TUGAS 5

KEAMANAN KOMPUTER

NAMA : RHADI INDRAWAN

STB : 192467

KELAS : O

JURUSAN : TEKNIK INFORMATIKA

UNIVERSITAS DIPA MAKASSAR

DESEMBER 2021

1. Perbandingan antara algoritma

- **Elgamal**

Algoritma kriptografi kunci publik ElGamal adalah algoritma blok chipper yang melakukan proses enkripsi pada blok-blok plainteks yang kemudian menghasilkan blok-blok chipertext, yang nantinya blok-blok chipertext tersebut akan didekripsi kembali dan hasilnya kemudian digabungkan menjadi plainteks semula.

Algoritma ini memiliki kelebihan yaitu pembangkitan kunci yang menggunakan logaritma diskrit dan metode enkripsi dekripsi yang menggunakan proses komputasi yang besar sehingga hasil enkripsinya berukuran dua kali dari ukuran semula. Kekurangan algoritma ini adalah membutuhkan resource yang besar karena chipertext yang dihasilkan dua kali panjang plaintext serta membutuhkan processor yang mampu untuk melakukan komputasi yang besar untuk perhitungan logaritma perpangkatan besar.

Untuk proses dekripsi, algoritma ini membutuhkan waktu yang lebih lama karena kompleksitas proses dekripsinya yang rumit. Dibutuhkan dua kali komputasi karena ukuran chiperteks yang lebih besar dibandingkan plainteksnya.

Besar-besaran yang digunakan didalam algoritma ElGamal adalah sebagai berikut :

- Bilangan prima, p (bersifat public atau tidak rahasia)
- Bilangan acak, g (dimana g < p dan bersifat public atau tidak rahasia)
- Bilangan acak, x (dimana x < p dan bersifat private atau rahasia)
- Bilangan acak, k (dimana k < p dan bersifat private atau rahasia)
- m merupakan plainteks dan bersifat private/rahasia
- a dan b merupakan pasangan chiperteks hasil enkripsi bersifat private atau tidak rahasia

Setelah singkat kata sudah mengenal Algoritma ElGamal pada penjelasan di atas, selanjutnya untuk memperkuat pemahaman langsung saja kita terapkan pada contoh kasusnya.

Contoh perhitungan manual proses pembentukan kunci, proses enkripsi, dan dekripsi algoritama ElGamal :

Perhitungan Pembentukan Kunci.

Misalkan A membangkitkan pasangan kunci dengan memilih bilangan :

p = 257

g = 11

x = 13

Kemudian p, g, x digunakan untuk menghitung y :

y = gx mod p

y = 1113 mod 257

y = 22

jadi kunci public A adalah y = 22, g = 11, p = 257 dan kunci private A adalah x =13, p = 257.

- **Knapsack**

Istilah lain yang masih ada sangkut pautnya yaitu knapsnack problem. Knapsnack problem adalah masalah yang mana seseorang berhadapan dengan persoalan optimasi pemilihan benda mana yang bisa ditampung ke dalam suatu wadah berkapasitas terbatas. Adapun optimasi dimaksudkan agar dalam proses pemilihan benda mana yang hendak dimasukkan ke dalam suatu wadah yang dimaksud dihasilkan keuntungan semaksimal mungkin. Masing-masing dari benda yang hendak dimasukkan ini berat dan nilainya difungsikan dalam menentukan prioritasnya pada pemilihan tersebut.

Adapun nilai yang dimaksud bisa berupa harga barang, tingkat kepentingan, nilai sejarah dan lain-lain. Untuk wadah dalam bahasan ini mempunyai nilai konstanta sebagai nilai pembatas terhadap tiap-tiap benda yang hendak dimasukkan ke dalam wadah yang tersedia itu. Dalam hal ini ada sebuah tuntutan untuk menggunakan sebuah metode memasukkan benda yang dimaksud tersebut ke dalam sebuah wadah agar menghasilkan hasil yang optimal tetapi tidak melampaui kemampuan wadahnya.

Setelah mengetahui apa itu knapsnack problem, rasanya kurang lengkap jika tidak mengenal apa saja jenis-jenisnya. Jenis-jenis knapsnack problem bisa diamati dalam beberap variasi di antaranya:

- 0/1 Knapack problem dimana tiap barang cuma tersedia sebanyak 1 unit, ambil atau lepaskan begitu saja.

- Fracksional knapsack problem. Dalam hal ini barang bisa dibawa hanya sebagian. Jenis problem ini bisa masuk akal jika barang yang ada bisa dibagi-bagi seperti tepung, gula dan lain-lain.
- Bounded Knapsack problem. Pada jenis ini, masing-masing barang tersedia tersedia dalam N unit yang mana jumlahnya terbatas.
- Unbounded Knapshack problem. Untuk jenis Knapsack problem yang satu ini masing-masing barang yang tersedia jumlahnya minimal dua unit atau bahkan tak terbatas.

- **Diffie-Hellman**

Diffie-Hellman key exchange adalah metode dimana subyek menukar kunci rahasia melalui media yang tidak aman tanpa mengekspos kunci. Metode ini diperlihatkan oleh Dr. W. Diffie dan Dr. M. E. Hellman pada tahun 1976 pada papernya "New Directions in Cryptography". Metode ini memungkinkan dua pengguna untuk bertukar kunci rahasia melalui media yang tidak aman tanpa kunci tambahan. Metode ini memiliki dua parameter sistem, p dan g. Kedua parameter tersebut publik dan dapat digunakan oleh semua pengguna sistem. Parameter p adalah bilangan prima, dan parameter g (sering disebut generator) adalah integer yang lebih kecil dari p yang memiliki properti berikut ini: Untuk setiap bilangan n antara 1 dan p-1 inklusif, ada pemangkatan k pada g sehingga gk = n mod p Penggunaan Algoritma Diffie-Hellman dalam pertukaran kunci dapat dilakukan secara aman dan efektif dalam pemrosesan jika dibandingkan dengan algoritma RSA yang cenderung lebih lama dalam pemrosesan algoritmanya. Proses pertukaran kunci ini dapat dilakukan lebih dari 2 orang asal memenuhi 2 prinsip yang telah dibahas tadi. Algoritma Diffie-Hellman lebih memfokuskan dalam perubahan nilai kunci dan proses matematis dalam penentuan kunci akhir yang sama.

- **RSA**

RSA adalah algoritma yang sangat maju dalam bidang kriptografi kunci public (kriptografi public key) yang sangat popular dan masih digunakan sampai saat ini. RSA merupakan algoritma yang paling cocok untuk digital signature seperti halnya enkripsi. Algoritma RSA masih digunakan secara luas dalam protocol electronic commerce dan dipercaya dalam pengamanan dengan kunci yang sangat panjang. Algoritma RSA disebut sebagai kunci publik karena kunci enkripsi dapat dibuat public yang berarti semua orang dapat mengetahuinya. Walaupun dibuat public key, keamanan algoritma RSA sangat terjaga. Hal itu dikarenakan kunci yang digunakan untuk enkripsi pada algoritma RSA berbeda dengan kunci

yang digunakan untuk dekripsinya. Keamanan enkripsi dan dekripsi algoritma RSA terletak pada kesulitan untuk memfaktorkan modulus n yang sangat besar. Penamaan algoritma RSA diambil dari nama penemunya, yaitu Rivest, Shamir dan Adleman yang diplubikasikan pada tahun 1977 di MIT yang bertujuan untuk menjawab tantangan dari Algoritma Pertukaran Kunci Diffie Helman.

Algoritma RSA mengikuti skema Block Cipher, yaitu sebelum dilakukan enkripsi, plainteks yang ada dibagi ke dalam blok-blok yang sama panjang dimana plainteks dan cipherteksnya berupa integer antara 1 sampai n dengan n biasanya berukuran 1024 bit dan panjang bloknya berukuran tidak lebih dari log (n) + 1 dengan basis 2. Fungsi enkripsi dan dekripsi algoritma RSA adalah sebagai berikut.

Fungsi Enkripsi: C = Me mod n

Fungsi Dekripsi: M = Cd mod n

Ket:

C = Cipherteks

M = Message (plainteks)

e = Kunci public

d = kunci private

Penggunaan algoritma RSA harus memenuhi kriteria-kriteria sebagai berikut.

- Memungkinkan untuk mencari nilai e, d, dan n dimana Med mod n = M untuk semua M < n.
- Relative mudah untuk menghitung nilai Me mod n dan Cd mod n untuk semua nilai M < n.
- Tidak memungkinkan mencari nilai d jika diberikan nilai n dan e.

Syarat nilai e dan d: gcd(d, e) = 1

2. Contoh penerapan

- **Elgamal**

Perhitungan Enkripsi

Misalkan B ingin mengirim plainteks “ENKRIPSI” kepada A, kemudian setiap karakter plainteks tersebut diubah kedalam bentuk ASCII sehingga menghasilkan tabel sebagai berikut:

Plainteks bentuk ASCII

E	N	K	R	I	P	S	I
69	78	75	82	73	80	83	73

Kemudian nilai ASCII tersebut dimasukkan kedalam blok-blok nilai m secara berurutan, sehingga menjadi :

m1 = 69, m2 = 78, m3 = 75, m4 =82, m5 = 73, m6 = 80, m7 = 83, m8 = 73.

Kemudian B memilih bilangan acak k untuk masing-masing nilai m dimana nilai ki ini bernilai $0 < k < p - 1$. Sehingga diambil nilai acak k untuk masing-masing nilai m sebagai berikut :

mn Nilai ki

m1 = 69 58

m2 = 78 178

m3 = 75 251

m4 = 82 62

m5 = 73 137

m6 = 80 27

m7 = 83 256

m8 = 73 173

Kemudian menghitung tiap-tiap blok :

dengan rumus : a = g^k mod p

Nilai m1 : a1 = gk mod p a1 = 1158 mod 257 a1 = 30	Nilai m5 : a5 = gk mod p a5 = 11137 mod 257 a5 = 190
Nilai m2 : a2 = gk mod p a2 = 11178 mod 257 a2 = 137	Nilai m6 : a6 = gk mod p a6 = 1127 mod 257 a6 = 184
Nilai m3 : a3 = gk mod p a3 = 11251 mod 257 a3 = 73	Nilai m7 : a7 = gk mod p a7 = 11256 mod 257 a7 = 1
Nilai m4 : a4 = gk mod p a4 = 1162 mod 257 a4 = 17	Nilai m8 : a8 = gk mod p a8 = 11173 mod 257 a8 = 235

Kemudian perhitungan bi :

dengan rumus : bi = yk. m mod p

Nilai m1 : b1 = yk m mod p b1 = 2258.69 mod 257 b1 = 201	Nilai m5 : b5 = yk m mod p b5 = 22137.73 mod 257 b5 = 16
Nilai m2 : b2 = yk m mod p b2 = 22178.78 mod 257 b2 = 82	Nilai m6 : b6 = yk m mod p b6 = 2227.80 mod 257 b6 = 203
Nilai m3 : b3 = yk m mod p b3 = 22251.75 mod 257 b3 = 147	Nilai m7 : b7 = yk m mod p b7 = 22256.83 mod 257 b7 = 83
Nilai m4 : b4 = yk m mod p b4 = 2262.82 mod 257 b4 = 220	Nilai m8 : b8= yk m mod p b8 = 22173.73 mod 257 b8 = 249

Setelah mendapatkan nilai a dan b, hasil perhitungan tersebut disusun dengan pola :

a1, b1, a2, b2, a3,b3, a4, b4, a5, b5, a6, b6, a7, b7, a8, b8.

Sehingga membentuk chiperteks :

30 201 137 82 73 147 17 220 190 16 184 203 1 83 235 249

Perhitungan Dekripsi

A mendekripsikan chiperteks dari B dengan melakukan perhitungan dengan rumus sebagai berikut :

dengan rumus : mi = bi.ai p-1-xmod p

Nilai m1 : m1 = bi.ai p-1-xmod p m1 = 201.30257-1-13 mod 257 m1 = 69	Nilai m5 : m5 = bi.ai p-1-xmod p m5 = 16.190257-1-13 mod 257 m5 = 73
Nilai m2 : m2 = bi.ai p-1-xmod p m2 = 82.137257-1-13 mod 257 m2 = 78	Nilai m6 : m6 = bi.ai p-1-xmod p m6 = 203.184257-1-13 mod 257 m6 = 80
Nilai m3 : m3 = bi.ai p-1-xmod p m3 = 147.73257-1-13 mod 257 m3 = 75	Nilai m7 : m7 = bi.ai^p-1-xmod p m7 = 83.1257-1-13 mod 257 m7 = 83
Nilai m4 : m4 = bi.ai p-1-xmod p m4 = 220.17257-1-13 mod 257 m4 = 82	Nilai m8 : m8 = bi.ai^p-1-xmod p m8 = 249.235257-1-13 mod 257 m8 = 73

Setelah mendapatkan nilai mi, masing-masing nilai m hasil dekripsi menjadi kode ASCII diubah kembali menjadi karakter. Dengan hasil sebagai berikut :

69, 78, 75, 82, 73, 80, 83, 73.

Kemudian kode ASCII tersebut diubah menjadi plainteks dengan hasil sebagai berikut :

ASCII Plainteks

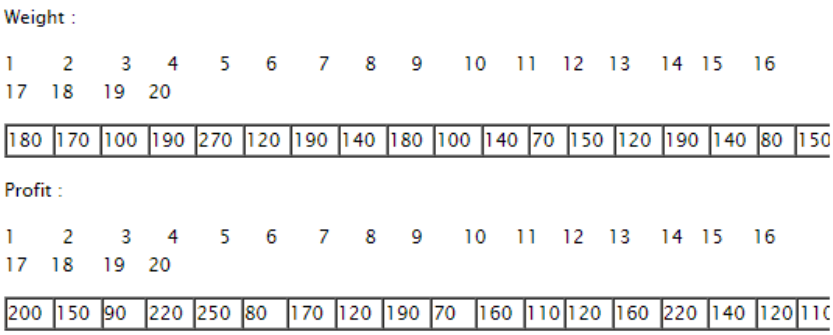
E	N	K	R	I	P	S	I
69	78	75	82	73	80	83	73

Sehingga hasil dekripsi membentuk plainteks “ENKRIPSI”, sama dengan plainteks sebelum di enkripsi.

- **Knapsack**

Representasi Barang

Merepresentasikan barang dalam dua array, dimana array pertama berisi weight (berat) barang, dan array kedua berisi profit (keuntungan) barang.



Constraint

Adapun constraint yang kami gunakan dalam aplikasi ini adalah weight. Jadi,total berat dari sekumpulan barang yang dipilih tidak boleh melebihi kapasitas Knapsack.

Encoding Kromosom

Untuk merepresentasikan kromosom, kami menggunakan array 1 dimensi yang berisi 1 atau 0.

Misal :

Kromosom : 1 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0

Arti : Barang 1, 4, 8, 9, 10, 12, 14, 16, 18 diambil

Barang 2, 3, 5, 6, 7, 11, 13, 15, 17, 19, 20 tidak diambil

Termination Conditions

Pencarian solusi berhenti jika terdapat > 60% kromosom yang mempunyai nilai fitnes maksimum ATAU jumlah evolusi lebih besar limit evolusi yang telah ditentukan (jika jumlah evolusi > 1000).

Fitness Function

Pada evolusi di dunia nyata, individu bernilai fitness tinggi akan bertahan hidup. Sedangkan individu bernilai fitnesss rendah akan mati. Pada AG, suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran niali fitness-nya. Pada aplikasi ini, fitness dihitung dengan menjumlahkan profit tiap barang yang masuk ke dalam knapsack. Jika berat total dalam satu kromosom lebih besar daripada kapasitas maksimum knapsack, maka nilai fitnessnya diassign 0.

Selain dihitung nilai fitnessnya, dihitung pula berat total dari tiap kromosom untuk kemudian dilakukan pengecekan, dimana apabila ada kromosom yang berat totalnya melebihi kapasitas dari knapsack, maka akan dilakukan pencarian gen dalam kromosom tersebut yang bernilai 1 untuk diganti dengan nilai 0. Hal ini dilakukan terus menerus sampai dipastikan bahwa semua kromosom tidak ada yang melanggar constraint.

Untuk mencegah adanya individu yang dominan dalam suatu populasi (dalam pemilihan parent untuk dicrossover), maka diperlukan suatu fungsi Linier Fitness Ranking. Fungsi ini akan menurunkan perbedaan nilai fitness antar individu, sehingga perbedaan antara nilai fitness terbaik dengan nilai fitness terendah dapat diperkecil. Dengan begitu setiap kromosom memiliki kemungkinan untuk terpilih menjadi parent secara lebih merata (lebih adil).

Selection Function

Aplikasi ini menggunakan metode seleksi Roulette Wheel yang dikombinasikan dengan Elitism. Roulette Wheel merupakan suatu metode pemilihan kromosom untuk dijadikan parent, dimana komosom dengan fitness tinggi mempunyai peluang lebih besar untuk dijadikan parent. Sedangkan Elitism adalah suatu metode yang berguna untuk mempertahankan nilai best fitness suatu generasi agar tidak turun di generasi berikutnya. Dalam AG caranya adalah dengan mengcopykan individu terbaik (maxfitness) sebanyak yang dibutuhkan.

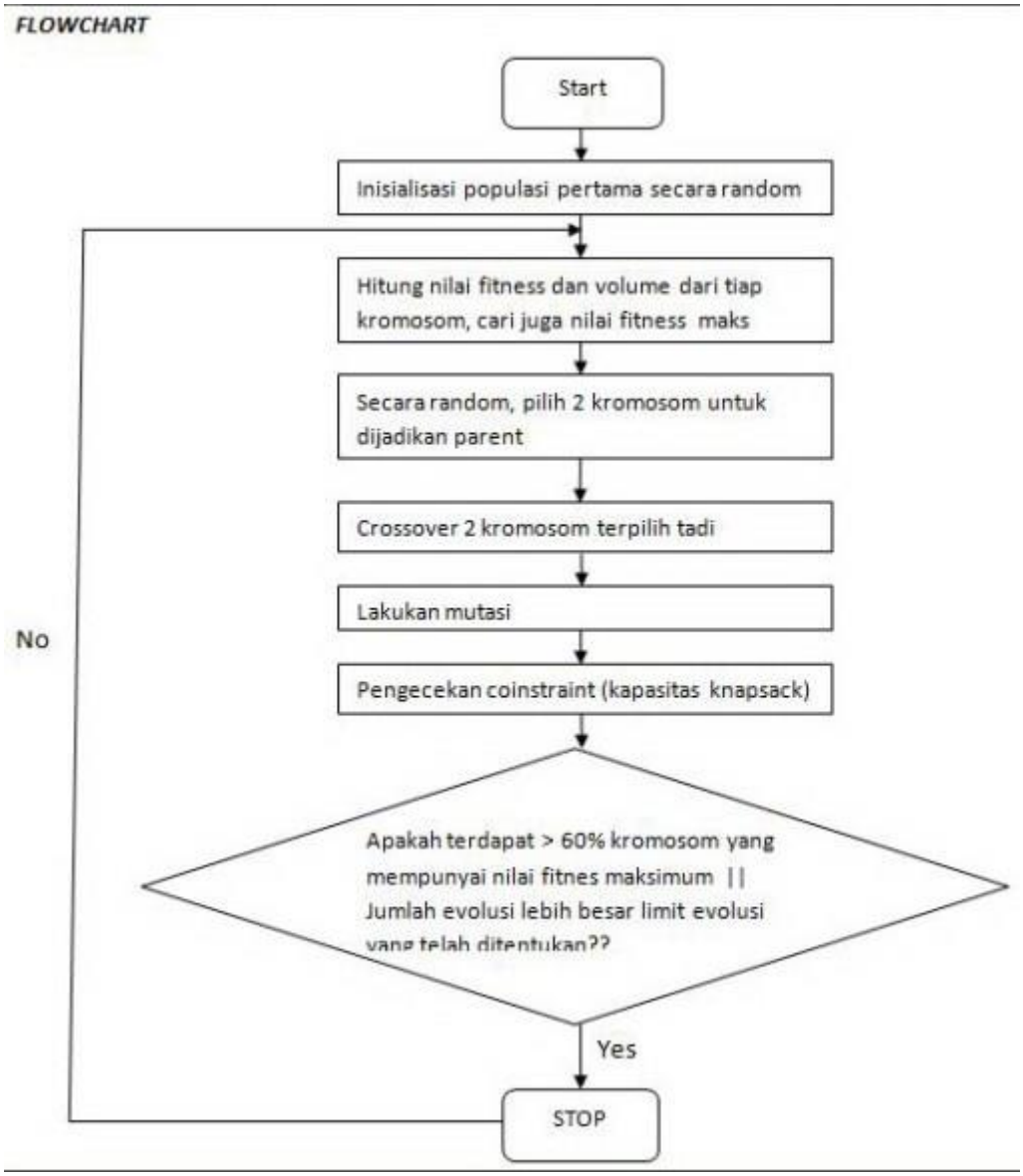
Crossover

Crossover merupakan proses mengkombinasikan bit-bit dalam satu kromosom dengan kromosom lain yang terpilih sebagai parent. Jumlah kromosom yang mengalami crossover ditentukan oleh parameter Pcrossover. Dimana Pcrossover ini kami assign sebesar 80%, karena kami mengharapkan 80% dari populasi mengalami crossover agar populasi individu menjadi lebih variatif.

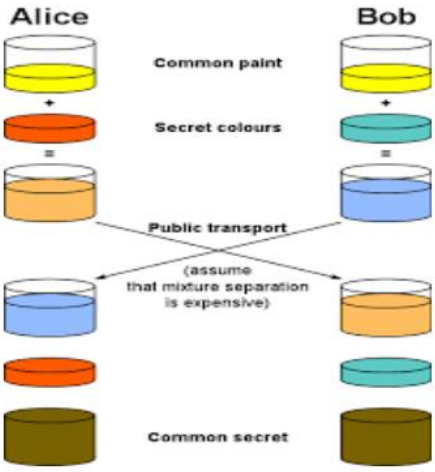
Mutation

Mutation diperlukan untuk mengembalikan informasi bit yang hilang akibat crossover. Mutasi ini dilakukan pada tingkat gen, dan jumlah gen yang dimutasi kami batasi dalam suatu variabel Pmutasi sebesar 5%. Nilai ini kami rasa cukup karena semakin banyak gen yang dimutasi maka kualitas dari suatu individu bisa mengalami penurunan.

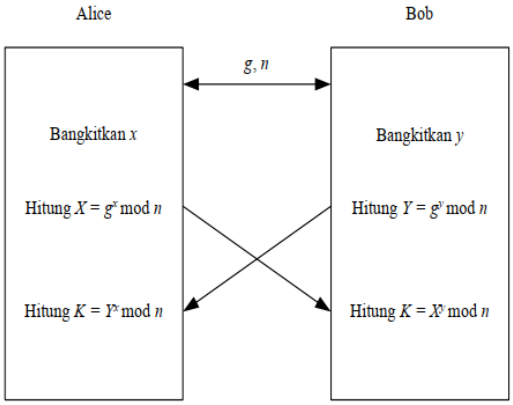
Setelah dilakukan mutasi, kembali dicek untuk tiap kromosomnya apakah melanggar constraint atau tidak. Jika ada kromosom yang total beratnya melebihi kapasitas Knapsack, maka secara random, gen yang bernilai 1 akan diganti dengan 0 sampai kromosom tersebut tidak melanggar constraint. Jadi dapat disimpulkan, aplikasi kami akan selalu menemukan solusi.



- **Diffie Hellman**



Gambar 1.2 pertukaran kunci Diffie- Hellman



Gambar 1.3 Proses Diffie-Helman

Alice dan Bob menyepakati $n = 97$ dan $g = 5$ $(g < n)$

1. Alice memilih $x = 36$ dan menghitung
   $X = g^x \bmod n = 5^{36} \bmod 97 = 50$
   Alice mengirimkan X kepada Bob.
2. Bob memilih y = 58 dan menghitung
   $Y = g^y \bmod n = 5^{58} \bmod 97 = 44$
   Bob mengirimkan Y kepada Alice.
3. Alice menghitung kunci simetri K,
   $K = Y^x \bmod n = 44^{36} \bmod 97 = 75$
4. Bob menghitung kunci simetri K,
   $K = X^y \bmod n = 50^{58} \bmod 97 = 75$

Jadi, Alice dan Bob sekarang sudah mempunyai kunci enkripsi simetri yang sama, yaitu $K = 75$.

- **RSA**

Berikut ini adalah contoh perhitungan manual enkripsi dan dekripsi menggunakkan algoritma RSA.

Dimana sebelum-nya kita harus menentukkan dulu Public Key Dan Private Key nya.

Dan berikut langkah-langkah algoritma RSA mendapatkan Public Key Dan Private Key :

- Pertama, menentukan 2 buah bilangan prima untuk p dan q :

p = 11

q = 13

- Selanjutnya mendapatkan nilai n dimana rumus-nya :

n = p * q

dan akan menjadi seperti ini :

n = 11 * 13

n = 143

- Mendapatkan nilai m dimana rumus-nya : m = (p - 1) * (q - 1)

dan akan menjadi seperti ini :

m = (11 - 1) * (13 - 1)

m = (10) * (12)

m = 120

- Menentukan nilai e dengan syarat :

e = e > 1 and GCD(m,e) = 1

Dimana "17" adalah nilai yang memenuhi syarat untuk nilai e

e = GCD(120,17) = 1

- Menentukan nilai d dengan syarat :

d = (d * e) mod m = 1

Dimana "473" adalah nilai yang memenuhi syarat untuk nilai d

d = (473 * 17) mod 120 = 1

- Dari proses diatas, maka akan mendapatkan kunci public dan kunci privat dimana :

public key = (e,n)

private key = (d,n)

Dan kunci akan menjadi seperti ini :

public key = (17,143)

private key = (473,143)

- Setelah kita mendapatkan public key dan private key, proses selanjutnya

melakukan Enkripsi dan Dekripsi, yaitu kata "INDONESIA". Berikut prosesnya :

Text	ASCII (A)	Proses Enkripsi (X) C = A ^ e mod n	Proses Dekripsi (Y) Y = C ^ d mod n
I	73	= (7 ^ 17) mod 143 = 50 = (3 ^ 17) mod 143 = 9 = 50.9	= (50 ^ 473) mod 143 = 7 = (9 ^ 473) mod 143 = 3 = 73 → I
N	78	= (7 ^ 17) mod 143 = 50 = (8 ^ 17) mod 143 = 112 = 50.112	= (50 ^ 473) mod 143 = 7 = (112 ^ 473) mod 143 = 8 = 78 → N
D	68	= (6 ^ 17) mod 143 = 41 = (8 ^ 17) mod 143 = 112 = 41.112	= (41 ^ 473) mod 143 = 6 = (112 ^ 473) mod 143 = 8 = 68 → D
O	79	= (7 ^ 17) mod 143 = 50 = (9 ^ 17) mod 143 = 81 = 50.81	= (50 ^ 473) mod 143 = 7 = (81 ^ 473) mod 143 = 9 = 79 → O
N	78	= (7 ^ 17) mod 143 = 50 = (8 ^ 17) mod 143 = 112 = 50.112	= (50 ^ 473) mod 143 = 7 = (112 ^ 473) mod 143 = 8 = 78 → N
E	69	= (6 ^ 17) mod 143 = 41 = (9 ^ 17) mod 143 = 81 = 41.81	= (41 ^ 473) mod 143 = 6 = (81 ^ 473) mod 143 = 9 = 69 → E
S	83	= (8 ^ 17) mod 143 = 112 = (3 ^ 17) mod 143 = 9 = 112.9	= (112 ^ 473) mod 143 = 8 = (9 ^ 473) mod 143 = 3 = 83 → S
I	73	= (7 ^ 17) mod 143 = 50 = (3 ^ 17) mod 143 = 9 = 50.9	= (50 ^ 473) mod 143 = 7 = (9 ^ 473) mod 143 = 3 = 73 → I
A	65	= (6 ^ 17) mod 143 = 41 = (5 ^ 17) mod 143 = 135 = 41.135	= (41 ^ 473) mod 143 = 6 = (135 ^ 473) mod 143 = 5 = 65 → A

3. Source Code

Untuk melihat contoh source code pada algoritma diatas dapat mengunjungi link dibawah ini

https://github.com/rhadi16/kunci_publik