Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.201853844
Vol. 5, No.3, Agustus 2018, hlm. 325-x p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 51/E/KPT/2017 e-ISSN: 2528-6579

ANALISA KOMBINASI ALGORITMA MERKLE-HELLMAN KNAPSACK DAN LOGARITMA DISKRIT PADA APLIKASI CHAT

Aminudin[1], Ahmad Faisal Helmi[2], Sofyan Arifianto[3]

[1,2,3]Jurusan Teknik Informatika, Universitas Muhammadiyah Malang, Malang, Indonesia
Email: [1]aminudin2008@gmail.com,[2]faisalhelmi99@gmail.com, [3]sofyan.arifianto@gmail.com

(Naskah masuk: 6 Juni 2018, diterima untuk diterbitkan: 8 Agustus 2018)

Abstrak

Informasi melalui jaringan internet sangat rentan terhadap penyadapan oleh pihak yang tidak bertanggung jawab. Agar informasi tersebut aman, maka dibutuhkan teknik kriptografi untuk melindungi dan mengamankan informasi tersebut. Salah satu contoh algoritma kriptografi yang dapat digunakan untuk mengamankan informasi adalah algoritma Merkle-Hellman Knapsack. Akan tetapi, algoritma ini sudah dinyatakan tidak aman karena sudah dapat dipecahkan oleh Shamir (1984). Beberapa tahun terakhir muncul perkembangan dari algoritma knapsack yaitu kombinasi algoritma knapsack dan logaritma diskrit. Algoritma kombinasi ini diklaim lebih aman daripada algoritma knapsack karena pada kombinasi algoritma ini dibutuhkan dua kali proses enkripsi dan dua kali dekripsi sehingga kriptosistem dari kombinasi algoritma ini lebih baik daripada algoritma knapsack. Berdasarkan hasil pengujian performa algoritma didapatkan bahwa waktu pembangkitan kunci, waktu enkripsi dan waktu dekripsi algoritma gabungan knapsack dengan logaritma diskrit memilki waktu yang lebih lama pemrosesannya dibandingkan dengan algoritma knapsack standard. Kemudian untuk pengujian keamanan algoritma dengan menggunakan metode *avalanche effect* didapatkan hasil bahwa gabungan knapsack dengan logaritma diskrit memiliki pengubahan bit yang signifikan daripada knapsack standard yaitu mencapai 3x lipatnya. Serta pengujian *known plainteks attact* terbukti bahwa penggunaan bit 1024 pada algoritma knapsack ditemukan 14% sedangkan gabungan algoritma ditemukan plainteks sebesar 11,60 %.

Kata kunci: *algoritma merkle-hellman knapsack, logaritma diskrit, avalanche effect, known plainteks attact*

ANALYSIS OF A COMBINATION OF MERKLE-HELLMAN ALGORITHMS AND DISCRETE LOGARITHMS IN CHAT APPLICATION

Abstract

Information through the internet network is very vulnerable to wiretapping by irresponsible parties. In order for the information to be safe, cryptographic techniques are needed to protect and secure the information. One example of a cryptographic algorithm that can be used to secure information is the Knapsack Merkle-Hellman algorithm. However, this algorithm has been declared unsafe because it can already be solved by Shamir (1984). In recent years the development of the knapsack algorithm has emerged, namely the combination of knapsack algorithms and discrete logarithms. This combination algorithm is claimed to be safe than the knapsack algorithm because in this combination of algorithms it takes twice the encryption process and twice decryption so that the cryptosystem of this algorithm combination is better than the knapsack algorithm. Based on the results of the algorithm performance testing, it is found that the key generation time, encryption time and decryption time of the combined knapsack algorithm with discrete logarithms have a longer processing time compared to the standard knapsack algorithm. Then to test the security of the algorithm using the Avalanche effect method, it was found that the combined knapsack with discrete logarithms had a significant bit conversion than the standard knapsack which reached 3 times maximum. As well as testing the known plaintext attact, it was proven that the use of 1024 bits in the knapsack algorithm was found to be 14% while the algorithm combined found plaintext at 11.60%.

Keywords: *merkle-hellman knapsack Algorithm, discrete logarithm, avalanche effect, known plainteks attact*

1. PENDAHULUAN

Informasi yang melalui jaringan internet sangat rentan terhadap penyadapan oleh pihak lain. Penyadap akan mudah mengetahui informasi yang didapat apabila informasi tersebut tidak dilindungi (Fadlan & Hadriansa, 2017). Masalah keamanan informasi tersebut juga rentan terhadap ketika akan membangun aplikasi chat. Aplikasi chat dinilai masih kurang dalam segi keamanannya. Kekurangan aplikasi chat adalah dapat memberikan jalan terbuka dari serangan penyadap (Chouhan & Ravi, 2013). Agar sebuah pesan tersebut aman dari penyadap,

dibutuhkan teknik enkripsi dan dekripsi di aplikasi chat. Algoritma kriptografi asimetris merupakan algoritma yang cocok untuk diimplementasikan ke dalam aplikasi chat (A Hidayat & Akmal, 2016). Skema dari kriptografi asimetris adalah semua orang dapat mengenkripsi pesan menggunakan kunci publik, tetapi hanya orang yang mempunyai kunci privat untuk mendekripsikannya (Chouhan & Ravi, 2013).

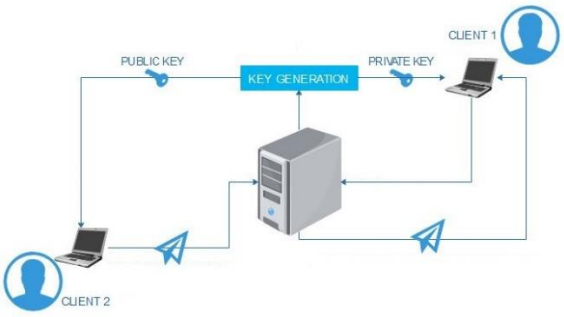
Salah satu algoritma kriptogafi asimetris adalah algoritma merkle-hellman knapsack. Akan tetapi algoritma ini sudah dinyatakan tidak aman karena sudah dipecahkan oleh Shamir (Shamir, 1984). Beberapa tahun terakhir muncul perkembangan dari algoritma knapsack yaitu kombinasi algoritma knapsack dan logaritma diskrit. Kombinasi algoritma ini diklaim lebih aman daripada algoritma knapsack karena pada proses enkripsi dilakukan dua kali. Kombinasi algoritma knapsack dengan logaritma diskrit dijelaskan pada penelitian sebelumnya yang dilakukan oleh (Ray & Bhat, 2013). Logaritma diskrit yang digunakan berdasarkan konsep algoritma RSA, jadi semua properti yang ada di logaritma diskrit tersebut sama dengan properti dari RSA. Tujuan utama menggabungkan knapsack dengan logaritma diskrit adalah untuk meningkatkan keamanan lingkungan kriptosistem dari algoritma merkle-hellman knapsack. Kekuatan keamanan dari logaritma diskrit berdasarkan konsep RSA adalah dari bilangan primanya. Bilangan prima tersebut dapat membentuk kunci publik dan kunci privat. Oleh karena itu, semakin besar bilangan prima yang diterapkan maka semakin kuat pula kunci privatnya (Akik Hidayat, 2007). Pada penelitian ini, kombinasi algoritma knapsack dan logaritma diskrit diterapkan pada chat.

Pengujian untuk membuktikan pesan chat tersebut sudah aman dari berbagai serangan dapat dilakukan serangkaian metode pengujian. Metode tersebut adalah *avalanche effect*, *known plaintext attack* dan performa algoritma. *Avalanche effect* merupakan metode untuk mengetahui berapa persen pengubahan pesan setelah proses enkripsi. Metode pengujian *known plaintext* merupakan pengujian untuk mengetahui pesan asli tanpa menggunakan kunci privat. Cara kerja pengujian ini dengan cara mengenkripsi semua karakter dengan kunci publik yang didapat dan mencocokkan hasil enkripsi tersebut dengan chiperteks yang didapat pula (Somani & Mangal, 2014). Performa kombinasi algoritma knapsack dan logaritma diskrit memanfaatkan waktu pembangkitan kunci, enkripsi dan dekripsi.

Ide utama dari penelitian ini adalah melakukan analisa dan implementasi kombinasi algoritma knapsack dan logaritma diskrit. Dilakukanya penggabungan algoritma ini diharapkan membuat lingkungan aplikasi chat lebih aman dikarenakan untuk memperoleh informasi dari aplikasi chat harus memecahkan kedua algoritma penggabungan itu. Hasil akhirnya akan dilakukan pengujian analisa ketahanan performa algoritma baik algoritma knapsack standard maupun algoritma knapsack dengan logaritma diskrit yang meliputi waktu proses pembangkitan kunci, enkripsi dan dekripsi dengan menggunakan metode *avalanche effect* dan *known plaintext attack*.

2. METODOLOGI PENELITIAN

2.1. Rancangan Aplikasi Chat



Gambar 1. Skema Aplikasi Chat

Algoritma merkle-hellman knapsack dan kombinasi algoritma knapsack dan logaritma diskrit diterapkan pada aplikasi chat pada jaringan localhost. Peran server dalam aplikasi chat tersebut adalah untuk menghubungkan klien agar dapat terhubung. Proses aplikasi chat adalah sebagai berikut:

a. Klien 1 mengisikan port yang sesuai dengan port server agar dapat terhubung. Hal tersebut juga berlaku terhadap klien 2.
b. Masing-masing klien membangkitkan kunci publik dan kunci privat. Kunci publik yang sudah dibangkitkan tersebut akan didistribusikan ke klien lain memalui server.
c. Klien 1 mengirim pesan yang terenkripsi ke klien 2 menggunakan kunci publik milik klien 2.
d. Klien 2 menerima pesan acak dari klien 1. Agar pesan tersebut dapat dibaca, pesan acak tersebut didekripsi menggunakan kunci privat klien 2.

2.2. Algoritma Merkle-Hellman Knapsack

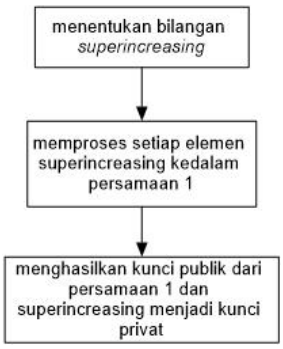
Algoritma merkle-hellman knapsack merupakan algoritma asimetris. Artinya, algoritma ini memliki kunci publik dan kunci privat untuk proses enkripsi dan dekripsinya. Algoritma ini mempunyai tiga proses mekanisme. Proses yang pertama adalah proses pembangkitan kunci publik dan kunci privat, proses enkripsi dan proses dekripsi. Sebelum melakukan ketiga proses tersebut, beberapa mekanisme dari algoritma ini antara lain:

a. Menentukan deretan superincreasing di mana setiap elemen di dalam deretan harus lebih besar daripada jumlah elemen sebelumnya. Deretan superincreasing ini digunakan sebagai kunci privat.
b. Memilih bilangan prima *m* dengan ketentuan bilangan tersebut harus lebih besar daripada elemen terakhir deretan superincreasing

c. Memilih bilangan *n*, di mana bilangan tersebut relatif prima terhadap bilangan *m*.
d. Membangkitkan kunci publik dengan persamaan 1:

$$pi = si * n \; mod \; m \quad (1)$$

Variabel i pada pada persamaan (1) merupakan indeks dari deretan superincreasing. Jadi, setiap elemen di dalam bilangan superincreasing diproses pada persamaan (1) dan menghasilkan kunci publik. Kunci publik tersebut memiliki panjang elemen yang sama dengan deretan superincreasing. Gambaran umum tentang pembangkitan kunci algoritma knapsack seperti pada Gambar 2.



Gambar 2.Pembangkitan Kunci Knapsack

e. Melakukan proses enkripsi dengan persamaan (2):

$$c = b_1 * p_1 + b_2 * p_2 + \ldots + b_n * p_n \quad (2)$$

Selain persamaan (2), adapun *pseudocode* untuk proses enkripsi adalah sebagai berikut:

```
INPUT : publickey[]
OUTPUT : chipertext
FOR(i=0;i<text.lenght; i++) DO
  ASCII=plaintext.charAt(i)
  biner= Int.toBinary(ASCII)
   c = 0
  FOR(j=0;j<biner.lenght;j++) DO
c=c+(biner.charAt[j].publickey[j
])
  ENDFOR
  chipertext=chipertext+c+" "
ENDFOR
```
Gambar 3. Pseudocode enkripsi knapsack

Variabel *b* pada persamaan (2) merupakan bentuk biner dari plainteks yang akan dienkripsi. Setiap indeks biner tersebut dikalikan dengan elemen kunci publik dengan indeks yang sama. Hasil tersebut ditambahkan dengan indeks selanjutnya. Proses tersebut dilakukan berulang sampai pada indeks terakhir.
f. Melakukan proses dekripsi dengan persamaan (3):

$$c * n^{-1} \; mod \; m \quad (3)$$

Selain dari persamaan (3) ada pula pseudocode untuk proses dekripsi algoritma knapsack seperti pada Gambar 4.

```
INPUT: m, n, chiprtxt
OUTPUT: plaintext
m_inv = n.modInvers(m)
String temp = ""
FOR(i=0;ciphrtxt.lenght;i++) DO
if((chiprtxt.charAt(i)+"")=" ")
do
 p = m_inv.mod(m)
 ASCII = Integer.parseInt
 plaintext = plaintext+ASCII
 temp = ""
 ELSE
 temp=temp+chipertext.charAt(i)
ENDIF
ENDFOR
```
Gambar 4. Pseudocode Dekripsi Knapsack

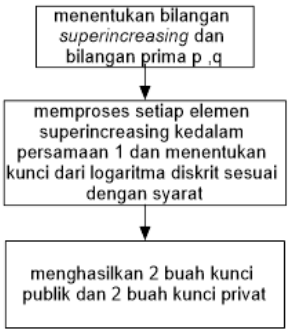
Nilai n^{-1} pada persamaan (3) merupakan nilai invers dari bilangan prima *m* yang sudah dibangkitkan. Penghitungan nilai invers ini menggunakan *extended euclidean*. Setelah dilakukannya proses dekripsi, proses selanjutnya adalah mengkonversi hasil tersebut ke dalam bentuk biner. Untuk mencari nilai binernya, hasil dekripsi dikurangi dengan elemen superincreasing yang besarnya mendekati hasil dekripsi tersebut. Proses tersebut berlanjut sampai hasil pengurangan menjadi 0. Diberikan nilai 1 pada indeks superincreasing apabila terpilih menjadi operasi pengurangan dan diberikan nilai 0 jika tidak terpilih. Hasil dari serangkaian proses tersebut menghasilkan nilai biner dan dikonversi kembali dalam bentuk karakter (Kortsarts & Kempner, 2010).

2.3. Kombinasi Algoritma Merkle-Hellman Knapsack dan Logaritma Diskrit

Mekanisme dari kombinasi algoritma ini hampir sama dengan algoritma knapsack. Hal yang berbeda terdapat pada dua kali proses pada setiap mekanisme karena pada kombinasi algoritma ini terdapat dua kunci publik dan dua kunci privat. Beberapa mekanisme dari kombinasi algoritma knapsack dan logaritma diskrit sebagai berikut:

a. Menentukan deretan superincreasing di mana setiap elemen didalam deretan harus lebih besar daripada jumlah elemen sebelumnya. Deretan superincreasing ini digunakan sebagai kunci privat pertama.
b. Memilih bilangan prima *m* dengan ketentuan bilangan tersebut harus lebih besar daripada elemen terakhir deretan superincreasing
c. Memilih bilangan *n*, di mana bilangan tersebut relatif prima terhadap bilangan *m*.

d. Membangkitkan kunci publik pertama dengan persamaan 1.
e. Memilih bilangan prima *p* dan *q*.
f. Menghitung nilai *n_rsa = p*q*.
g. Menghitung nilai $\phi (n) = (p-1)(q-1)$.
h. Memilih bilangan *e* dengan syarat $(e, \phi (n)=1)$. bilangan *e* ini digunakan sebagai kunci publik kedua.
i. Menghitung nilai d dengan syarat $d * e \bmod (\phi (n)) = 1$. Nilai d ini digunakan sebagai kunci privat kedua. Gambaran umum tentang pembangkitan kunci kombinasi algoritma ini dapat dilihat pada Gambar 5.



Gambar 5. Pembangkitan Kunci Kombinasi Algoritma

j. Melakukan proses enkripsi dengan persamaan (4):

$$c = m^e \bmod n\_rsa \quad (4)$$

Selain persamaan diatas, adapun pseucode pada proses enkripsi kombinasi algoritma knapsack dan logaritma diskrit seperti pada Gambar 6.

```
INPUT : e, n_rsa, kunci_publik[]
OUTPUT : chipertext
Str plaintext
FOR(;i<plaintext.lenght; i++) DO
 //enkripsi knapsack
 ASCII=plaintext.charAt(i)
 Str biner=toBinaryString(ASCII)
 c = 0
  FOR(j=0;j<biner.lenght;j++) DO
  c=
c+(biner.charAt[j].kuncipublik[j]
)
  ENDFOR
 //enkripsi RSA
 c = c.modpow(e, n_rsa)
 chipertext = chipertext+c+" "
ENDFOR
```
Gambar 6. Pseudocode Enkripsi Kombinasi Algoritma

Variabel m pada persamaan (4) merupakan hasil dari proses persamaan (2). Artinya, proses enkripsi dilakukan dua kali di persamaan (4).

k. Melakukan proses dekripsi dengan persamaan (5):

$$m = c^d \bmod n\_rsa \quad (5)$$

Selain persamaan diatas, adapun pseudocode proses dekripsi dari kombinasi algoritma seperti pada Gambar 7.

```
INPUT : chiper, d, n_rsa, m, n
OUTPUT : plain
m_inv = n.modInvers(m)
plain = chiper.modPow(d, n_rsa)
plaintext = konversi ke ASCII
FOR(i=0; i<chiper.length; i++) DO
 IF(chiper.charAt(i)+"" = " ") DO
 //dekripsi logaritma diskrit
 P = plaintext.modPow(d, n_rsa)
 //dekripsi knapsack
 p = m_inv.mod(m)
 integer ASCII = Integer.parseInt
 plaintext = plaintext+ASCII
 temp = ""
 ENDIF
ENDFOR
```
Gambar 7. Pseudocode Dekripsi Kombinasi Algoritma

Proses dekripsi pada kombinasi algoritma ini dilakukan dua kali. Proses yang pertama menggunakan persamaan (5). Hasil dari persamaan ini didekripsi lagi menggunakan persamaan (3).

2.4. *Avalanche Effect*

Avalanche Effect merupakan rasio antara jumlah bit chiperteks yang berubah akibat pengubahan plainteks ataupun terhadap jumlah bit total. Jika pengubahan bit adalah setengan dari jumlah bit chiperteks (50%) maka akan sulit dipecahkan. Pada pengujian ini akan dilakukan pengujian avalanche effect (pengubahan plainteks) pada proses enkripsi untuk mengetahui pengubahan bit-bit pada plainteks sehingga dapat diperoleh hasil persentase yang menentukan baik atau tidaknya algoritma Merkle-Hellman Knapsack yang digunakan (A Hidayat & Akmal, 2016). Rumus *avalanche effect* tersebut dapat dilihat pada persamaan (6):

$$Avalanche\ Effect = \frac{Jumlah\ bit\ yang\ berubah}{jumlah\ bit\ total}\ x\ 100\% \quad (6)$$

2.5. *Known Plaintext Attack*

Known plaintext attack merupakan jenis serangan yang digunakan untuk menemukan plaintext *(p)* dari chipertext *(c)* tanpa menggunakan kunci privat dari suatu algoritma kriptografi. Pada metode serangan ini, terdapat data yang harus dipenuhi yaitu chipertext *(c)* dan plaintext *(p)*. Cara untuk menemukan teks asli adalah dengan mencocokkan nilai *(p)* dan *(c)*. Sebagai skenario penyerangan terdapat tiga user yaitu user A, user B dan user C. pada simulasinya user A memiliki kunci publik (23,391) yang dibagikan pada user B.

Kemudian user B mengenkripsi sebuah pesan dengan kunci publik yang dibagikan tadi. User C yang sebagai penyadap mencoba mengenkripsi himpunan huruf (a-z) yang sebelumnya sudah diubah ke bilangan desimal dan memberikan hasil seperti pada tabel.

Tabel 1. Hasil Enkripsi Himpunan a-z

K	P	C	K	P	C	K	P	C
a	97	143	j	106	336	s	115	276
b	98	259	k	107	61	t	116	346
c	99	329	l	108	269	u	117	25
d	100	8	m	109	63	v	118	118
e	101	101	n	110	202	w	119	119
f	102	102	o	111	19	x	120	126
g	103	103	p	112	153	y	121	213
h	104	196	q	113	343	z	122	283
i	105	266	r	114	160			

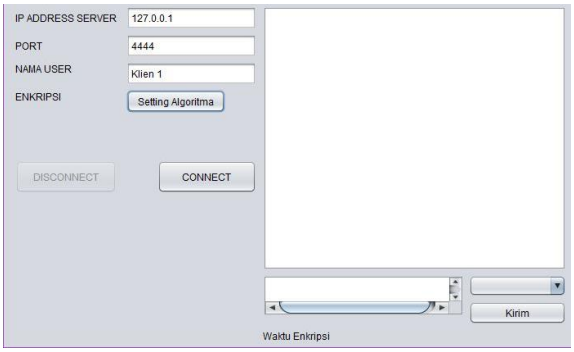
User B mengenkripsi dengan kunci publik yang dibagikan oleh User A yang menghasilkan nilai chiperteks 266 346. Disisi lain user C berhasil mendapatkan nilai chiperteks yang dienkripsi oleh user B yaitu 266 346. Dari hasil tersebut user C mencocokkan nilai chiperteks yang berkoresponden terhadap karakter. Dalam pencocokkannya, user C mendapatkan nilai yang berkoresponden yaitu C=266=105, C=346=116. Dari hasil tersebut user C mendapatkan plainteks P = 105 116 yang kemudian diubah ke karakter menjadi "it". Dari skenario diatas dapat disimpulkan bahwa user C hanya mendapatkan kunci publik dan chiperteks untuk mengetahui karakter asli dari chiperteks dan tidak perlu menggunakan kunci privat.

3. IMPLEMENTASI DAN PENGUJIAN

Implementasi dan pengujian yang dilakukan untuk mengetahui perbedaan kedua algoritma dengan memanfaatkan waktu pembangkitan kunci, waktu enkripsi dan waktu dekripsi. Ketiga proses tersebut diuji dengan panjang bit bilangan prima yang berbeda dan diuji beberapa kali agar mendapatkan nilai rata-rata. Pengujian keamanan kedua algoritma ini dilakukan dengan metode *avalanche effect* dan *known plainteks attack*. Hasil dari pengujian akan dibandingkan agar mengetahui performa dari masing-masing algoritma

3.1. Implementasi Aplikasi Chat

Aplikasi chat dibuat dengan bahasa pemrograman Java dikarenakan pemrograman Java mendukung semua platform (Aminudin, Azhari, & Ahmad, 2018) dan berjalan pada jaringan lokal. Pada aplikasi ini pesan yang terkirim akan dienkripsi dengan pilihan algoritma knapsack atau kombinasi knapsack dan logaritma diskrit. Masing-masing dari algoritma tersebut dapat diatur panjang bit bilangan prima meliput 16, 32, 64, 128, 256, 512 dan 1024 bit.



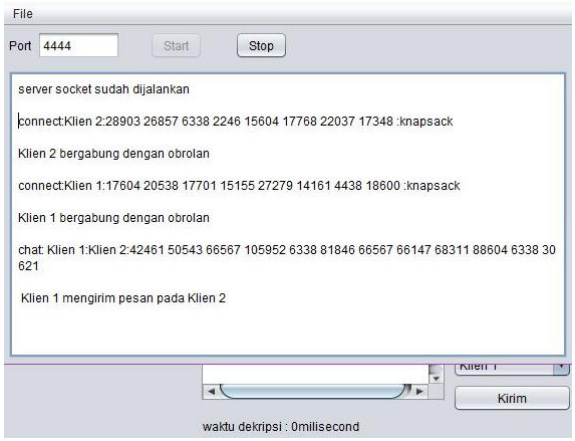
Gambar 8. Tampilan Klien

Klien yang akan terhubung ke server harus mengisikan alamat *IP server*, *port*, nama user dan pengaturan algoritma. Alamat *IP server* diisikan 127.0.0.1 karena alamat *IP* ini menunjukkan jaringan localhost sedangkan port yang diisikan harus sesuai dengan port server. Klien dapat mengatur algoritma yang akan digunakan untuk mengirim pesan. Pengaturan algoritma dapat dibuka pada tombol "Setting Algoritma". Terdapat dua pilihan algoritma yang dapat digunakan. Masing-masing algoritma mempunyai pilihan panjang bit seperti pada Gambar 4. Contoh Klien 1 mengirim pesan kepada Klien 2 terlihat pada Gambar 9.

Gambar 9. Tampilan Klien 1 Mengirim Pesan

Gambar 10. Klien 2 Menerima Pesan

Terlihat pada Gambar 9 klien 1 mengirim pesan kepada klien 2. Pesan tersebut sudah dienkripsi dengan menggunakan salah satu pilihan algoritma. Pesan yang sudah dienkripsi tersebut dapat dilihat melalui server seperti pada Gambar 10.



Gambar 11. Log Server

Terlihat pada Gambar 11 terdapat log pendistribusian kunci publik dan percakapan antar klien. Pendistribusian kunci publik dapat dilihat pada label ”connect:Klien 1” artinya adalah Klien 1 memiliki kunci publik 17604 20538 17701 15155 27279 14161 4438 18600. Begitu pula terhadap Klien 2. Percakapan antar klien juga dapat dilihat pada label ”chat Klien 1:Klien2:” artinya klien 1 mengirimkan pesan kepada klien 2 tetapi pesan yang ada di log berupa angka-angka yang menunjukkan bahwa pesan tersebut telah dienkripsi.

3.2. Pengujian Waktu Pembangkitan Kunci

Analisa perbandingan hasil pengujian waktu pembangktian kunci dilakukan dengan membandingkan rata-rata waktu pembangkitan kunci pada tiap panjang bit bilangan prima. Perbandingan ini dilakukan untuk menganalisa performa waktu pembangkitan kunci antara algoritma knapsack dan gabungan algoritma knapsack dengan logaritma diskrit. Perbandingan rata-rata waktu pembangkitan kunci dengan parameter panjang 256, 512 dan 1024 bit.

Tabel 2. Perbandingan Waktu Pembangkitan Kunci

Bit	Rata-Rata (ms)	
	Knapsack	Knapsack+Logaritma Diskrit
256 bit	83 ms	1552,33 ms
512 bit	343,66 ms	12656,33 ms
1024 bit	2880 ms	153918,66 ms

Hasil dari Tabel 2 menunjukkan bahwa waktu pembangkitan kunci algoritma knapsack lebih cepat daripada gabungan algoritma knapsack dengan logaritma diskrit. Beberapa faktor yang mempengaruhi waktu pembangkitan kunci adalah kecepatan sistem dalam membangkitkan bilangan prima dan besarnya bit yang digunakan. Selain itu, faktor lainnya mempengaruhi waktu pembangkitan kunci adalah jumlah kunci publik dan kunci privat yang dibangkitkan. Algoritma knapsack mempunyai satu kunci publik sedangkan gabungan algoritma knapsack dan logaritma diskrit mempunyai dua buah kunci publik, sehingga pada gabungan algoritma ini waktu pembangkitan kunci lebih lama daripada algoritma knapsack.

Proses untuk membangkitkan bilangan prima tersebut menggunakan algoritma miller rabin. Mula-mula sistem membangkitkan bilangan ganjil secara acak dengan panjang bit yang sudah ditentukan, kemudian dilakukan serangkaian pengujian untuk menentukan status keprimaan bilangan. Jika uji keprimaan algoritma miller rabin bernilai true maka proses pembangkitan bilangan prima selesai, sebaliknya jika uji keprimaan algoritma miller rabin bernilai false maka sistem mengulang proses pembangkitan bilangan prima dari awal. Besarnya bilangan bit yang digunakan juga mempengaruhi waktu pembangkitan kunci, karena semakin besar parameter bit yang digunakan maka semakin lama juga proses uji keprimaan algoritma miller rabin.

3.3. Pengujian Waktu Enkripsi

Pengujian waktu enkripsi pada algoritma knapsack maupun kombinasi algortima knapsack dilakukan beberapa kali pengujian dengan pengaturan panjang bit bilangan prima yang berbeda agar dapat nilai rata-rata. Panjang bit yang digunakan dalam pengujian enkripsi adalah 256, 512 dan 1024 bit sedangkan panjang karakter panjang karakter yang diuji ialah 100, 250 dan 400 karakter. Masing-masing panjang karakter diuji dengan panjang bit bilangan prima dan diuji beberapa kali agar mendapatkan nilai rata-rata. Satuan waktu yang digunakan dalam pengujian waktu enkripsi menggunakan milisecond (ms).

Tabel 3. Perbandingan Waktu Enkripsi

Panjang Karakter	Bit	Rata-rata (ms)	
		Knapsack	Knapsack+Logaritma Diskrit
100	256	15,66 ms	311,33 ms
	512	16 ms	2796,66 ms
	1024	26 ms	20896 ms
250	256	31.66 ms	1041,66 ms
	512	47 ms	7052,33 ms
	1024	109 ms	52016,66 ms
400	256	62,33 ms	1682,33 ms
	512	109,66 ms	11463,33 ms
	1024	246 ms	85678,33 ms

Hasil dari Tabel 3 menunjukkan bahwa perbedaan waktu rata-rata dari masing-masing algoritma yang diuji terlihat cukup jauh. Perbedaan waktu ini dikarenakan pada algoritma knapsack proses enkripsi membutuhkan satu kunci publik sedangkan untuk gabungan algoritma knapsack dan logaritma diskrit membutuhkan dua kunci publik sehingga waktu rata-rata gabungan algoritma lebih lama daripada algoritma knapsack. Faktor lainnya yang mempengaruhi waktu enkripsi adalah terdapat perbedaan proses enkripsi dari masing-masing algoritma. Proses enkripsi algoritma knapsack dilakukan satu kali, sedangkan gabungan algoritma knapsack dan logaritma diskrit dilakukan sebanyak dua kali yang menyebabkan proses enkripsi gabungan algoritma membutuhkan waktu yang lama. Selain perbedaan proses tersebut, waktu yang

mempengaruhi proses enkripsi adalah panjang bit bilangan prima dan panjang karakter yang diuji. Terlihat pada Tabel 3 semakin besar panjang bit dan karakter yang digunakan, hasil waktu enkripsi juga mengalami kenaikan.

3.4. Pengujian Waktu Dekripsi

Analisa perbandingan hasil pengujian waktu dekripsi dilakukan dengan membandingkan rata-rata waktu dekripsi pada tiap panjang karakter dan panjang bit bilangan prima. Perbandingan ini dilakukan untuk menganalisa performa waktu dekripsi antara algoritma knapsack dan gabungan algoritma knapsack dengan logaritma diskrit. Perbandingan rata-rata waktu enkripsi dengan parameter panjang 256, 512 dan 1024 bit dan panjang karakter 100, 250, dan 400 karakter.

Tabel 4. Perbandingan Waktu Dekripsi

Panjang Karakter	Bit	Rata-rata (ms)	
		Knapsack	Knapsack+Logaritma Diskrit
100	256	31 ms	823 ms
	512	51,66 ms	5437,33 ms
	1024	114,33 ms	34182,66 ms
250	256	31,33 ms	1979 ms
	512	62,33 ms	13573,33 ms
	1024	177,33 ms	64656,66 ms
400	256	46,66 ms	3177,33 ms
	512	83,33 ms	31828,33 ms
	1024	265,66 ms	162340,66 ms

Hasil dari Tabel 4 menunjukkan bahwa perbandingan waktu rata-rata dekripsi algoritma knapsack lebih cepat dibandingkan gabungan algoritma knapsack dengan logaritma diskrit. Beberapa faktor yang mempengaruhi waktu dekripsi adalah rumus dekripsi dari masing-masing algoritma, panjang bit bilangan prima dan panjang karakter. Proses dekripsi pada algoritma knapsack memiliki satu langkah proses dekripsi dan membutuhkan satu kunci privat dalam proses dekripsinya. Hal yang berbeda terjadi pada gabungan algoritma knapsack dengan logaritma diskrit yang mempunyai dua langkah dekripsi dan membutuhkan dua buah kunci privat sehingga waktu dekripsi dari gabungan algoritma ini cukup lama jika dibandingkan dengan algoritma knapsack. Selain perbedaan proses dekripsi tersebut, panjang bit bilangan prima dan panjang karakter juga mempengaruhi waktu dekripsi. Panjang bit bilangan prima ini merupakan faktor pembentuk kunci privat. Jika pengujian menggunakan panjang bit bilangan prima yang besar maka kunci privat yang dihasilkan juga semakin besar. Hal tersebut akan mempengaruhi waktu dekripsi karena sistem akan menghitung ciphertext menggunakan kunci privat yang besar. Untuk mempercepat pemrosesan dari parameter yang digunakan dapat juga menggunakan konsep multithreading dengan menggunakan salah satu framework diantaranya Apache Mahout, Apache Spark atau Apache Hadoop (Aminudin & Alwi, 2018).

3.5. Pengujian *Avalanche Effect*

Avalanche effect merupakan metode pengujian untuk mengetahui seberapa besar pengubahan chipertext yang dihasilkan oleh suatu kunci publik. Jika pengubahan chipertext lebih dari 50% maka dapat dikatakan algoritma tersebut aman. Pada sistem ini akan dilakukan pengujian avalanche effect (pengubahan *plaintext*) pada proses enkripsi untuk mengetahui pengubahan bit-bit pada plaintext sehingga dapat diperoleh hasil persentase yang dapat menentukan baik atau tidaknya algoritma Merkle-Hellman Knapsack maupun gabungan knapsack dan logaritma diskrit yang digunakan. Hasil dari pengujian avalanche effect pada masing-masing algoritma dapat dilihat pada Tabel 5.

Tabel 5. Perbandingan Hasil *Avalanche Effect*

Teks	Cipherteks		Persentase	
	Knapsack	Knapsack+LD	Knapsack	Knapsack+LD
IT IG	340 281 340 408	1172 314 1172 1276	9 %	27,2 %
UMM UMY	336 436 436 336 436 485	708 469 469 708 469 1320	9%	24,24%
Inform	340 658 403 713 452 491	1172 748 124 398 1528 1468	6%	7,56%
Infarm	340 658 403 135 452 491	1172 748 124 1501 1528 1468		

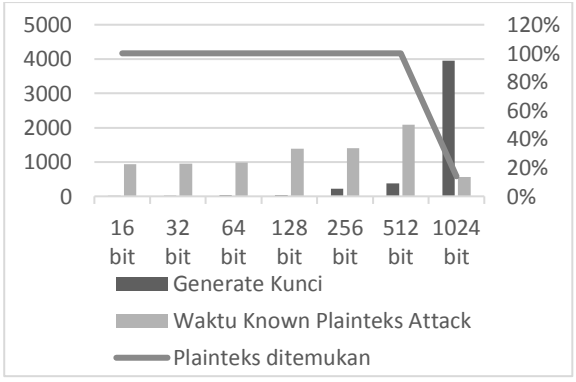
Dijelaskan pada Tabel 5 terdapat *ciphertext* dari masing-masing algoritma yang dibandingkan. *Ciphertext* tersebut merupakan hasil dari proses enkripsi dengan menggunakan kunci publik dari masing-masing algoritma yang sudah ditentukan. Nilai kunci publik yang digunakan oleh kedua algoritma tersebut sama tetapi pada gabungan algoritma knapsack dan logaritma diskrit ada tambahan kunci publik.

Nilai persentase gabungan algoritma knapsack dengan logaritma diskrit pada Tabel 5 mempunyai nilai yang lebih besar daripada nilai persentase algoritma knapsack. Besarnya nilai persentase dari gabungan algoritma ini dipengaruhi oleh proses enkripsi dari gabungan algoritma tersebut. Proses enkripsi dari gabungan algoritma knapsack dan logaritma diskrit dilakukan dua kali sehingga perbedaan ciphertext yang dihasilkan terlihat cukup besar. *Ciphertext* dari kedua *plaintext* tersebut diubah kedalam bentuk biner dan dibandingkan bit-bit yang berbeda, jika nilai bit-bit yang berbeda cukup banyak maka nilai persentase juga bernilai besar.

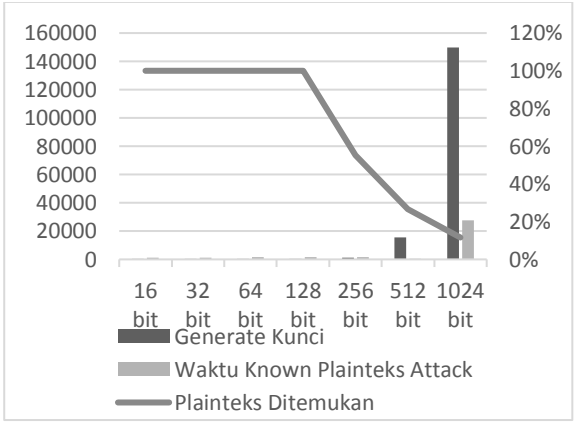
3.6. Pengujian *Known Plaintext Attack*

Pengujian *known plaintext attack* pada gabungan algoritma knapsack dan logaritma diskrit dilakukan dengan memanfaatkan informasi dari server. Adapun skenario pengujian tersebut adalah penyerang telah mendapatkan ciphertext dan memiliki kunci publik dari hasil sniffing menggunakan aplikasi Rawcap. Pengujian ini dilakukan dengan panjang karakter yang berbeda dan panjang bit yang berbeda pula. Panjang bit yang digunakan adalah 16, 32, 64, 128, 256, 512 dan 1024 bit. Panjang bit yang dimaksud adalah panjang bit bilangan prima *(m)* dan bilangan prima *(p, q)* yang merupakan faktor pembentuk dari kunci publik. Satuan waktu dalam menghitung berapa lama proses metode *known plaintext attack* menggunakan *millisecond (ms)*.

Analisa perbandingan known plaintext dilakukan dengan membandingkan waktu pembangkitan kunci, waktu eksekusi program *known plaintext* dan persentase plainteks yang ditemukan dari masing-masing algoritma yang diuji. Pembangkitan kunci ini digunakan untuk membangkitkan kunci publik dan kunci privat. Kunci publik tersebut akan digunakan pada proses enkripsi. Perbandingan ini dilakukan untuk menganalisa performa kedua algoritma terhadap *known plaintext attack*. Hasil perbandingan kedua algoritma dapat dilihat pada Gambar 12 dan 13.



Gambar 12. Performa Algoritma Knapsack Terhadap Known Plaintext Attack



Gambar 13. Performa Kombinasi Algoritma Terhadap Known Plaintext Attack

Jika dilakukan pengamatan dari hasil pembangkitan kunci pada masing-masing bit, algoritma knapsack memiliki waktu yang cepat dibandingkan dengan gabungan algoritma knapsack dan logaritma diskrit. Perbedaan waktu tersebut dipengaruhi oleh banyaknya bilangan prima yang dibangkitkan oleh algoritma miller rabin. Jumlah bilangan prima yang dibangkitkan dari algoritma knapsack adalah satu bilangan prima *(m)*, sedangkan untuk gabungan algoritma knapsack dan logaritma diskrit adalah tiga bilangan prima *(m, p dan q)*, sehingga waktu yang dibutuhkan untuk membangkitkan kunci juga semakin lama.

Proses waktu eksekusi program *known plaintext* dari masing-masing bit juga berbeda. Waktu eksekusi program dari algoritma knapsack lebih cepat daripada gabungan algoritma knapsack dan logaritma diskrit. Waktu eksekusi program tersebut dipengaruhi oleh proses enkripsi dari masing-masing algoritma dan proses mencocokkan antar karakter. Proses enkripsi pada algoritma knapsack dilakukan satu kali sedangkan pada gabungan algoritma knapsack dan logaritma diskrit dilakukan sebanyak dua kali, sehingga waktu eksekusi program pada gabungan algoritma knapsack dan logaritma diskrit lebih lama daripada algoritma knapsack.

Plaintext yang ditemukan pada algoritma knapsack maupun logaritma diskrit tidak semuanya ditemukan secara utuh. Terlihat pada Gambar 12 plainteks yang tidak ditemukan secara utuh berada pada bit 1024, sedangkan pada Gambar 13 plainteks yang tidak utuh berada pada bit 256, 512 dan 1024. Tidak ditemukannya plainteks tersebut secara utuh dikarenakan ada paket-paket yang tidak terekam pada proses sniffing. Paket-paket yang tidak terekam disebabkan oleh bit pembentuk kunci publik yang digunakan besar, sehingga ciphertext yang dihasilkan juga semakin besar. Hal ini mempengaruhi proses sniffing, semakin besar bit yang digunakan maka semakin banyak pula paket-paket yang direkam. Faktor lainnya yang mempengaruhi tidak terekamnya paket adalah perangkat yang digunakan, karena pada proses sniffing dibutuhkan waktu yang lama sehingga jika perangkat yang digunakan mati maka paket-paket yang direkam juga tidak lengkap.

4. KESIMPULAN

Berdasarkan implementasi dan pengujian diatas, maka dapat disimpulkan bahwa:

a. Implementasi algoritma knapsack dan kombinasi knapsack dan logaritma diskrit dapat diaplikasikan pada aplikasi chat. Aplikasi chat tersebut dapat melindungi pesan agar pesan tersebut lebih aman dan tidak mudah dibaca oleh orang yang tidak berhak karena pesan tersebut ketika dikirim berbentuk ciphertext yang sudah dienkripsi menggunakan kunci publik dan dapat

didekripsi oleh orang yang memiliki kunci privat.

b. Pengujian *known plaintext attack* membuktikan bahwa jika kedua algoritma ini masih rentan terhadap serangan ini dengan pembuktian nilai *plaintext* ditemukan 100% pada data uji lampiran 3. Tetapi tidak semuanya plainteks ditemukan 100% pada tiap bit pengujian. Terbukti bahwa penggunaan bit 1024 pada algoritma knapsack ditemukan 14% sedangkan penggunaan bit 256, 512 dan 1024 pada gabungan algoritma ditemukan plainteks sebesar 55,10%, 26,70% dan 11,60 %.
c. Pengujian *avalanche effect* membuktikan bahwa kombinasi knapsack dan logaritma diskrit memiliki nilai persentase yang besar dibandingkan algoritma knapsack.
d. Algortima knapsack memiliki performa yang lebih cepat dibandingkan kombinasi knapsack dan logaritma diskrit. Hal tersebut dikarenakan pada kombinasi knapsack proses pembangkitan kunci, enkripsi dan dekripsi dilakukan dua kali proses sehingga waktu eksekusi juga ikut berpengaruh.
e. Untuk penelitian selanjutnya dapat menggunakan konsep multithreading agar pemrosesan pembangkitan kunci, waktu enkripsi dan waktu dekripsi lebih cepat.

5. DAFTAR PUSTAKA

AMINUDIN, A., & ALWI, M. (2018). Analisa Multithreading Pada Sistem Rekomendasi Menggunakan Metode Collaborative Filtering Dengan. *Techno. Com*, *17*(1), 1–11.

AMINUDIN, A., AZHARI, S. N., & AHMAD, B. (2018). Automatic Question Generation (AQG) dari Dokumen Teks Bahasa Indonesia Berdasarkan Non-Factoid Quesion. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *5*(2), 217–224.

CHOUHAN, K., & RAVI, S. (2013). Public key encryption techniques provide extreme secure chat environment.

FADLAN, M., & HADRIANSA, H. (2017). Rekayasa Aplikasi Kriptografi dengan Penerapan Kombinasi Algoritma Knapsack Merkle Hellman dan Affine Cipher. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *4*(4), 268–274.

HIDAYAT, A. (2007). Kriptosistem Knapsack. *Abstrak*.

HIDAYAT, A., & AKMAL, R. (2016). Cryptography Asymmetries Merkle-Hellman Knapsack Digunakan untuk Enkripsi dan Dekripsi Teks. *Riset.Fmipa.Unpad.Ac.Id*, 27–28. Retrieved from http://riset.fmipa.unpad.ac.id/data/uploads/paper/semnas/2016/014.-066-068-akik-hidayat.pdf

KORTSARTS, Y., & KEMPNER, Y. (2010). *Merkle-Hellman Knapsack Cryptosystem in Undergraduate Computer Science Curriculum*.

RAY, A., & BHAT, S. (2013). Enhancement of Merkle-Hellman knapsack cryptosystem by use of discrete logarithmics. *Int J Sci Res Publ*, *3*(4), 230.

SHAMIR, A. (1984). A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem. *IEEE Transactions on Information Theory*, *30*(5), 699–704. https://doi.org/10.1109/TIT.1984.1056964

SOMANI, N., & MANGAL, D. (2014). An improved RSA cryptographic system. *International Journal of Computer Applications*, *105*(16).

Halaman ini sengaja dikosongkan