Pengaplikasian Teori Bilangan untuk Kriptografi

Christian Jonathan (13516092)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
christianchrisjo@students.itb.ac.id

Abstract*— *makalah ini berisi tentang algoritma RSA sebagai salah satu metode kriptografi yang menerapkan teori bilangan seperti aritmatika modulo, kongruensi dan sebagainya. Makalah ini juga berisi tentang cara kerja,kelemahan,kelebihan dan keuntungan serta kerugian pemakaian algoritma RSA.

Keywords — **Algoritma RSA, aritmatika modulo, kongruensi,** *public-key* kriptografi**, teori bilangan,** *totient function*

I. PENDAHULUAN

Kriptografi adalah teknik untuk mengamankan pesan sehingga informasi yang disampaikan dari suatu pihak ke pihak yang dituju tidak diketahui oleh pihak ketiga yang tidak berwenang untuk mengetahui. Cara kerja kriptografi itu sendiri adalah dengan mengubah suatu teks atau pesan yang memiliki makna (*plaintext*) menjadi sesuatu yang tidak bermakna atau tidak masuk akal (atau dikenal sebagai *ciphertext*), hal ini disebut dengan enkripsi. Setelah pihak yang dituju mendapatkan pesan tersebut, terjadilah penerjemahan ulang hasil enkripsi agar pesan yang "tidak bermakna" tersebut menjadi bermakna kembali, hal ini disebut dekripsi.

Algoritma dan teknik-teknik *cypher*(metode enkripsi dan dekripsi) mulai berkembang. Salah satu metodenya adalah metode *Public-key crypthography*, yang menggunakan kunci yang berbeda untuk proses enkripsi-dekripsi (menggunakan kunci publik dan kunci privat).

Algoritma yang menggunakan metode *public-key* ini salah satunya adalah algoritma RSA yang diambil dari nama pembuatnya yaitu Ron **R**ivest, Adi **S**hamir dan Leonard **A**dleman. Algoritma RSA ini menggunakan "kunci" yang dibentuk oleh dua bilangan prima yang acak. Algoritma RSA itu sendiri merupakan hasil dari penerapan prinsip-prinsip teori bilangan, khususnya prinsip bilangan prima, aritmatika modulo serta teorema Totient Euler.

Berbeda dengan beberapa metode kriptografi, Algoritma RSA menggunakan kriptografi berjenis asimetris yang berarti kunci untuk dekripsi dan kunci untuk enkripsi berbeda, algoritma lain yang menggunakan kriptografi asimetris adalah ECC. Sedangkan kriptografi simetris menggunakan kunci yang untuk dekripsi dan enkripsinya sama (contoh nya : DES, AES, MARS, dsb)

II. DASAR TEORI

2.1 Bilangan Prima dan Relatif Prima

2.1.1 Bilangan Prima

Bilangan prima adalah bilangan yang hanya memiliki faktor pembaginya adalah 1 dan bilangan itu sendiri dengan contoh:

29 = 29 x 1
23 = 23 x 1

Berbeda dengan bilangan komposit, bilangan komposit adalah suatu bilangan yang memiliki 1 atau lebih bilangan prima sebagai faktor pembaginya dengan contoh:

4 = 2 x 2
175 = 5 x 5 x 7

2.1.2 Relatif Prima

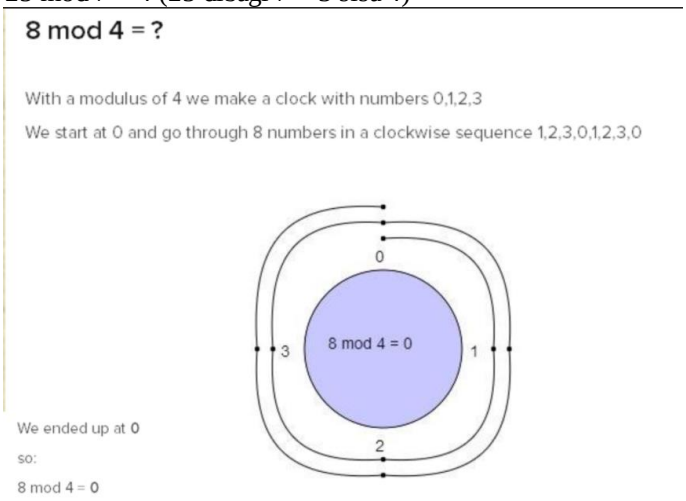
Dua buah bilangan bulat dikatakan relatif prima jika PBB(a,b) = 1 dimana PBB adalah Pembagi Bersama Terbesar. Atau seringkali di notasikan dengan GCD(a,b) dimana GCD adalah *Greatest Common Divisor*. Dengan contoh dua bilangan yang relatif prima :

20 dan 3 relatif prima sebab PBB(20,3) = 1
7 dan 11 relatif prima sebab PBB(7,11) = 1

2.2 Aritmatika Modulo

Operasi modulo akan memberikan sisa dari hasil pembagian suatu bilangan dengan bilangan modulo nya. Misalkan a dan m adalah bilangan bulat dimana (m>0). Operasi a modulo m atau dinotasikan dengan **a mod m** akan memberikan sisa jika a dibagi dengan m. dengan contoh:

10 mod 3 = 1 (10 dibagi 3 = 3 sisa 1)
25 mod 7 = 4 (25 dibagi 7 = 3 sisa 4)



Sumber gambar diambil dari khanacademy.org

Dari ilustrasi tersebut, diketahui bahwa 8 kongruen dengan 0 (mod 4) atau dinotasi $8 \equiv 0 \bmod(4)$. Aritmatika modulo sangat membantu dalam menyelesaikan berbagai masalah di teori

bilangan diantaranya adalah :

Misal a ≡ b (mod m) dan c adalah bilangan bulat

1. a + c ≡ b + c (mod m)
2. a – c ≡ b – c (mod m)
3. ac ≡ bc (mod m)
4. $a^e \equiv b^e \ mod(m)$ dengan e adalah bilangan bulat positif
5. $\frac{a}{f} \equiv \frac{b}{f} \ (mod \ \frac{m}{PBB(m,f)})$

Misal a ≡ b (mod m) dan c ≡ d (mod m) maka :

1. a + c ≡ b + d (mod m)
2. a – c ≡ b – d (mod m)
3. ac ≡ bd (mod m)

2.3 Teorema Totient Euler

2.3.1 Fungsi Totient / Fungsi phi

Fungsi totient berfungsi untuk menghitung banyaknya bilangan bulat yang 1 ≤ x ≤ n yang memenuhi syarat x dan n adalah relatif prima. Dengan contoh penggunaan fungsi totient :

$\varphi(9) = 6$ karena dari sembilan bilangan 1-9 terdapat 6 bilangan yang relatif prima dengan 9 yaitu 1,2,4,5,7 dan 8.

Fungsi totient memiliki kasus unik yaitu:

1. $\varphi(p)$ = p - 1 dengan p adalah bilangan prima. contoh: $\varphi(13) = 12$
2. $\varphi(b^e) = b^e - b^{e-1}$ dengan b adalah bilangan prima. E.g: $\varphi(11^2) = 110$
3. $\varphi(p.n) = \varphi(p) * \varphi(n)$ dengan p dan n adalah bilangan bulat. E.g: $\varphi(36) = \varphi(9) * \varphi(4) = 6 * 2 = 12$
4. $\varphi(p)$ selalu bernilai genap untuk p > 2.

2.3.2 Teorema Euler.

Teorema Euler merupakan hasil generalisasi dari Fermat's Little Theorem (FLT) karena FLT hanya dapat bekerja dengan baik ketika bilangannya adalah bilangan prima. Teorema FLT berisi:

$a^{p-1} \equiv 1 \ (mod \ p)$

Sedangkan Teorema Euler yang merupakan hasil generalisasi FLT berisi :

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

Untuk m positif bilangan bulat dan a adalah bilangan bulat dimana PBB(a,m) = 1.

Contoh aplikasi pengunaan Teorema Euler :

1. Tentukan digit terakhir dari 3^{1000}
   solusi:
   $3^{\varphi(10)} = 3^4 \equiv 1 \ (mod \ 10)$
   $3^{1000} = 3^{4*250} \equiv 1^{250} \ (mod \ 10) \equiv 1 \ (mod \ 10)$
   jadi digit terakhir dari $3^{1000} = 1$
2. Tentukan hasil dari 3^{100000} dibagi 35
   solusi :
   $3^{\varphi(35)} \equiv 1 \ (mod \ 35)$

$3^{24} \equiv 1 \ (mod \ 35)$ dan karena $3^{100000} = 3^{24*4166} * 3^{16}$ maka hasil sisanya setara dengan $3^{16} \ mod \ 35$ selanjutnya diselesaikan dengan cara biasa

$3^{16} \ mod \ 35 \equiv 81^4 \ mod \ 35 \equiv 11^4 \ mod \ 35 \equiv 121^2 \ mod \ 35 \equiv 16^2 \ mod \ 35 \equiv 256 \ mod \ 35 \equiv 11 \ mod \ 35$

Jadi sisanya adalah 11.

III. PRINSIP KERJA DAN PENJELASAN SINGKAT

3.1 Penerapan Teori Bilangan

Keamanan algoritma RSA didasarkan dari seberapa sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran itu dilakukan untuk memperoleh kunci pribadi. Berikut variabel-variabel yang digunakan algoritma RSA terbagi dari rahasia dan tidak:

1. P dan q bilangan prima (rahasia)
2. $\varphi(r)$ = r $(1 - \frac{1}{p1})(1 - \frac{1}{p2})$ (rahasia)
3. SK sebagai kunci dekripsi (rahasia)
4. X / *plaintext* berisi pesan asli (rahasia)
5. r = p * q (tidak rahasia)
6. Y / *cipertext* berisi hasil enkripsi (tidak rahasia)
7. PK sebagai kunci enkripsi (tidak rahasia)

Algoritma RSA sangat didasarkan pada teorema Euler yang berisi $a^{\varphi(m)} \equiv 1 \ (mod \ m)$ dengan ketentuan:

1. A dan m harus relatif prima
2. $\varphi(r)$ = r$(1 - \frac{1}{p1})(1 - \frac{1}{p2}) \ldots (1 - \frac{1}{pn})$ dengan p1,p2,…,pn adalah faktor prima dari r

   Dengan menggunakan sifat kongruensi modulo yang sudah dijabarkan pada dasar teori, rumus teorema Euler dapat diturunkan menjadi $a^{x*\varphi(m)+1} \equiv a \ (mod \ m)$ dengan a relatif prima terhadap m.

Misalkan SK dan PK dipilih sedemikian sehingga nilai SK*PK ≡ 1 $(mod \ \varphi(m))$ atau x * $\varphi(m)$ + 1. Sehingga dengan mensubstitusikan nilai SK*PK kedalam hasil turunan rumus Euler didapat persamaan $a^{SK.PK} \equiv a \ (mod \ m)$ yang berarti X dipangkat PK lalu di pangkat dengan SK akan menghasilkan nilai X lagi. Sehingga dapat ditarik kesimpulan bahwa enkripsi dan dekripsi dirumuskan :

Enkripsi(X) = Y ≡ X^{PK} (mod r)

Dekripsi(X) = Y ≡ X^{SK} (mod r)

Namun agar enkripsi dan dekripsi akan selalu tetap benar seperti persamaan diatas, nilai X harus dibatasi dengan nilai {0,1,2,…,r-1).

3.2 Algoritma RSA

3.2.1 Penetapan nilai pasangan kunci PK & SK

Cara untuk menetapkan pasangan kunci sbb:

1. Pilih dua bilangan prima secara sembarang dan dirahasiakan anggap p dan q.
2. Hitung r= p.q (besaran ini tidak perlu dirahasiakan)
3. Hitung nilai $\varphi(r)$ = r $(1 - \frac{1}{p1})(1 - \frac{1}{p2})$ setelah nilai ini dihitung, p dan q dapat dihapus untuk diketahui pihak lain
4. Pilih suatu bilangan bulat yang relatif prima dengan $\varphi(r)$ untuk dijadikan public key (PK)
5. Nilai SK dapat dihitung dengan menggunakan kekongruenan SK*PK ≡ 1 (mod($\varphi(r)$) atau ekivalen dengan SK*PK = 1 + m* $\varphi(r)$ sehingga SK = $\frac{1+m*\varphi(r)}{PK}$, akan ada nilai m yang akan mengakibatkan nilai SK adalah bilangan bulat.

3.2.2 Enkripsi

Berikut cara kerja enkripsi pada RSA :

1. Pesan pada *plaintext* dipecah menjadi blok-blok *plaintext*

(x1,x2,x3,….) dan harus memenuhi persyaratan nilai blok xi ada pada himpunan {0,1,..,r-1} agar perhitungan tidak berada di luar himpunan.

2. Nilai blok *cipertext* didapat dengan menggunakan rumus yi = xi^{PK} mod r

3.2.3 Dekripsi

Nilai blok *cipertext* hasil enkripsi akan didekripsi ulang dengan menggunakan rumus dekripsi

Xi = yi^{SK} (mod r)

IV. PENERAPAN DAN PERFORMA

4.1 Contoh penggunaan RSA algorithm:

Misalkan *plaintext* berisi X = HARI INI atau dalam ASCII bernilai 7265827332737873

Langkah pertama untuk menyelesaikan ini, perlu dibuatnya pasangan kunci PK dan SK.

Pertama memilih dua buah bilangan prima secara random, anggap p = 47 dan q = 71 selanjutnya menghitung nilai r = p. q = 3337 dan $\varphi(r)$ = r (1 - $\frac{1}{p1}$) (1 - $\frac{1}{p2}$) = 3220

Pilih kunci PK misal PK = 79 karena 79 relatif prima dengan 3220 dan SK dengan rumus SK = $\frac{1+m*3220}{79}$ dengan mencoba nilai m = 1,2,3,…,r-1 didapat nilai SK yang bulat adalah 1019.

Plaintext dibagi menjadi beberapa blok dengan perblok 3 digit misal, x1 = 726 x2 = 582 x3 = 733 x4 =273 x5 = 787 x6 = 003, diperhatikan bahwa nilai blok blok nya masih dalam rentang 3337 – 1 sesuai dengan syarat. Lalu masing-masing blok di enkripsi dengan rumus yi = xi^{PK} mod r. didapat y1 = 215 y2 = 776 y3 = 1743 y4 = 933 y5 = 1731 y6 = 158. Sehingga hasil dari *cipertext* berisi Y = 215776174393317311158. Dekripsi dilakukan dengan menggunakan kunci SK menggunakan rumus Xi = yi^{SK} (mod r) maka akan didapat x1 = 726 x2 = 582 x3 = 733 x4 = 273 x5 = 787 x6 = 003 (kembali ke bentuk asal) dan diperoleh X = 7265827332737873 atau jika dikonversi dalam karakter ASCII adalah X = HARI INI.

4.2 Tingkat Keamanan Algoritma RSA

Tingkat keamanan dari algoritma RSA sangat terletak pada tingkat kesulitan dalam memfaktorkan suatu bilangan menjadi faktor prima terutama ketika bilangan itu sangat besar nilainya, yang dalam hal ini memfaktorkan nilai r menjadi p dan q nya, yang jika diketahui p dan q nya maka bisa dihitung $\varphi(r)$ = r (1 - $\frac{1}{p1}$) (1 - $\frac{1}{p2}$). Lalu ketika nilai $\varphi(r)$ diketahui dan PK diketahui (karena public key tidak dirahasiakan) maka kunci dekripsi (SK) bisa dihitung dengan persamaan SK*PK = 1 + m* $\varphi(r)$. Ketika semua elemen diatas diketahui maka pihak ketiga yang tidak berwenang untuk mengetahui isi dari pesan akan bisa melakukan dekripsi *cipertext* dan mendapatkan *plaintext* nya.

Kunci RSA pada umumnya sepanjang 1024-2048 bit. Beberapa pakar meyakini kunci 1024 bit ada kemungkinan dipecahkan pada waktu dekat(walaupun masih dalam perdebatan), namun tidak ada seorangpun pakar berpendapat kunci 2048-bit akan pecah pada masa yang terprediksi. Jika r sepanjang 256-bit, maka kunci RSA akan ditemukan dalam waktu beberapa jam. Jika kunci 512-bit maka dalam waktu ratusan jam menggunakan ratusan komputer akan dipecahkan seperti pada tahun 1999.

Sehingga penemu Algoritma RSA menyarankan agar nilai p dan q memiliki panjang yang lebih dari 100 digit, dengan demikian nilai dari r akan berukuran lebih dari 200 digit (2048-bit), hal ini bertujuan agar menyulitkan pihak ketiga untuk melakukan pemfaktoran bilangan bulat menjadi faktor primanya. Menurut salah satu pencipta algorita RSA (Rivest) untuk mencari faktor bilangan dari suatu bilangan yang panjangnya 200 digit dibutuhkan waktu komputasi selama 4 milyar tahun dengan asumsi algoritma pemfaktoran yang digunakan adalah algoritma tercepat saat ini dan komputer mempunyai kecepatan 1 milidetik

Kelebihan lain dari algoritma RSA adalah daya tahannya terhadap berbagai serangan terutama serangan *brute force*. Hal ini dikarenakan kompleksitas dekripsinya (ditentukan secara dinamis dengan cara menentukan nilai p dan q yang besar pada saat proses pembuatan pasangan kunci) sehingga tahan terhadap serangan, dan sampai saat ini belum ada teknik pembobolan lain yang lebih efektif daripada *brute force*.

4.3 Kelemahan dari Algoritma RSA

Ketika di masa depan tercipta suatu algoritma pemfaktoran yang jauh lebih cepat dari saat ini, maka keamanan dari RSA akan sangat terancam. Untungnya algoritma yang efisien tersebut belum ditemukan sehingga sampai saat ini algoritma RSA masih sangat direkomendasikan untuk digunakan untuk melakukan enkripsi dan dekripsi (karena masih sulit untuk di pecahkan). Kelemahan lain dari algoritma RSA terletak pada jumlah waktu yang diperlukan untuk memproses akibat besarnya ukuran kunci privat, terutama untuk pesan yang besar. Oleh karena itu RSAbiasanya digunakan hanya untuk meng-enkripsi pesan berukuran kecil seperti kata kunci dari enkripsi simetris seperti algoritma DES dan AES.

Kelemahan lain adalah dengan adanya celah penyerangan dengan menggunakan CRT (Chinese Remainder Theorem) akibat pengiriman pesan yang sama ke penerima yang berbeda yang tentunya dengan SK yang berbeda. Dengan ilustrasi sebagai berikut:

$X^m \equiv E_1 \ (mod\ r1)$

$X^m \equiv E_2 \ (mod\ r2)$

$X^m \equiv E_3 \ (mod\ r3)$

Penyerang tidak bisa mencari invers dari salah satu diatas, namun karena pesannya sama untuk semua penerima(walaupun beda *cipertext*), penyerang dapat mengkonversi operasi-operasi ini dimana operasi invers nya mudah dengan menggunakan CRT untuk menggabungkan 3 *cipertext* untuk mendapatkan:

$X^m \equiv E_1 \ (mod\ (r1 * r2 * r3))$

Untuk mengatasi serangan semacam ini, maka digunakannya teknik padding atau menambahkan sesuatu pada *plaintext* sehingga setiap pesan memiliki *plaintext* yang berbeda, hal ini sangat penting karena akan sangat berbahaya untuk mengirim *plaintext* yang sama beberapa kali walaupun ke penerima yang sama. Sehingga pada tahun 1998, algoritma RSA menggunakan PCKS # 1 v1 padding scheme, sayangnya pada saat itu padding scheme nya memiliki kecacatan sehingga terdapat banyak celah penyerangan hingga akhirnya algoritma RSA beralih ke padding

scheme bernama Optimal Asymmetric Encryption Padding dan RSA telah merilis versi terbaru dari PCKS # 1 yang tidak lemah terhadap serangan.

4.4 Keuntungan dan Kerugian RSA

Keuntungan utama dari RSA adalah menambah keamanan dan kenyamanan.karena kunci privat tidak perlu dikirim atau diberi ke pihak lain. Lain dengan sistem kunci rahasia lain, kunci ini harus dikirim secara manual atau lewat kamunikasi dan terjadi kemungkinan penyerang dapat mencari tahu kunci rahasia tersebut saat proses pengiriman ataupun melakukan penyadapan komunikasi.

Kelemahan dari RSA yang paling terasa adalah waktu pemrosesan nya yang lambat ditimbang dengan algoritma-algoritma lain yang sifatnya simetrikal seperti DES dan AES. Namun dengan menggabungkan kedua algoritma dapat dinikmati metode enkripsi yang terbaik di dunia. Yaitu melakukan enkripsi *plaintext* dengan algoritma lain dan kunci nya di enkripsi oleh RSA, sehingga kedua keuntungan dari masing-masing metode yang digunakan akan bisa dinikmati oleh user.

4.5 Penggunaan Algoritma RSA di Kehidupan

Sampai saat ini, penggunaan RSA digunakan di berbagai macam produk, industri dan standar di seluruh dunia. RSA dipakai di berbagai perangkat lunak komersil. RSA dibangun ke dalam sistem operasi yang dipakai seperti Microsoft, Apple, Sun dan Novell. Tidak hanya *software* RSA juga diimplementasikan pada *hardware*, RSA dapat digunakan pada *secure telephone*, kartu jaringan Etherneet dan kartu cerdas. RSA juga digabungkan ke dalam internet protokol berskala besar untuk digunakan pada komunikasi internet yang aman, seperti SSL, S-HTTP, SEPP, S/MIME, S/WAN, STT dan PCT dan yang terakhir RSA digunakan di beberapa institusi termasuk cabang-cabang dari pemerintah, perusahaan-perusahaan besar, laboratorium dan universitas.

V. KESIMPULAN

Teori bilangan sangat bermanfaat untuk kehidupan sehari-hari, bahkan teori bilangan dapat diimplementasikan untuk membuat suatu metode kriptografi yang aman dan sangat berguna untuk kehidupan kita yang semakin kedepan semakin erat dengan teknologi dan informasi.

Algoritma RSA adalah salah satu bukti mutlak bahwa teori bilangan itu sangat diperlukan oleh kita, baik untuk membuat algoritma maupun untuk meminimalisir kemungkinan-kemungkinan serangan dan mengurangi tingkat keberhasilan dari serangan pada informasi yang kita miliki. Saya juga meyakini bahwa kedepannya ilmu teori bilangan ini akan terus dikembangkan dan diimplementasikan untuk kemajuan teknologi.

VI. APPENDIX

Berikut adalah Algoritma RSA dalam bahasa C++
Diambil dari : http://www.metode-algoritma.com/2013/06/algoritma-rsa.html

```
#include <iostream>
#include <math.h>
#include <string.h>
#include <stdlib.h>

using namespace std;

long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i ;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
int prime(long int pr)
{
        int i;
        j=sqrt(pr);
        for(i=2;i<=j;i++)
        {
                if(pr % i == 0) return 0;
        }
        return 1;
}
int main()
{
        cout << "\nENTER FIRST PRIME NUMBER\n";
        cin >> p;
        flag = prime(p);
        if(flag==0)
        {
                cout<<"\nWRONG INPUT\n";
                exit(1);
        }
        cout << "\nENTER ANOTHER PRIME NUMBER\n";
        cin >> q;
        flag = prime(q);
        if(flag == 0 || p==q)
        {
                cout << "\nWRONG INPUT\n";
                exit(1);
        }
        cout << "\ENTER MESSAGE\n";
        fflush(stdin);
        cin >> msg;
        for(i=0;msg[i] != NULL;i++)
                m[i] = msg[i];
        n= p*q;
        t = (p-1) * (q-1);
        ce();
        cout << "\nPOSSIBLE VALUES OF e AND d ARE\n";
        for (i=0; i< j;i++)
                Cout << e[i] << "\t" << d[i] << "\n";
        encrypt();
        decrypt();
        return 0;
```

```
}

void ce()
{
	int k;
	k=0;
	for(i=2;i<t;i++)
	{
		if(t % i == 0)
			continue;
		flag = prime(i);
		if (flag == 1 && i != p && i != q)
		{
			e[k] = i;
			flag = cd(e[k]);
			if(flag > 0)
			{
				d[k] = flag;
				k++;
			}
			if(k ==99)
				break;
		}
	}
}
long int cd(long int x)
{
	long int k = 1;
	while (1)
	{
		k = k + t;
		if (k % x == 0)
			return (k/x);
	}
}
void encrypt()
{
	long int pt,ct,key = e[0],k,len;
	i=0;
	len = strlen(msg);
	while(i != len)
	{
		pt = m[i];
		pt = pt - 96;
		k = 1;
		for (j=0; j<key;j++)
		{
			k = k * pt;
			k = k % n;
		}
		temp[i] = k;
		ct = k + 96;
		en[i] = ct;
		i++;
	}
	en[i] = -1;
	cout << "\nTHE ENCRYPTED MESSAGE IS\n";
	for(i = 0;en[i] != -1;i++)
		print("%c", en[i]);
}
void decrypt()
{
	long int pt,ct,key = d[0], k;
	i=0;
	while(en[i] != -1)
	{
		ct = temp[i];
		k = 1;
		for(j = 0; j < key; j++)
		{
			k = k * ct;
			k = k % n;
		}
		pt = k + 96;
		m[i] = pt;
		i++;
	}
	m[i] = -1;
	cout << "\nTHE DECRYPTED MESSAGE IS\n";
	for(i=0;m[i] != -1; i++)
		printf("%c", m[i]);
}
```

Dan berikut adalah contoh dari keluaran hasil algoritma diatas, diambil dari:
http://www.metode-algoritma.com/2013/06/algoritma-rsa.html

```
g++ -o RSA RSA.cpp -lm
RSA

ENTER FIRST PRIME NUMBER
47

ENTER ANOTHER PRIME NUMBER
53

ENTER MESSAGE
Dharmendra

POSSIBLE VALUES OF e AND d ARE
3 1595
5 957
7 1367
11 435
17 985
19 1259
29 165
31 463
37 1293
41 2217
43 1947
59 1419
61 549
67 2035
71 1415
73 1409
```

```
79 1847
83 2075
89 2177
97 1233
101 1421
103 2183

THE ENCRYPTED MESSAGE IS
x`a???]??a
THE DECRYPTED MESSAGE IS
Dharmendra
------------------
(program exited with code: 0)
Press return to continue
```

Sebagai catatan, algoritma ini hanya merupakan algoritma sederhana RSA untuk memperlihatkan kira-kira bagaimana cara kerja RSA tanpa adanya skema padding untuk meminimalisasi kemungkinan serangan dari pihak lain. Dengan penjelasan seperti yang ada pada contoh penerapan RSA pada Bab IV.

VII. UCAPAN TERIMA KASIH

Saya sangat berterima kasih kepada Tuhan Yang Maha Esa, kepada orang tua saya atas kesempatan membuat makalah ini dan atas suksesnya pengerjaan makalah ini tanpa adanya halangan sehingga saya dapat menyelesaikan makalah ini dengan cukup baik. Saya juga mengucapkan terima kasih kepada dosen pengajar mata kuliah matematika diskrit IF2120 atas pemberian materi teori bilangan sehingga saya bisa menemukan inspirasi pengerjaan makalah ini dan bisa belajar banyak hal dari tugas makalah ini. Yang terakhir tidak lupa saya mengucapkan terima kasih kepada teman-teman yang turut mendukung saya agar terus termotivasi dalam pembuatan makalah ini.

REFERENCES

[1] http://www.inmyidea.id/2012/09/eulers-totient-function.html , diases pada 1 Desember 2017
[2] http://hendrydext.blogspot.co.id/2009/09/teorema-euler.html , diakses pada 1 Desember 2017
[3] http://mathworld.wolfram.com/TotientFunction.html , diakses pada 2 Desember 2017
[4] http://informatika.stei.itb.ac.id/~rinaldi.munir , Slide Teori Bilangan diakses pada 1 Desember 2017
[5] https://rdist.root.org/2009/10/06/why-rsa-encryption-padding-is-critical/ , diakses pada 3 Desember 2017
[6] https://www.academia.edu/7587985/STUDI_PEMAKAIAN_ALGORITMA_RSA_DALAM_PROSES_ENKRIPSI_dan_APLIKASINYA , diakses pada 3 Desember 2017
[7] http://www.metode-algoritma.com/2013/06/algoritma-rsa.html , diakses pada 3 Desember 2017.
[8] http://www.geeksforgeeks.org/rsa-algorithm-cryptography/ , diakses pada 3 Desember 2017.
[9] https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/euler-s-totient-function-phi-function , diakses pada 2 Desember 2017
[10] https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/what-is-modular-arithmetic , diakses pada 2 Desember 2017
[11] https://www.khanacademy.org/computing/computer-science/cryptography , diakses pada 2 Desember 2017
[12] https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/congruence-modulo , diakses pada 2 Desember 2017

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Christian Jonathan (13516092)