found $b = \sqrt{z}$
- If z is not a perfect square, we increment a , and try again until we get a z which is a perfect square or we ascertain that n is a prime.
- Once we have a and b , the factors of n are calculated as:
$$c = a + b$$
$$d = a - b$$

2 Cryptosystems

2.1 RSA Encryption

2.1.1 How it works

Assume Bob wants to send Alice an encrypted message.

- Alice begins by choosing two large primes, p and q
- Alice then computes $n = pq$
- Alice then chooses e s.t. $e \in (1, \varphi(n))$ and $\text{GCD}(e, \varphi(n)) = 1$
- Alice's **PUBLIC KEY** is (e, n)
- Alice then computes her **PRIVATE KEY** d using $de \equiv 1 \pmod{\varphi(n)}$
(ie d is the inverse of e under modulus $\varphi(n)$)
- Alice then shares her public key with Bob
- Bob encrypts his message P using Alice's public key (e, n) in the formula:
$$C \equiv P^e \pmod{n}$$
- Alice can decrypt the ciphertext C using her private key d in the formula:
$$P \equiv C^d \pmod{n}$$

This works because:

$$\begin{aligned} & C^d \pmod{n} \\ & \equiv (P^e)^d \pmod{n} \\ & \equiv P^{ed} \pmod{n} \\ & \equiv P^1 \pmod{n} \\ & \equiv P \pmod{n} \end{aligned}$$

2.1.2 Signed Messages

Assume Alice wants to send Bob a signed message.

Alice has public key (e_A, n_A) and private key d_A .

Bob has public key (e_B, n_B) and private key d_B .

Alice's signature is the plaintext P .

- Alice calculates $S \equiv P^{d_A} \pmod{n_A}$
- Alice then computes $C \equiv S^{e_B} \pmod{n_B}$ and sends c to Bob
- Bob first performs the usual deciphering $S \equiv C^{d_B} \pmod{n_B}$
- Bob then computes $P \equiv S^{e_A} \pmod{n_A}$

Only Alice can perform the first step since only she knows her private key. The plaintext P will only be recovered correctly if Alice did indeed send the message (since Bob will be using her public key to decrypt).

2.1.3 Evaluation

If someone came into possession of a ciphertext C and the sender's public key (e, n) they would not be able to easily decipher the ciphertext C since they would need to find the inverse of e modulo $\varphi(n)$. This would be an easy task but finding $\varphi(n)$ is very difficult to do when the prime factorization of n is not known. It is an unproved assumption that there is no easier way to decipher C than by first factoring $n = pq$ and then using this to determine $\varphi(n)$ and in turn using this to find the inverse of e modulo $\varphi(n)$.

2.2 Elgamal Encryption

2.2.1 How it works

Assume Bob wants to send Alice an encrypted message.

- Alice chooses a prime p
- Alice calculates g , the primitive root of p
- Alice selects her **PRIVATE KEY** a , s.t. $0 < a < p - 1$
- Alice computes $b \equiv g^a \pmod{p}$
- Alice's **PUBLIC KEY** is (g, b, p)
- Bob's message must be subject to $0 < M < p$
- Bob chooses a random number r s.t. $1 < r < p - 1$
- Bob then calculates the following two numbers using Alice's public key:
$$y_1 \equiv g^r \pmod{p}$$
$$y_2 \equiv Mb^r \pmod{p}$$
- The ciphertext is then $E_x(M) = (y_1, y_2)$
- To decrypt Alice uses her private key in the formula:
$$D_k(y_1, y_2) \equiv y_2(y_1)^{-a} \pmod{p}$$

This works because:

$$\begin{aligned} D_k(y_1, y_2) &\equiv y_2 y_1^{-a} \pmod{p} \\ &\equiv y_2 (g)^{-ar} \pmod{p} \\ &\equiv y_2 (g^a)^{-r} \pmod{p} \\ &\equiv mb^r (b)^{-r} \pmod{p} \\ &\equiv m \pmod{p} \end{aligned}$$

2.2.2 Signed Messages

Assume Alice wants to send Bob a signed message.

Bob's public key is (g, b, p) .

The message is m .

- Alice selects a random number r s.t. $\text{GCD}(r, p-1) = 1$ and computes $y \equiv g^r \pmod{p}$
- Alice then computes $S \equiv (m - ag)r^{-1} \pmod{p-1}$
- The signature is (y, s)