Studi Sistem Kriptografi Kunci Publik NTRU

Ronald Augustinus Penalosa – NIM : 13503117

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13117@students.if.itb.ac.id

Abstrak

Makalah ini membahasa mengenai sebuah sistem kiptografi kunci publik yang diberikan nama NTRU. Sistem kriptografi kunci publik adalah sebuah sistem yang menggunakan sepasang kunci dalam sistemnya. Sistem yang menggunaan kunci publik marak digunakan dalam dunia kriptografi karena selain memiliki keamanan yang tangguh namun juga memiliki tingkat kepraktisan yang tinggi, karena kunci publik tidak perlu dirahasiakan. Hanya kunci privat saja yang perlu dirahasiakan untuk menjaga kerahasiaan pesan.

Algoritma NTRU adalah sebuah algoritma yang menggunakan 2 buah tipe kunci sebagaimana algoritma lain yang ada dalam keluarga algoritma kunci publik. NTRU pertama kali dikenalkan pada tahun 1996. Algoritma ini dibuat oleh 3 orang yaitu Jeffrey Hoffstein, Jill Pipher dan Joseph Silverman. Kekuatan algoritma ini adalah pada letak sulitnya menemukan vektor yang singkat dari sebuah *lattice*. Sehingga dalam prosesnya NTRU menggunakan operasi terhadap polinom. Sehingga dalam prakteknya pesan perlu diubah dahulu kedalam bentuk polinom untuk dapat dilakukan proses melalui sistem ini.

Selain pembahasan mengenai algoritma NTRU dilakukan pula studi perbandingan kecepatan algoritma ini dengan algoritma kunci publik lain yang sejenis. Pembahasan mengenai Application Programming Interface (API) dan aplikasi buatan NTRU CryptoSystem yang menggunakan algoritma ini pun turut dibahas. Pembahasan mengenai kemungkinan serangan yang terjadi pun turut dibahas, sehingga dapat diperoleh bagaimana gambaran singkat dari kekuatan dari algoritma ini.

***Kata kunci**: Sistem Kriptografi Kunci Publik,Lattice,Polinom,Application Programing Interface*

1. Pendahuluan

Dalam dunia kriptografi penyembunyian pesan dapat dilakukan dngan barbagai cara. Cara yang paling umun diterapkan ialah dengan menyamarkan pesan yang ingin dikirimkan sehingga menjadi pesan yang berbeda. Kemudian ketika pesan tersebut sampai ke tangan orang yang berwenang pesan yang sudah tersamarkan tersebut akan dapat dilihat kembali dengan melakukan sebuah proses pembalikan pesan tersamar tersebut.

Proses penyamaran sebuah pesan dan pengembalian pesan tersebut disebut sebagai sebuah sistem kriptografi. Dalam sebuah sistem kriptografi biasanya melibatkan adanya kunci. Kunci tersebut digunakan untuk proses penyamaran atau yang sering disebut dengan enkripsi dan juga proses pengembalian pesan tersamar ke dalam bentuk semula atau yang sering disebut dekripsi dalam dunia kriptografi.

Sebelum tahun akhir tahun 1970, hanya ada sistem kriptografi simetri. Karena sistem kriptografi simetri menggunakan kunci yang sama untuk enkripsi dan dekripsi, maka hal ini mengimplikasikan dua pihak yang berkomunikasi saling mempercayai. Kedua pihak harus menjaga kerahasiaan kunci (sehingga, junci enkripsi/dekripsi disebut juga *secret key*)

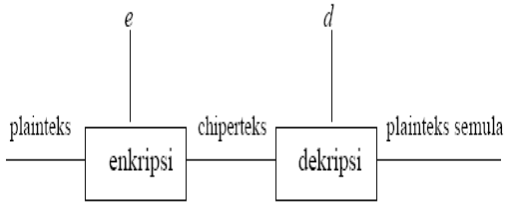
Tentunya sistem ini akan mempersulit pengiriman pesan, karena apabila pesan ingin dikirimkan, kunci dari pesan tersebut pun harus

ikut dikirimkan. Sehingga hal ini akan mempersulit pengiriman pesan antara dua belah pihak.

Melihat akan kesulitan yang ditimbulkan dari penggunaan sistem kriptografi kunci publik maka ditemukannlah sistem kriptografi kunci publik yang menggunaan kunci yang berbeda untuk enkripsi dan dekripsi. Sehingga dalam komunikasi kunci untuk enkripsi dapat diketahui oleh semua orang sedangkan kunci untuk dekripsi hanya diketahui oleh orang-orang yang berwenang. Sehingga hal ini menambah kemudahan dan keamanan dalam komunikasi/ pengiriman pesan yang bersifat rahasia

2. Sistem Kriptografi Kunci Publik

Sistem Kriptograsi Kunci Punlik adalah sebuah sistem kriptografi yang menggunakan sepasang kunci dalam sistemnya. Satu kunci digunakan untuk proses enkripsi dan satu lagi dugunakan untuk proses dekripsi. Hal ini dijelaskan dalam gambar dibawah ini :



Gambar 1 Sistem Kriptografi kunci publik

Sebagaimana digambarkan dalam gambar diatas maka sepasan kunci dalam sistem kriptografi kunci publik dapat dijelaskan sebagai berikut :

1. Kunci untuk enkripsi diumumkan kepada publik – oleh karena itu tidak rahasia – sehingga dinamakan **kunci publik** (*public-key*), disimbolkan dengan *e*.

2. Kunci untuk dekripsi bersifat rahasia – sehingga dinamakan**kunci privat** (*private key*), disimbolkan dengan *d*.

Karena ada kunci enkripsi $\neq$ kunci dekripsi, maka sistem kriptografi kunci-publik kadang-kadang disebut juga sistem **kriptografi asimetri**.

Sistem kriptografi kunci-publik didasarkan pada fakta:

1. Komputasi untuk enkripsi/dekripsi pesan mudah dilakukan.

2. Secara komputasi hampir tidak mungkin (*infeasible)* menurunkan kunci privat, *d*, bila diketahui kunci publik,*e*, pasangannya.

Kedua fakta di atas analog dengan:

- Perkalian *vs* pemfaktoran
Mengalikan dua buah bilangan prima, $a \times b = n$, mudah, tetapi memfaktorkan n menjadi faktor-faktor primanya sulit.
Contoh:
$31 \times 47 = 1457$ (perkalian)
$1457 = ? \times ?$ (pemfaktoran)

- Perpangkatan *vs* logaritmik
Melakukan perpangkatan, $y = ax$, mudah, tetapi menghitung $x = a \log y$ sulit jika a tidak diketahui.

Contoh:
$125 = 248832$ (perpangkatan)
$x = a \log 248832 = ?$ (logaritmik)

Berdasarkan konsep-konsep tersebut maka secara general dapat dituliskan untuk proses enkripsi dan dekripti dengan E adalah fungsi enkripsi dan D adalah fungsi enkripsi maka untuk setiap pasangan kunci (e,d) dapat dituliskan rumus sebagai berikut :

$$Ed(m) = c$$

$$Dd(c) - m$$

Untuk fungsi enkripsi

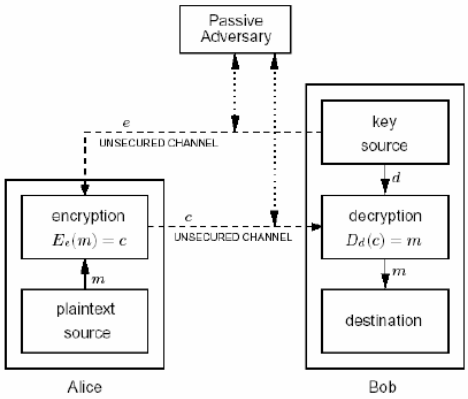
Dimana m adalah plaintext dan c adalah chipherteks.Kedua persamaan ini menyiratkan bahwa dengan mengetahui *e* dan *c*, maka secara komputasi hampir tidak mungkin menemukan *m*. Asumsi lainnya, dengan mengetahui *e*, secara komputasi hampir tidak mungkin menurunkan *d*.

E_e digambarkan sebagai fungsi pintu-kolong (*trapdoor*) satuarah dengan *d* adalah informasi *trapdoor* yang diperlukan untuk menghitung fungsi inversinya, *D*, yang dalam hal ini membuat proses dekripsi dapat dilakukan.

Konsep di atas menjadi penting bila kriptografi kunci-publik digunakan untuk mengamankan pertukaran pesan dari dua entitas yang berkomunikasi.

Contoh dalam dunia nyata ialah ketika Misalkan Alice berkomunikasi dengan Bob. Bob memilih pasangan kunci (*e*, *d*). Bob mengirimkan kunci enkripsi *e* (kunci publik) kepada Alice melalui sembarang saluran tetapi tetap menjaga kerahasiaan kunci dekripsinya, *d* (kunci privat).

Kemudian, Alice ingin mengirim pesan *m* kepada Bob. Alice mengenkripsikan pesan *m* dengan menggunakan kunci publik Bob, untuk mendapatkan *c* = *Ee*(*m*), lalu mengirimkan *c* melalui saluran komunikasi (yang tidak perlu aman). Bob mendekripsi cipherteks *c* dengan menggunakan kunci privatnya untuk memperoleh *m* = *Dd*(*c*), Perhatikan skema komunikasi dengan kriptografi kuncipublik pada Gambar 2.



Gambar 2 Enkripsi/dekripsi dengan kritografi kunci-publik

Gambar ini memperlihatkan perbedaan mendasar sistem asimetri dengan sistem simetri. Di sini kunci enkripsi dikirim kepada Alice melalui saluran yang tidak perlu aman (*unsecure channel*). Saluran yang tidak perlu aman ini mungkin sama dengan saluran yang digunakan untuk mengirim cipherteks.

Dengan sistem kriptografi kunci-publik, tidak diperlukan pengiriman kunci privat melalui saluran komunikasi khusus sebagaimana pada sistem kriptografi simetri. Meskipun kunci publik diumumkan ke setiap orang di dalam kelompok, namun kunci publik perlu dilindungi agar otentikasinya terjamin (misalnya tidak diubah oleh orang lain).

Baik kriptografi simetri maupun kriptografi asimetri (kuncipublik),keduanya mempunyai kelebihan dan kelemahan.

Kelebihan dari sistem kriptografi simetri adalah sebagai berikut :

1. Algoritma kriptografi simetri dirancang sehingga proses enkripsi/dekripsi membutuhkan waktu yang singkat.

2. Ukuran kunci simetri relatif pendek.

3. Algoritma kriptografi simetri dapat digunakan untuk membangkitkan bilangan acak.

4. Algorima kriptografi simetri dapat disusun untuk menghasilkan *cipher* yang lebih kuat.

5. Otentikasi pengirim pesan langsung diketahui dari cipherteks yang diterima, karena kunci hanya diketahui oleh pengirim dan penerima pesan saja.

Kelemahan dari sistem kriptografi simetri adalah sebagai berikut :

1. Kunci simetri harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasisan kunci ini.

2. Kunci harus sering diubah, mungkin pada setiap sesi komunikasi.

Kelebihan kriptografi kunci-publik (asimetri):

1. Hanya kunci privat yang perlu dijaga kerahasiaannya oleh seiap entitas yang berkomuniaksi (tetapi, otentikasi kunci publik tetap harus terjamin). Tidak ada kebutuhan mengirim kunci kunci privat sebagaimana pada sistem simetri.

2. Pasangan kunci publik/kunci privat tidak perlu diubah, bahkan dalam periode waktu yang panjang.

3. Dapat digunakan untuk mengamankan pengiriman kunci simetri.

4. Beberapa algoritma kunci-publik dapat digunakan untuk memberi tanda tangan digital pada pesan

Kelemahan kriptografi kunci-publik (asimetri):

1. Enkripsi dan dekripsi data umumnya lebih lambat daripada sistem simetri, karena enkripsi dan dekripsi menggunakan bilangan yang besar dan melibatkan operasi perpangkatan yang besar.
2. Ukuran cipherteks lebih besar daripada plainteks (bisa dua sampai empat kali ukuran plainteks).

3. Ukuran kunci relatif lebih besar daripada ukuran kunci simetri.

4. Karena kunci publik diketahui secara luas dan dapat digunakan setiap orang, maka cipherteks tidak memberikan informasi mengenai otentikasi pengirim.

5. Tidak ada algoritma kunci-publik yang terbukti aman (sama seperti *block cipher*). Kebanyakan algoritma mendasakan keamanannya pada sulitnya memecahkan persoalan - persoalan aritmetik (pemfaktoran,logaritmik,dsb) yang menjadi dasar pembangkitan kunci.

3. Pembahasan *"Truncated Polynomial Rings"* yang digunakan dalam algoritma NTRU

Prinsip dari objek yang digunakan oleh NTRU Public Key Crypto System dalah menggunaan sebuah polinom dengan derajat N-1 seperti berikut :

$$\mathbf{a} = a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots + a_{N-2}X^{N-2} + a_{N-1}X^{N-1}$$

Koefisien dari polinom (a_0,.....a_{N-1}) adalah sebuah integer, beberapa nilai koefisien dapat merupakan bilangan 0.

Set dari sebuah polinomial ini dalam makalah ini akan disebut sebagai **R**. Polinom dalam **R** akan dapat dilakukan operasi pertambahan dengan cara menambahkan koefisien dengan derajat yang bersesuaian. Contoh :

$$\mathbf{a} + \mathbf{b} = (a_0+b_0) + (a_1+b_1)X + \ldots + (a_{N-1}+b_{N-1})X^{N-1}$$

Polinom ini pun dapat dilakukan operasi perkalian terhadapnya dengan cara yang biasa, kecuali dengan satu perubahan yaitu setelah dilakukan proses perkalian pangkat X^N harus diganti dengan 1, pangkat X^{N+1} harus diganti dengan X, pangkat dari X^{N+2} harus diganti dengan X^2 dan seterusnya.
Contoh :

N=3
a=2-X+3X^2
b=1+2X-X^2.

Hasil operasi :

$$\mathbf{a} + \mathbf{b} = (2-X+3X^2) + (1+2X-X^2) = 3+X+2X^2$$

$$\mathbf{a}*\mathbf{b} = (2-X+3X^2)*(1+2X-X^2) = 2+3X-X^2+7X^3-3X^4 = 2+3X-X^2+7-3X = 9-X^2$$

Berdasarkan operasi diatas maka dapat dituliskan rumus untuk perkalian dua polinom dalam **R** sebagai berikut :

$$\mathbf{a}*\mathbf{b} = c_0 + c_1X + c_2X^2 + c_3X^3 + \ldots + c_{N-2}X^{N-2} + c_{N-1}X^{N-1}$$

Dimana koefisien ke Kth dari c_k dapat dituliskan dengan rumus sebagai berikut :

$$c_k = a_0b_k + a_1b_{k-1} + \ldots + a_kb_0 + a_{k+1}b_{N-1} + a_{k+1}b_{N-2} + \ldots + a_{N-1}b_{k+1}$$

Formula diatas memang terlihat sedikit rumit, padahal sebenarnya koefisien c_k adalah hasil kali titik antara 2 buah koefisien a dan b, dengan aturan bahwa koefisien b dibalik dan diputar sepanjang posisi k.

Dengan menggunakan aturan penambahan dan perkalian tersebut maka semua aturan yang terkait dapat pula dilakukan, semisal aturan distribusi pun berlaku contoh :

$$\mathbf{a * (b + c) = a * b + a * c}$$

Dalam terminologi modern proses penambahan dan perkalian tersebut membuat R menjadi sebuah cincin, yang disebut *Ring of Truncated Polynomials*. Dalam bahasa aljabar modern, cincin R tersebut isomorfik dengan persamaan cincin $Z[X]/(X^N - 1)$.

Contoh :

N=7
a=3+2X^2-3X^4+X^6
b=1-3X+X^2+2X^5-X^6
a + **b** = 4 - 3X + 3X^2 - 3X^4 + 2X^5
a*b = 4 - 10X – X^2 - 3X^3 + X^4 + 14X^5 - 5X^6

Algoritma NTRU Public Key Cripto System menggunakan ring of truncated polynomials R digabung dengan aritmetika modulo. Dua hal ini digabungkan dengan melakukan operasi pengurangan koefisien polinom a dengan melakukan operasi modulo dengan integer q, atau dapat ditulis dengan rumus :

a (modulo q)

rumus diatas berarti bahwa dilakukan operasi modulo q terhadap koefisien dari a, sehingga membagi setiap koefisien dengan q dan mengambil sisanya. Dengan begitu operasi :

a = b (modulo q)

berarti bahwa setiap koefisien adalah hasil dari **a-b** adalah kelipatan dari q.

Catatan :

Untuk membuat penyimpanan dan perhitungan lebih mudah, biasanya penyimpanan koefisien dari polinom akan disimpan tanpa penulisan pangkat dari variabel X, sebgai contoh untuk polinom :

a = a_0+ a_1X + a_2X^2 + a_3X^3 + . . . + $a_{N-2}X^{N-2}$ + $a_{N-1}X^{N-1}$

akan ditulis atau disimpan sebagai list dari N buah bilangan sebagai berikut :

a = (a_0, a_1, a_2, . . . , a_{N-2}, a_{N-1})

Perlu diingat bahwa apabila sebuah nilai pangkat dari X hilang maka harus koefisien yang berkaitan harus disimpan sebagai bilangan 0, sebagai contoh untuk sebuah polinom

a=3+2X^2-3X^4+X^6

jika berderajat 7 maka akan disimpan sebagai a = (3,0,2,0,-3,0,1) sedangkan jika berderajat 9 maka akan disimpan sebagai a = (3,0,2,0,-3,0,1,0,0).

4. Pembahasan Algoritma NTRU

NTRU adalah algoritma yang tergolong kedalam algoritma yang menggunakan sistem kunci publik. Algoritma ini dibuat oleh 3 orang yaitu Jeffrey Hoffstein, Jill Pipher dan Joseph Silverman. Menurut pembuatnya algoritma ini diklaim lebih cepat, mudah dalam penciptaan kunci dan juga tidak memerlukan memory yang tinggi dalam prosesnya.

Algoritma ini pertama kali diperkenalkan ke publik pada tahun 1996 dalam sebuah acara yang bernama CRYPTO'96. Perlu dicatat bahwa algoritma ini dipatenkan sehingga untuk penggunaan algoritma ini memerlukan ijin dari pembuat algorima ini, hal ini lah menjadi alasan mengapa algoritma ini jarang dipakai di lingkungan software.

Dasar dari algoritma NTRU ialah sulitnya menemukan vektor yang singkat dari sebuah *lattice* (subgroup diskrit dari sebuah kumpulan vektor yang mencakup seluruh lingkungan vektor). Sehingga dalam prosesnya NTRU menggunakan operasi terhadap polinom. Sehingga dalam prakteknya pesan perlu diubah dahulu kedalam bentuk polinom untuk dapat dilakukan proses melalui sistem ini.

4.1 Parameter dari NTRU Public Key Cripto System

Parameter dari algoritma NTRU ini berkaita dengan masalah tingkat keamanannya. Basis dari koleksi objek yang digunakan oleh NTRU Public Key Criptosystem adalah sebuah cincin R yang berisikan semua polinom terpotong dengan derajat N-1 yang mempunyai koefisien integer sebagai berikut :

a = a0 + a1X + a2X2 + a3X3 + . . . + a*N*-2X*N*-2 + a*N*-1X*N*-1.

Operasi penambahan dapat dilakukan secara biasa terhadap polinom dan juga dapat dilakukan operasi perkalian namun dengan sebuah perbedaan yaitu X^N digantikan oleh 1, X^{n+1} diganti oleh X, X^{n+2} diganti oleh X^2 dan seterusnya.

Dalam implementasi sebenarnya dari NTRU *Public Key Criptosystem* N diisikan dengan sebuah bilangan. Dalam hal ini ada 3 hal yang perlu diperhatikan yaitu :

N : polinom dari R yang mempunyai derajat N-1

q : *large modulus,* koefisien dari R yang dikurangi dengan operasi mod q

p : small moculus: koefisien dari pesan yang dikurangi dengan mod p

Menyangkut masalah kemanan maka perlu dipastikan bahwa p dan q tidak memiliki faktor yang sama. Tabel dibawah menggambarkan nilai-nilai parameter dari NTRU dengan level keamanan yang bersesuaian.

	N	q	p
Keamanan Menengah	167	128	3
Keamanan Standar	251	128	3
Kemanan Tinggi	347	128	3
Kemanan Tertinggi	503	256	3

4.2 Proses Penghasilan Kunci

Karena algoritma ini menggunakan skema kunci yang asimetirik maka dalam proses pembuatan kunci akan dibuat 2 buah tipe kunci yaitu kunci publik dan kunci privat. Proses pembuatan kunci akan dijelaskan sebagai berikut :

1. Secara acak pilih dua buah polinom "kecil" f dan g dari "*ring of truncated polinom*" R seperti yang telah dibahas pada bab sebelumnya.

   Keterangan :polinom kecil adalah sebuah polinom yang relatif kecil terhadap polinom acak mod q. dalam sebuah polinom yang acak maka koefisien akan secara acak terdistribusi dalam mod q, sehingga dalam polinom kecil koefisien akan jauh lebih kecil dari q.

2. Isi dari polinom f dan g bersifat rahasia karena jika salah satunya diketahui oleh pihak lain, maka orang dapat mendekripsikan pesan. Langkah selanjutnya adlaah menghitung invers dar f modulus q dan invers dari f modulo p, kemudian menghitung f_q dan f_p dengan rumus :

   $F*f_q$ = 1 (modulus q)
   $F* f_p$ = 1(modulus p)

   Jika ternyata nilai invers tidak ditemukan maka harus dicari lagi nilai f yang lain.

3. Kemudian setelah itu hitung nilai kunci publik dengan rumus :

   $H = pf_q * g$(modulo q)

Setelah menyelesaikan 3 langkah diatas maka diperoleh 2 pasangan nilai polinom f dan f_p sebagai kunci privat dan sebuah polinom h sebagai kunci publik.

4.3 Contoh Pembuatan Kunci

Sebagai mana telah dijelaskan sebelumnya, bahwa pemilihan parameter akan menyangkut masalah keamanan dalam algoritma ini. Namun sebagai contoh akan dipilih nilai parameter yang rendah, sehingga perhitungan dapat dilakukan dengan mudah. Parameter yang dipilih adalah :

N = 11
q = 32
p = 3

Kita pun perlu mendefinisikan polinomial kecil secara lebih tepat, dalam contoh ini maka kita gunakan nilai d_f dan d_g

1. polinom f memiliki d_f dengan koefisien yang sama dengan +1, (d_f) dengan koefisien yang sama dengan -1 dan sisanya sama dengan 0.

2. polinom g memiliki d_g dengan koefisien yang sama dengan +1, (d_g) dengan koefisien yang sama dengan -1 dan sisanya sama dengan 0.

Alasan dari adanya sedikit perbedaan bentuk dari f dan g ialah karena f harus dapat dibalik sedangkan g tidak.

Dalam contoh ini maka diambil nilai

$df = 4$

$dg = 3$.

Sehingga kita perlu memilih polinom f dengan derajat 10 dengan empat 1 dan tiga -1 dan kita pun perlu memilih polimon g dengan derajat 10 dengan tiga 1 dan tiga -1. Anggap hasil yang kita peroleh adalah :

f = -1 + X + X2 - X4 + X6 + X9 - X10
g = -1 + X2 + X3 + X5 - X8 - X10

Kemudian kita hitung invers dari f_p dari f mod p dan invers dari f_q dari f mod q. hasil yang diperoleh adalah :

$\mathbf{f}p$ = 1 + 2X + 2X3 + 2X4 + X5 + 2X7 + X8 + 2X9

$\mathbf{f}q$ = 5 + 9X + 6X2 + 16X3 + 4X4 + 15X5 + 16X6 + 22X7 + 20X8 + 18X9 + 30X10

Langkah terakhir adalah perhitungan kunci publik

$\mathbf{h} = p\mathbf{f}q*\mathbf{g}$ = 8 + 25X + 22X2 + 20X3 + 12X4 + 24X5 + 15X6 + 19X7 + 12X8 + 19X9 + 16X10 (modulo 32).

Sehingga diperoleh nilai kunci publik yaitu dua buah polinom **f** dan $\mathbf{f_p}$ dan kunci publik polinom **h.**

4. 4 Proses Enkripsi

Langkah-langkah melakukan enkripsi adalah sebagai berikut :

1. Karena algoritma ini menggunakan sistem polinom maka pertama-tama pesan haru diubah ke dalam bentuk polinom anggap polinom yang dihasilkan bernama **m** dengan koefisien yang dipilih adalah modulo p, anggap berada dalam –p/2 dan p/2 (dengan kata lain m adalah polinom kecil dari mod q).

2. Kemudian secara acak pilih polinom kecil lainnya beri nama **r**. ini adalah "*blinding value*" yang digunakan untuk mengamankan sebuah pesan (sama dengan teknik yang digunakan dalam algoritma ElGamal yang menggunakan nilai acak ketika melakukan enkripsi).

3. Kemudian setelah memperoleh nilai m dan r maka perhitungan nilai terenkripi dapat dilakukan dengan rumus :

   e = r*h + m(modulo q)

   polinom e adalah pesan terenkripsi yang hendak dikirimkan/

Contoh proses enkripsi

Sebagai mana telah disebutkan sebelumnya bahwa diperlukan nilai polinom acak ryag merupakan polinom kecil, maka kita mengunakan nilai d_r.

1. polinom r memiliki d_r dengan koefisien yang sama dengan +1, (d_r) dengan koefisien yang sama dengan -1 dan sisanya sama dengan 0.

Dalam contoh ini maka diambil nilai :

$\mathbf{d_r} = 3$.

Kemudian anggap bahwa pesan yang ingin dikirimkan telah dalam bentuk polinom yaitu :

m = -1 + X3 - X4 - X8 + X9 + X10

Dengan kunci publik yang ada dalam contoh sebelumnya yaitu :

h = 8 + 25X + 22X2 + 20X3 + 12X4 + 24X5 + 15X6 + 19X7 + 12X8 + 19X9 + 16X10.

Kemudian hitung polinom acak r dengan derajat 10 dengan tiga 1 dan tiga -1. Anggap bahwa hasil yang diperoleh adalah :

r = -1 + X2 + X3 + X4 - X5 - X7.

Maka berdasarkan rumus enkripsi diperoleh nilai pesan yang terenkripsi sebagai berikut :

e = **r*h** + **m** = 14 + 11X + 26X2 + 24X3 + 14X4 + 16X5 + 30X6 + 7X7 + 25X8 + 6X9 + 19X10 (modulo 32).

e adalah pesan yang terenkripsi dan siap dikirim. Terlihat di sini bahwa pesan yang terkirim adalah dalam bentuk polinom.

4.5 Proses Dekripsi

Langkah-langkah melakukan proses dekripsi adalah sebagai berikut :

1. Lakukan perhitungan polinom pesan **e** dengan menggunakan polinom privat **p** dengan rumus :

   a = **f*e** (modulo *q*).

2. Kemudian karena nilai **a** dihitung dari modulus q, maka dipilihlah koefisien **a** yang berada dalam -q/2 dan q/2. Hal ini sangat penting dilakukan untuk dilakukan sebelum dilakukan perhitungan polinom b dengan rumus :

   b = **a** (modulo *p*)

3. Setelah dilakukan perhitungan pengurangan koefisien **a** maka digunakan polinom privat $\mathbf{f_p}$ untuk menghitung nilai pesan dengan rumus :

   c = **f***p***b** (modulo *p*)

Polinom **c** adalah pesan yang dikirimkan

4.6 Contoh dekripsi

Dalam contoh ini anggap bahwa pesan terenkripsi yang diterima adalah :

e = 14 + 11X + 26X2 + 24X3 + 14X4 + 16X5 + 30X6 + 7X7 + 25X8 + 6X9 + 19X10

Kemudian digunakan kunci privat f untuk menghitung nilai a :

a = **f*e** = 3 - 7X - 10X2 - 11X3 + 10X4 + 7X5 + 6X6 + 7X7+ 5X8 - 3X9 - 7X10 (modulo 32).

Perlu diperhatikan bahwa dilakukan pengurangan jumlah koefisien dengan nilai yang berada antara -15 dan 16 bukan antara 0 dan 31. hal ini sangan penting dilakukan seperti telah dijelaskan dalam proses dekripsi.

Kemudian dilakukan pengurangan koefisien a dengan melakukan operasi a modulo 3 :

b = **a** = - X - X2 + X3 + X4 + X5 + X7 - X8 - X10 (modulo 3).

Terakhir digunakan kunci privat yang lainnya, yaitu f_p untuk menghitung nilai akhir yaitu :

c = **f***p***b** = - 1 + X3 - X4 - X8 + X9 + X10 (modulo 3).

C adalah pesan yang dikirimkan, terlihat bahwa pesan yang dihasilkan identik dengan pesan yang ada dalam contoh enkripsi sebelumnya.

4.6 Penjelasan tambahan

Setelah melihat proses enkripsi dan dekripsi diatas tentunya ada sebuah hal yang aneh yaitu ketika pengirim mengirimkan pesan dengan menggunakan rumus **e = r*h + m**(modulo q). terlihat dalam rumus tersebut bahwa ada dua nilai yang tidak diketahui oleh penerima,yaitu nilai **r** dan **m**. Namun hal ini tidak menjadi masalah karena pada langkah dekripsi pertama-tama dilakukan perhitungan **f*e** dan pengurangan koefisien dengan modulo q. Perlu diingat bahwa nilai kunci publik **h** dihasilkan dari rumus $\mathbf{pf_q*g}$ dan dilakukan pengurangan koefisien dengan modulo q. sehingga meskipun penerima tidak mengetahui nilai **r** dan **m** ketika penerima melakukan penghitungan **a= f*e** (modulo q) ia sebenarnya melakukan perhitungan sebagai berikut :

a = **f*e** (modulo *q*)
= **f***(**r*h** + **m**) (modulo *q*)
[**e** = **r*h** + **m** (modulo *q*)]
= **f***(**r***p***f***q***g** + **m**) (modulo *q*)
[**h** = *p***f***q***g** (modulo *q*)]
= *p***r*g** + **f*m** (modulo *q*)
[**f*f***q** = 1 (modulo *q*)]

Sekarang lihat kembali kepada ukurandari parameter yang ada, polinom **r**,**g**,**f** dan **m** semuanya memiliki koefisien yang cukup kecil. Hal ini berarti koefisien dari produk **r*g** dan **f*m** juga berukuran kecil jika dibandingkan dengan **q**. Karena **p** juga kecil jika dibandingkan terhadap **p**, maka dapat diartikan bahwa koefisien polinom **pr*g** dan **f*m** sudah berada dalam –q/2 dan q/2, sehingga mengurangi koefisien modulo q tidak mempunyai efek apapun.

Dengan kata lain ketika penerima menghitung **a** dengan mengalikannya terhadap **f*e** dan mengurangi koefisien dengan modulo q, polinomial **a** adalah polinom yang tepat sama dengan **pr*g + f*m**. Ketika penerima mengurani koefisien **a** dengan melakukan operasi modulo p untuk menghasilkan polinom **b**, ia sebenarnya melakukan pengurangan koefisien terhadap **pr*g + f*m** modulo p, sehingga **b** pada akhirnya sama dengan :

$$\mathbf{b} = \mathbf{f}*\mathbf{m} \text{ (modulo } p\text{)}$$

Perlu diperhatikan bahwa sebenarnya pengirim belum mengetahui nilai dari **m**, namun karena ia mengetahui nilai dari maka dalam langkah terakhir dilakukan perkalian **b** oleh $\mathbf{f}_p$ dan gunakan fakta bahwa **fp*f** = 1 (modulo p) untuk menghitung :

$$\mathbf{c} = \mathbf{f}p* \mathbf{b} = \mathbf{f}p* \mathbf{f}*\mathbf{m} = \mathbf{m} \text{ (modulo } p\text{)}$$

Hal ini lah yang membuat penerima mampu menghitung nilai polinom yang dikirimkan.

5. Perbandingan NTRU dengan Algoritma Sistem Kunci Publik yang Lain

Dalam bidang *public key cripto system* atau yang biasa disebut kriptografi kunci asimetrik ada berbagai macam algoritma, sehingga dapat dilakukan perbandingan terhadap NTRU. Dalam makalah ini akan dilakukan perbandingan terhadap 3 algoritma dengan jenis yang serupa yaitu :

1. Rivest, Shamir , Adelman (RSA)
   Algoritma ini memiliki keamanan dari sulitnya memfaktorkan sebuah bilangan. Algoritma NTRU hanya memiliki kesamaan yang sedikit dengan RSA. Kesamaan hanya terletak pada digunakannya operasi modulus.

2. McEliece
   Algoritma ini berdasarkan dari error correcting code yang disebut "*Goppa code*". Algoritma NTRU memiliki beberapa fitur yang hampir sama dengan McEliece, yaitu perkalian cincin R yang dapat diformulasikan sebagai matrik dan enkripsi dari kedua buah sistem dapat ditulis sebagai perkalian matrik E = AX + Y, dimana A adalah kunci publik. Perbedaan antara kedua buah sistem ialah pada NTRU Y adalah pesan dan X adalah vektor acak, sedangkan pada McEliece Y adalah vektor acak dan X adalah pesan. Sehingga perbedaanya ialah pada letak variabel dalam rumus. Tetapi perbedaan sebenarnya dari dua buah algoritma ini adalah adanya *trap-door* yang digunakan pada dekripsi. Untuk algoritma McEliece, matrix A diasosiasikan dengan *error corecting (Goppa) code*. Sedangkan untuk NTRU matriks A adalah matrik *circulant* dan dekripdi tergantung dari dekomposisi A menjadi produk dari dua buah matrik yang mempunyai bentuk berbeda.

3. Goldreich, Goldwasser dan Halevi (GGH)
   Algoritma ini berdasarkan dari sulitnya menemukan bilangan yang hampir *orthogonalized* dalam *lattice*. Meskipun NTRU system pun harus menjaga adanya serangan terhadap pengurangan *lattice*, namun cara dekripsi amat berbeda dengan GGH yang mana dalam GGH dekripsi berdasarkan dari pengetahian akan *lattice* yang pendek. Dalam hal ini GCH sebenarnya mengunakan algoritma McEliece karena dalam kedua belah kasus dekripsi dilakukan berdasarkan pengenalan dan penghilangan bilangan acak. Berbeda dengan dua algoritma ini NTRU menghilangkan bilangan acak dengan menggunakan pembagian, tanpa perlu mengetahui bilangan acak tersebut.

Tabel dibawah menggambarkan perbandingan dari aspek operasi dari RSA, Mceliece, GGH dan NTRU. N menggambarkan parameter panjang pesan.

	NTRU	RSA	McEliece	GGH
Kecepatan Enkripsi	N^2	N^2	N^2	N^2
Kecepatan Dekripsi	N^2	N^3	N^2	N^2
Kunci Publik	N	N	N^2	N^2
Kunci Privat	N	N	N^2	N^2

Dalam proses kecepatan dilakukan proses perbandingan antara NTRU dengan RSA berdasarkan sumber dari NTRU tech dan website dari RSA. Analisis kecepatan hanya dilakukan terhadap 2 buah algoritma ini karena hanya ditemukan sumber yang cocok untuk 2 algoritma

ini . Pengujian terhadap algoritma lain sulit dilakukan karena sifat dari algoritma NTRU yang dipatenkan dan hampir tidak ada perangakat lunak *open source* yang menggunakan algoritma NTRU.

Analisis Pengujian dilakukan dengan melihat fakta yang dilampirkan di website NTRU system dan RSA. Pengujian yang dilampirkan oleh masing sumber menghitung berapa blok yang berhasil diekripsi dalam 1 detik dan juga waktu penciptaan kunci dengan tingkat keamanan yang berbeda-beda.

Berdasarkan hasil yang ada di website masing-masing maka pengujian dilakukan dalam 4 macam platform yaitu :

1. Untuk NTRU dengan processor Pentium 75 MHz, dijalankan diatas DOS
2. Untuk NTRU dengan processor Pentium 200 MHz, dijalankan diatas Linux
3. Untuk RSA dengan processor Pentium 90 MHz
4. Untuk RSA dengan processor Digital AlphaStation 255 MHz

Hasil Pengujian yang digunakan untuk perbandingan adalah sebagai berikut :

1. NTRU : Pentium 75 MHz, dijalankan diatas DOS

Tingkat Keamanan	Enkripsi	Dekripsi	Pembuatan Kunci
Menengah	1818	505	0,1080
Tinggi	649	164	0,1555
Tertinggi	103	19	0,8571

2. Untuk NTRU dengan processor Pentium 200 MHz, dijalankan diatas Linux

Tingkat Keamanan	Enkripsi	Dekripsi	Pembuatan Kunci
Menengah	16666	2273	0,0079
Tinggi	4762	724	0,0184
Tertinggi	730	79	0,1528

3. Untuk RSA dengan processor Pentium 90 MHz

Tingkat Keamanan	Enkripsi	Dekripsi	Pembuatan Kunci
512 bit	370	42	0,45
768 bit	189	15	1,5
1024 bit	116	7	3,8

4. Untuk RSA dengan processor Digital AlphaStation 255 MHz

Tingkat Keamanan	Enkripsi	Dekripsi	Pembuatan Kunci
512 bit	1020	125	0,26
768 bit	588	42	0,59
1024 bit	385	23	1,28

Perbandingan antara NTRU dan RSA dalam sistem Pentium 75 MHz dan 90MHz, dengan adanya perbedaan kecepatan prosessor maka perbandingan tingkat keamanan NTRU yang moderate dibandingkan dengan keamanan RSA yang 512 bit. Sehingga dapat disimpulkan bahwa NTRU 5,9 kali lebih cepat pada proses enkripsi dan 14,4 kali lebih cepat pada proses dekripsi dan 5 kali lebih cepat pada proses penciptaan kunci. Sedangkan pada level keamanan NTRU yang tertinggi dibandingkan dengan RSA dengan panjang kunci 1024 bit maka dapat disimpulkan bahwa dalam proses enkripsi hasil antara kedua algoritma adalah sama dan untuk proses dekripsi NTRU 3,2 kali lebih cepat serta untuk proses penciptaan kunci NTRU 5,3 kali lebih cepat.

Dalam perbandingan untuk sistem Pentium 200 MHz dan Digital Alpha 200 MHz perbandingan agak sulit dilakukan karena kedua buah sistem berbeda dalam banyak hal. Namun jika hanya membandingkan secara kasar maka pada keamanan dengan tingkat menenganh NTRU lebih cepat 16 kali pada enkripsi, 18 kali pada dekripsi dan 33 kali lebih cepat pada proses pembuatan kunci. Sedangkan ada tingkat keamanan yang tinggi maka NTRU lebih cepat 2 kali pada enkripsi, 3 kali lebih cepat pada dekripsi dan 8 kali lebih cepat pada proses pembuatan kunci. Dari hasil perbandingan terlihat bahwa secara umum NTRU lebih cepat dari RSA.

6. Pembahasan kemungkinan serangan terhadap NTRU

Dalam makalah ini pembahasan terhadap tipe serangan hanya dibatasi pada 3 tipe serangan yaitu :

1. Brute Force Attack

   Dalam model serangan ini, maka penyerang akan berusaha untuk mengetahui kunci dengan cara mencoba semua kemungkinan polinom f dalam *lattice* fdan melihat apakah f* h (mod q) mempunyai hasil yang kecil atau dengan mencoba polinom g dalam *lattice* gdan

mengecek apakah g mempunyai hasil yang kecil. Dengan cara yang sama penyerang akan mencoba untuk mengetahui pesan dengan cara mencari semua kemungkinan polinom Ǿ dalam *lattice* Ǿ dan mengecek apakah e - Ǿ * h (mod q) mempunyai hasil yang kecil.

Dalam prakteknya *lattice* g lebih kecil dari *lattice* g, sehingga keamanan kunci ditentukan dari kumpulan polinom dari *lattice* g ($\#\mathcal{L}_f$) dan keamanan dari masing-masing pesan ditentukan dari lattice Ǿ ($\#\mathcal{L}_{Ǿ}$). Sehingga tingkat kemanan sistem jika diserang dengan brute force attack akan ditentukan oleh persamaan :

$$\binom{\text{Key}}{\text{Security}} = \sqrt{\#\mathcal{L}_g} = \frac{1}{d_g!}\sqrt{\frac{N!}{(N-2d_g)!}}$$

$$\binom{\text{Message}}{\text{Security}} = \sqrt{\#\mathcal{L}_\phi} = \frac{1}{d!}\sqrt{\frac{N!}{(N-2d)!}}.$$

Sebagai contoh untuk melakukan pencarian terhadap sebuah nilai f jika N = 251 dan tiap f memiliki 8 buah nilai 1 maka jumlah kemungkian f yang ada adalah 251^3 buah

2. Meet-in –the-middle Attack

Dalam model serangan ini, anggap bahwa pesan yang terenkripsi berbentuk : e = Ǿ * h + m (mod q). Andre Odlyzko menyatakan bahwa serangan meet in the middle dapat digunakan untuk Ǿ, dan kita pun dapat mengamati bahwa serangan dengan tipe yang sama dapat dilakukan terhadap kunci privat f. Anggap bahwa f dibagi menjadi 2 yaitu $f = f_1 + f_2$ dan ada sebuah kesamaan dimana $f_1 * e - f_2 * e$, sehingga ada f_1 dan f_2 dengan jumlah yang sama. Sehingga dalam proses penjagaan keamanan sistem keamanan dengan derajat 2^{80} maka nilai f,g, Ǿ harus dipilih dari set yang mengandung 2^{160} elemen.

3. Multiple Transmission Attack

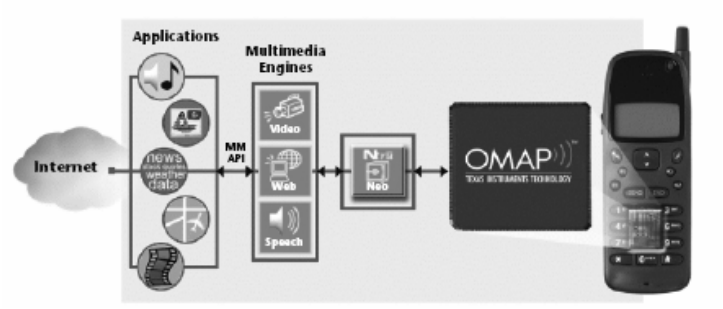
Dalam model serangan ini, anggap bahwa pengirim mengirimkan pesan yang sama berulang kali dengan kunci publik yang sama namun Ǿ yang berbeda maka si penyerang akan dapat mengetahui sebgai besar dari pesan. Secara singkat , andaikata pengirim mengirimkan e_i = Ǿ* h + m (mod q) untuk i=1,2,....,r maka penyerang akan dapat menghitung ($e_i - e_1$) * h^{-1} (mod q) sehingga memperoleh $Ǿ_i$- $Ǿ_1$ (mod q).

Namun karena koefisien dari Ǿ adalah kecil sehingga ketika penyerang berhasil menemukan $Ǿ_i$- $Ǿ_1$ ia tetap akan menemukan banyak koefisien dari $Ǿ_1$. apabila r ada dalam ukuran yang sedang (anggap 4 atau 5) maka penyerang harus mencari $Ǿ_1$ dalam jumlah yang cukup untuk mencari semua kemungkinan dari sisa koefisien dengan cara brute-force, untuk mencari nilai M. Sehingga dapat disimpulkan bahwa apabila penyerang akan mendekripsikan pesan dengan teknik ini maka informasi yang diperoleh tidak akan membantu dalam proses dekripsi.

7. Produk yang menggunakan algoritma NTRU

Dalam prakteknya algoritma NTRU banyak digunakan dalam dunia kriptografi kunci publik yang bergerak dalam bidang keamana pesan dalam dunia wireless. Hal ini didasarkan karena kecepatan algoritma NTRU yang dirasakan lebih cepat ketimbang RSA. Namun karena sifat dari algoritma ini yang masih dipatenkan maka jarang sekali ada aplikasi yang menggunakan algoritma ini.

Aplikasi yang menggunakan algoritma ini kebanyakan diproduksi oleh perusahaan yang bernama NTRU CryptoSystem. Salah satu applikasi yang menarik dari NTRU ialah dengan adanya kerjasama dengan Texas instrument untuk membuat sebuah chip yang dilengkapi dengan kemampuan menghitung algoritma NTRU. Chip tersebut bernamakan NTRU Neo OMAP.



Gambar 3 Diagram Penggunaan Chip Neo OMAP

Gambar 3 menggambarkan bagaimana sebuah *Chip* NTRU Neo OMAP digunakan dalam dunia keamanan di bidang *wireless*. Sehingga dengan adanya chip ini maka para pengembang aplikasi yang berbasiskan *wireless* dapat mengembangkan aplikasi yang menggunakan jalur komunikasi yang aman dan dapt pula membuat sebuah aplikasi yang real time tanpa perlu adanya delay dan komplesitas yang rumit. Sebgai hasilnya chip ini banyak digunakan dalam aplikasi yang membutuhkan keamanan dan performa yang cepat, sehingga aplikasi ini banyak digunakan dalam bidang :

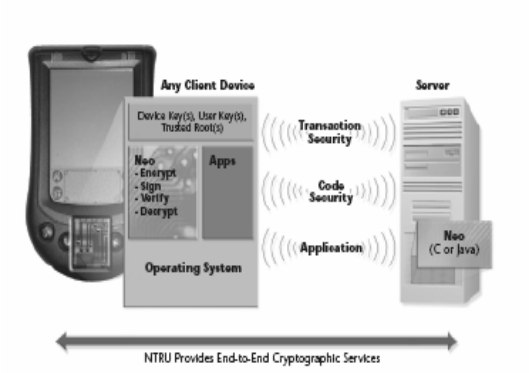
1. Distribusi Media Digital
2. Transaksi Perbangkan
3. Pengiriman Pesan Multimedia
4. Dunia Permainan
5. Platform Java
6. Perdagangan yang bersifat *mobile*.
7. Servis berbasis Lokasi
8. Konferensi Video

Hasil yang diperoleh pun cukup cepat dan menurut situs NTRU algoritmanya diklaim dapat berjalan jauh lebih cepat dengan perbandingan sebagai berikut :

	NTRU	RSA	Perbedaan performa
Pembuatan Kunci	153	0,5	306 kali lebih cepat
Enkripsi	3161	964	3,3 kali lebih cepat
Dekripsi	1653	23	72 kali lebih cepat

Selain kerjasama dengan pihak pembuat chip yang menggabungkan algoritma NTRU langsung kedalam perangkat keras sehingga perhitungan dapat menjadi lebih cepat. Algoritma NTRU pun diterapkan untuk menjadi sebuah API (Application Programming Interface), sehingga dapat diterapkan langsung kedalam perangkat lunak yang memerlukan dukungan keamanan dari sistem kriptografi kunci publik NTRU.

Diagram yang menggambarkan bagaimana letak dari API ini dalam sebuah aplikasi adalah sebagai berikut :



Gambar 4 Diagram Penggunaan API NTRU CryptoSystem

API untuk NTRU pun telah dibuat oleh NTRU Criptosystem. API ini ada dalam 2 edisi yaitu :

1. NTRU NEO C
   API ini dirancang untuk digunakan dalam semua lingkungan pemrogramman yang menggunakan bahasa C/C++. Hal ini karenaNEO C ditulis dengan bahasa C dan sesuai dengan ANSI. Sehingga membuat tingkat portabilitasnya yang tinggi. API ini berukuran tidak terlalu besar yaitu hanya 83 kb untuk Win32 dan 81 kb untuk WinCE. Didalam API ini pun tidak hanya disertakan fungsi-fungsi untuk melakukan perhitungan dalam NTRU saja namun pula ditambahkan beberapa fungsi lain, fungsi-fungsi yang ada dalam API ini adalah :

   1. Enkripsi
   2. Dekripsi
   3. Pembuatan Kunci
   4. Tanda tangan Digital
   5. Verifikasi
   6. SHA-1
   7. AES
   8. IEEE P1362 Standard PRNG

Seperti yang terlihat dalam daftar fungsi diatas NTRU NEO C pun mempunyai fungsi *hashing* seperti SHA-1. Hal ini memungkinkan perpaduan antara fungsi *hashing* dan sistem kriptografi kunci publik NTRU untuk dipadukan dalam membentuk sebuah sistem tandatangan digital.

2. NTRU Neo for Java
   API untuk platform java ini dirancang untuk dapat berjalan dalam semua platform java. Hal ini mencakup server, desktop dampai dengan handphone yang mendukung java. Paket dari API ini berukuran cukup kecil yaitu hanya 3,8 kilobytes. Paket ini pun sudah termasuk dengan algoritma SHA yang digunakan untuk hashing dokumen apabila ingin menggunakan tandatangan digital. Dengan ukuran yang sangat kecil tersebut maka API ini sudah sangat layak untuk dipakai dalam J2ME yang mempunyai kebatasan media penyimpanan. Sama seperti versi C/C++ didalam API ini pun tidak hanya disertakan fungsi-fungsi untuk melakukan perhitungan dalam NTRU saja namun pula ditambahkan beberapa fungsi lain, fungsi-fungsi yang ada dalam API ini adalah :

   1. Enkripsi
   2. Dekripsi
   3. Pembuatan Kunci
   4. Tanda tangan Digital
   5. Verifikasi
   6. SHA-1
   7. AES
   8. IEEE P1362 Standard PRNG

   Dari daftar diatas terlihat bahwa ternyata dalam versi Java yang dapat berjalan diatas media handphone tidak ada pengurangan fitur dari API jika dibandingkan dengan API dalam bahasa C. Sehingga kemampuan dan fungsionalitas dari dua buah API berbeda bahasa ini dapat dikatakan sama.

Dengan adanya API ini maka para pengembang perangkat lunak dapat menambahkan fitur keamanan yang menggunakan algoritma NTRU kedalam perangkat lunak yang mereka kembangkan tanpa perlu repot. Hanya salah satu kekurangan dari API ini ialah sifatnya yang *proprietary* sehingga untuk penggunaannya diperlukan lisensi atau ijin dari NTRU CryptoSystem selaku pemegang lisensi dari algoritma NTRU. Hal inilah yang nampaknya mendasari alasan ketidak populeran algoritma ini dikalangan para pengembang perangkat lunak ialah karena kesulitan untuk menggunakan algoritma ini dalam sistem keamanan yang berbasis *open source*. Kesulitan untuk melakukan pembuatan alogritma ini pun dirasakan sulit karena situs pembuat jelas-jelas memasang lisensi yang menghambat orang lain untuk mengembangkan sistem keamanan dengan berbasiskan NTRU.

8. Kesimpulan

1. Algoritma ini dinilai lebih baik dalam kecepatannya ketimbang dengan algoritma RSA, karena setelah meliha beberapa hasil pengujian yang berjalan di platform PC maupun chip yang kecil , terlihat bahwa NTRU memiliki kecepatan yang lebih cepat dari RSA.

2. Algoritma ini lebih baik dalam melakukan pengiriman pesan, karena pesan dikirimkan dalam bentuk polinom sehingga pengirim cukup mengirimkan pesan dalam bentuk polinom dan menyertakan nilai dari variabel yang digunakan dalam polinom tersebut. Sehingga dapat menghemat tempat untuk melakukan pengiriman pesan, sehingga algoritma ini sangat cocok diterapkan untuk dunia wireless yang memiliki banyak keterbatasa dalam segi tempat untuk melakukan pengiriman/penerimaan data.

3. Algoritma ini dinilai cukup aman karena adanya kesulitan untuk mencari *lattice* dari lingkungan polinom yang dipakai oleh algoritma ini, penggunaan bilangan acak pun menambah kesulitan penyerangan terhadap alogritma ini.

4. Algoritma ini cukup sulit untuk diimplementasikan. Hal ini karena algoritma ini menggunakan polinom sebagai basis dari perhitungannya. Sehingga mempersulit implementasinya dalam bahasa pemrograman. Hal lain yang membuat sulitnya implementasi dari algoritma ini adalah adanya batasan hak paten dari algoritma ini. Sehingga tidak

ada produk open-source yang memakai algoritma ini.

5. Algoritma ini sangat cocok untuk diterapkan dalam sistem keamanan dunia wireless, karena selain lebih cepat berdasarkan technical report yang diberikan oleh NTRU CryptoSystem algoritma ini dapat menghemat data yang diperlukan untuk melakukan pengiriman.

9. Daftar Pustaka

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] Hoffstein Jeffrey, Jill Pipher, Joseph H. Silverman NTRU: A Ring Based Public Key Cryptosystem.Springer-Verlag, Berlin, 1998, halaman 267-288.

[3] J. Hoffstein, J. Silverman,Optimizations for NTRU, Walter de Gruyter, Berlin-New York, 2001.

[4] Hoffstein & Silverman ,Random Small Hamming Weight Products with Applications to Cryptography, Com2MaC ,2000.

[5] Cohen , A Course in Computational Algebraic Number Theory, Springer-Verlag, Berlin, 1993.

[6] Silverman ,A Friendly Introduction to Number Theory, Prentice-Hall, New Jersey, 1997.

[7] Cryptography: Theory and Practice, D. Stinson, CRC Press, Boca Raton, 1995.

[8] Menezes , Van Oorschot, Handbook of Cryptography, CRC Press, Boca Raton, 1996.

[9] Pembahasan singkat algoritma NTRUSign
http://en.wikipedia.org/wiki/NTRUSign
Tanggal akses: 20 Desember 2005 pukul 08:25.

[10] Situs resmi pembuat NTRUSign
http://www.ntru.com/
Tanggal akses: 20 Desember 2005 pukul 08:30.

[11] Pembahasan mendalam tentang algoritma NTRU
http://www.ntru.com/cryptolab/pdf/ANTS97.pdf
Tanggal akses: 20 Desember 2005 pukul 08:35.

[12] Tutorial cara kerja dari algoritma NTRU
http://www.ntru.com/cryptolab/pdf/ntrututorials.pdf
Tanggal akses: 28 Desember 2005 pukul 13:35.

[13] Pembahasan tentang pemilihan parameter dalam algoritma NTRU
http://www.ntru.com/cryptolab/pdf/ntru-sves-params-website.pdf
Tanggal akses: 28 Desember 2005 pukul 13:40 .

[14] Pembahasan mendalam tentang API NEO C
http://www.ntru.com/cryptolab/pdf/Neo_C1.pdf
Tanggal akses: 28 Desember 2005 pukul 13:40 .

[15] Pembahasan mendalam API Neo Java
http://www.ntru.com/cryptolab/pdf/Neo_Java1.pdf
Tanggal akses: 28 Desember 2005 pukul 13:41 .

[16] Pembahasan tentang Chip Neo OMAP
http://www.ntru.com/cryptolab/pdf/NeoOMAP1.pdf
Tanggal akses: 28 Desember 2005 pukul 13:42 .

[17] Penggunaan NTRU dalam dunia Wireless
http://www.ntru.com/cryptolab/pdf/wireless2.pdf
Tanggal akses: 28 Desember 2005 pukul 13:40 .

[18] Pembahasan tentang lattice dalam NTRU
http://www.ntru.com/cryptolab/pdf/NTRULattice-2005-1.pdf
Tanggal akses: 28 Desember 2005 pukul 14:00 .

[19] Pembahasan tentang lattice dalam NTRU
http://www.ntru.com/cryptolab/pdf/NTRULattice-2005-1.pdf
Tanggal akses: 28 Desember 2005 pukul 13:44 .

[20] Penjelasan tentang lattice
http://en.wikipedia.org/wiki/Lattice_(group)
29
Tanggal akses: 28 Desember 2005 pukul 14:20 .

[21]Pembahasan algoritma Goldreich, Goldwasser dan Halevi (GGH)
http://en.wikipedia.org/wiki/GGH_signature_scheme
Tanggal akses: 28 Desember 2005 pukul 13:45

[22] Pembahasan algoritma RSA
http://en.wikipedia.org/wiki/RSA
Tanggal akses: 28 Desember 2005 pukul 13:45 .

[23] Spesifikasi NTRU
http://www.szydlo.com/ntru-revised-full02.pdf
Tanggal akses: 28 Desember 2005 pukul 13:45 .

[23] Kumpulan paper tentang NTRU
http://grouper.ieee.org/groups/1363/lattPK/submissions.html#2005-08
Tanggal akses: 28 Desember 2005 pukul 13:00 .

[24] Keamanan dalam NTRU
http://eprint.iacr.org/2006/387
Tanggal akses: 28 Desember 2005 pukul 16:00 .

[25] Kemungkinan kegagalan dalam keamanan NTRU
http://www.ntru.com/cryptolab/pdf/
nr03_ntru.pdf
Tanggal akses: 28 Desember 2005 pukul 16:20 .

[26] Keamanan Bit dalam NTRU
http://www.ntru.com/cryptolab/pdf/
NTRU-bits.pdf
Tanggal akses: 28 Desember 2005 pukul 16:20 .

[27] Pembahasan NSS (Tandatangan digital yang menggunakan NTRU)
http://www.ntru.com/cryptolab/pdf/
nss.pdf
Tanggal akses: 28 Desember 2005 pukul 16:20 .

[28] Pernggunaan NTRU dalam Constrained Devices.
http://www.ntru.com/cryptolab/pdf/
ntruches2001.pdf
Tanggal akses: 28 Desember 2005 pukul 16:20 .

.