Komparasi Performa Aplikasi RC4 dan RC4+ Modifikasi untuk Pengamanan File Teks 50 Kata

Naufal Yahya Kurnianto - 13519141
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519141@std.stei.itb.ac.id

***Abstract*—Pada masa-masa maraknya terjadi globalisasi saat ini, persebaran informasi menjadi lebih mudah untuk dilakukan. Dengan adanya kemudahan tersebut, tentu juga ada sisi buruk di baliknya yaitu semakin mudah juga suatu informasi yang tidak seharusnya bocor malah bocor ke pihak-pihak yang tidak terkait dan bahkan disalah gunakan. Oleh karena itu, ilmu kriptografi yang bertujuan untuk menyembunyikan isi pesan ataupun informasi juga terus berkembang. Algoritma RC4 merupakan suatu algoritma *cipher* alir yang walaupun terbukti keamanannya sudah lumayan berumur. Hal ini menyebabkan munculnya berbagai ide yang salah satunya adalah algoritma RC4+ sebagai salah satu bentuk modifikasi dari algoritma RC4 original. Walaupun RC4+ sebagai varian baru terbilang lebih aman, belum tentu RC4+ lebih efisien untuk digunakan dibandingkan algoritma RC4. Makalah ini akan membandingkan performa dari RC4 dan RC4+ serta mencoba memodifikasi lebih jauh RC4+ terkait kemungkinan peningkatan efisiensi dari algoritma RC4+ itu sendiri.**

Keywords—Kriptografi; Algoritma RC4; Algoritma RC4+; Modifikasi

I. PENDAHULUAN

Zaman sekarang, saat dimana sudah marak sekali terjadinya globalisasi, sangat mudah munculnya kemungkinan bahwa akan ada suatu kebocoran dalam persebaran informasi. Kebocoran tersebut dapat terjadi karena setiap orang normalnya sudah memiliki perangkatnya masing-masing sehingga ada sekelompok orang yang memiliki pemikiran bahwa tiap-tiap masyarakat awam merupakan sasaran empuk untuk pembocoran informasi pribadi. Maka dari itu, terdapat suatu cabang keilmuan yang ada untuk menangkal terjadinya kebocoran informasi yaitu kriptografi.

Terdapat beberapa teknik untuk menyembunyikan suatu pesan sehingga tidak akan terjadi kebocoran. Salah satunya adalah menggunakan *cipher* alir. Salah satu aplikasi *cipher* alir adalah dengan menggunakan algoritma RC4. Algoritma ini dapat digunakan untuk menyembunyikan isi suatu pesan sehingga menjadi bentuk yang tidak dapat dicerna oleh orang awam. Namun demikian, bukan berarti algoritma ini bebas dari berbagai kemungkinan terjadinya suatu penyerangan sehingga isi pesan yang telah disembunyikan tidak bocor. Oleh karena itu, tercetus berbagai ide untuk memperumit atau membuat algoritma RC4 semakin kompleks. Ide-ide tersebutlah yang akhirnya merealisasikan terbentuknya RC4+. RC4+ memiliki berbagai perbedaan mendasar dengan RC4 yang akan dijelaskan kemudian.

RC4+ sebagai bentuk modifikasi dari algoritma RC4 yang original tentu dibentuk dengan tujuan mengurangi kekurangan yang dimiliki oleh RC4. Namun demikian, bukan berarti performa penggunaan RC4+ akan terasa lebih *appealing* bagi pengguna. Sangat mungkin efisiensi atau kecepatan enkripsi maupun dekripsi yang dilakukan oleh algoritma RC4+ sangat lambat dibandingkan algoritma RC4. Maka dari itu, dilakukanlah sebuah pengujian melalui pengamanan sebuah file teks .txt berisi teks dengan jumlah total kata sebanyak 50.

II. LANDASAN TEORI

A. Kriptografi

Menurut Menez, kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi [1]. Selain itu, secara umum kriptografi adalah suatu teknik pengamanan informasi dimana cara pengamanannya adalah dengan penggunaan suatu kunci tertentu melalui proses enkripsi sehingga informasi awalnya menjadi suatu informasi baru sehingga orang yang tidak berhak untuk melihat informasi tersebut tidak bisa melihatnya. Informasi bentuk baru tersebut kemudian bisa dikembalikan menjadi informasi awal melalui kunci yang dimiliki oleh orang yang berhak untuk melihat informasi tersebut. Berikut merupakan skema dari kriptografi secara simpel.

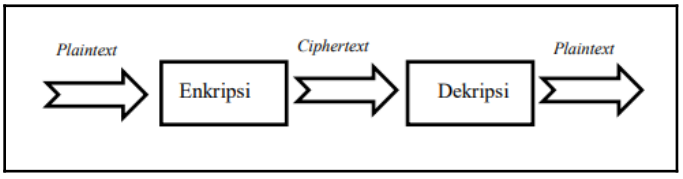

Fig. 1. Skema Simpel Kriptografi

Berbagai komponen dari kriptografi maupun berbagai terminologi dari kriptografi adalah sebagai berikut:

1. Pesan

   Pesan merupakan suatu data atau bentuk informasi yang dapat dimengerti maknanya. Pada skema sebelumnya, pesan ini merupakan *plaintext*.

2. Pengirim

   Pengirim merupakan suatu entitas yang berperan untuk mengirim sebuah pesan.

3. Penerima

   Penerima merupakan suatu entitas yang berperan untuk menerima sebuah pesan.

4. *Ciphertext*

   Ciphertext merupakan suatu data atau bentuk informasi yang tidak dapat dimengerti secara langsung makna dari isinya. *Ciphertext* ini merupakan sebutan untuk pesan yang sudah disembunyikan.

5. Enkripsi

   Enkripsi merupakan suatu proses untuk mengubah *plaintext* yang merupakan pesan asli menjadi sebuah *ciphertext*.

6. Dekripsi

   Dekripsi merupakan suatu proses untuk mengubah *ciphertext* yang merupakan pesan yang sudah disembunyikan menjadi sebuah *plaintext*.

7. *Cipher*

   Cipher merupakan sebuah fungsi matematika ataupun sebuah algoritma yang digunakan dalam proses enkripsi dan dekripsi.

8. *Key* (Kunci)

   Kunci merupakan sebuah faktor X atau parameter yang digunakan dalam suatu algoritma *cipher*.

9. Sistem Kriptografi

   Sistem ini merupakan gabungan dari berbagai komponen kriptografi yang harus ada di antaranya algoritma *cipher*, *plaintext* atau *ciphertext*, dan kunci.

10. Penyadap

    Penyadap merupakan sebutan bagi orang ketiga yaitu seseorang yang berniat untuk menangkap pesan yang sedang dalam masa transmisi antara pengirim dan penerima.

11. Kriptanalisis

    Kriptanalisis merupakan sebuah teknik untuk mendekripsi *ciphertext* menjadi *plaintext* tanpa mengetahui kunci yang digunakan untuk enkripsi.

Selain itu, kriptografi juga memiliki beberapa tujuan diantaranya:

1. Kerahasiaan (*confidentiality*)

   Kerahasiaan adalah layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak. Melalui kriptografi, pesan asli akan diubah menjadi bentuk *ciphertext* sehingga pesan tidak dapat dibaca oleh pihak yang tidak berhak.

2. Integritas (*Integrity*)

   Integritas adalah layanan yang menjamin bahwa pesan masih asli/utuh atau belum pernah dimanipulasi selama pengiriman. Sistem kriptografi juga harapannya dapat mengetahui apakah terjadi suatu kejadian penyisipan pesan lain dalam pesan awal sehingga pesan awal dapat dipastikan tetap utuh.

3. Otentikasi (*Authentication*)

   Otentikasi adalah layanan yang bertujuan untuk identifikasi kebenaran dari pihak-pihak yang berkomunikasi maupun mengidentifikasi kebenaran sumber pesan.

4. Pengirim Pesan tidak dapat menyangkal pengiriman pesan (*non-repudiation*)

   Merupakan suatu layanan untuk menghindari adanya penyangkalan dari pengirim pesan bahwa pihak terkait telah mengirim pesan.

B. Algoritma Simetris dan Asimetris

Algoritma kriptografi simetris merupakan teknik penggunaan kunci rahasia yang sama untuk proses enkripsi dan dekripsi. Berikut merupakan ilustrasi dari skema penggunaan algoritma kriptografi kunci simetris.

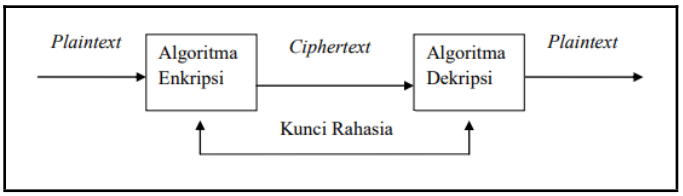

Fig. 2. Skema Simpel Penggunaan Algoritma Kriptografi Kunci Simetris

Berbagai algoritma kriptografi kunci simetris diantaranya adalah DES, AES, RC4, dan RC5, serta masih banyak lagi. Selain itu, algoritma kriptografi asimetris merupakan teknik yang menggunakan kunci berbeda untuk proses enkripsi dan dekripsi. Algoritma ini juga dapat disebut algoritma kunci publik karena dapat proses enkripsi dilakukan menggunakan kunci yang bersifat publik sedangkan proses dekripsi menggunakan kunci yang bersifat privat (hanya dimiliki penerima). Berikut merupakan ilustrasi dari skema penggunaan algoritma kriptografi kunci asimetris.

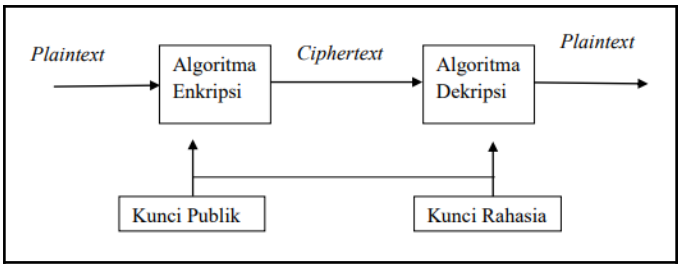

Fig. 3. Skema Simpel Penggunaan Algoritma Kriptografi Kunci Asimetris

C. *Algoritma RC4*

Algoritma RC4 merupakan salah satu algoritma dari bentuk *cipher* alir. *Cipher* alir sendiri merupakan suatu teknik *cipher* yang mengenkripsi plainteks menjadi *ciphertext* setiap bit per bit dengan bit-bit kunci (dinamakan bit keystream) atau byte per byte (1 byte setiap kali transformasi)[2]. Berikut merupakan ilustrasi dari diagram *cipher* alir.

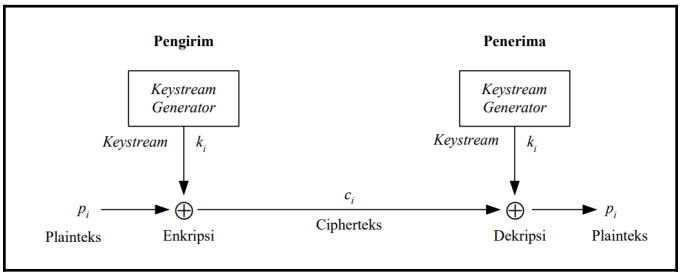

Fig. 4. Diagram Algoritma *Cipher* Alir

Algoritma RC4 sendiri dibuat oleh *Ron Rivest* (1987) dari laboratorium RSA. RC4 ini biasanya digunakan untuk protokol SSL (Secure Socket Layer); WEP (Wired Equivalent Privacy); serta WPA (Wi-fi Protect Access) untuk nirkabel. Ide dari algoritma ini adalah pembangkitkan kunci-alir (keystream) dalam ukuran byte setiap kalinya, yang kemudian di-XOR-kan dengan karakter plainteks[2]. Algoritma RC4 memiliki dua sub-proses yaitu Key-Scheduling Algorithm (KSA) dan Pseudo Random Generation Algorithm (PRGA). Sub-proses KSA adalah pembentukan larik yang akan digunakan sebagai bentuk kunci berjumlah 256 elemen. Kemudian, PRGA akan membangkitkan kunci-alir (keystream) dengan cara mengambil nilai dari larik hasil KSA dalam bentuk S[i] dan S[j], mempertukarkan nya, lalu menjumlahkan keduanya dalam modulus 256. Hasilnya pun kemudian akan di-XOR kan dengan sebuah karakter plainteks. Berikut merupakan ilustrasi implementasi algoritma RC4.

```
def rc4(plaintext,key):
    # KSA
    S = [i for i in range(256)]
    j = 0
    for i in range(256):
        j = (j + S[i] + ord(key[i % len(key)])) % 256
        S[i], S[j] = S[j], S[i]
    # PRGA
    i,j = 0,0
    result = ""
    for k in range(len(plaintext)):
        i = (i+1) % 256
        j = (j+S[i]) % 256
        S[i], S[j] = S[j], S[i]
        t = (S[i]+S[j]) % 256
        u = S[t]
        result += chr(u ^ ord(plaintext[k]))
    return result
```

Fig. 5. Implementasi Algoritma RC4

Dari sisi keamanan, RC4 yang merupakan suatu bentuk dari *cipher* alir membuatnya tidak kuat terhadap serangan seperti *flip-bit attack* maupun serangan-serangan *stream attack* lainnya. Maka dari itu, tercetus berbagai ide untuk memperkuat RC4 dengan penambahan berbagai algoritma sehingga terbentuklah RC4+ atau bisa dibilang RC4 modifikasi.

D. *Algoritma RC4+*

Algoritma RC4+ merupakan suatu jenis dari algoritma RC4. RC4 + adalah algoritma kunci simetris yang berbentuk synchronous stream cipher, yaitu cipher yang memiliki kunci simetris dan mengenkripsi atau mendekripsi plainteks secara digit per digit atau bit per bit[3]. Perbedaan dari RC4+ dengan RC4 original adalah pengaplikasian berbagai algoritma atau operasi matematika tambahan untuk memperkuat kemampuan enkripsi dari algoritma RC4. Sama seperti algoritma RC4, RC4+ juga memiliki dua sub-proses yang terdiri dari Key-Scheduling Algorithm (KSA) dan Pseudo Random Generation Algorithm (PRGA). Algoritma RC4+ yang akan dipakai adalah algoritma yang berdasarkan pada buku oleh Goutam Paul dan Subhamoy Maitra yaitu "RC4 Stream Cipher and Its Variants". Untuk KSA sendiri, proses yang dilalui diawali dengan proses algoritma RC4 original yaitu pembentukan larik dan perlakuan pengacakan dasar. Setelah itu, RC4+ tidak langsung masuk ke fase PRGA. KSA belum selesai karena adanya penambahan operasi matematika diantaranya aplikasi pengacakan menggunakan *Initialization Vector* yang berisi nilai acak bergantung pada key yang akan digunakan sebagai masukan. Untuk mempermudah proses dekripsi, pembentukan *IV* akan berdasarkan pada panjang *key* dan larik KSA sebelumnya. Setelah itu, ada aplikasi pengacakan secara zigzag yaitu pengacakan elemen berdasarkan sifat ganjil dan genap indeks. Berikut merupakan ilustrasi implementasi KSA dari RC4+.

```
# KSA plus
    S = [i for i in range(256)]
    j = 0
    # Basic Scramble
    for i in range(256):
        j = (j + S[i] + ord(key[i % len(key)])) % 256
        S[i], S[j] = S[j], S[i]
    # IV Scramble
    IV = [i for i in range(len(key))]
    for i in range(len(key)):
        IV[i] = S[i]
    j = 0
    for i in range(int(256/2)-1,-1,-1):
        j = ((j+S[i]) ^ (ord(key[i % len(key)])+IV[i % len(IV)])) %
256
        S[i], S[j] = S[j], S[i]
    for i in range(int(256/2),256,1):
        j = ((j+S[i]) ^ (ord(key[i % len(key)])+IV[i % len(IV)])) %
```

```
256
        S[i], S[j] = S[j], S[i]
    # ZigZag Scramble
    j = 0
    for y in range(256):
        temp = 0
        if y%2==0:
            temp = y/2
        else:
            temp = 256 - (y+1)/2
        i = int(temp)
        j = (j+S[i]+ord(key[i % len(key)])) % 256
        S[i], S[j] = S[j], S[i]
```

Fig. 6. Implementasi KSA Algoritma RC4+

Setelah terbentuk larik baru berdasarkan berbagai lapis pengacakan yang dilakukan oleh KSA dari algoritma RC4+, akan dilakukan fase PRGA. Perubahan PRGA ada pada variasi penggunaan *shift left* dan *shift right*. Berikut merupakan ilustrasi implementasi PRGA dari RC4+.

```
i,j = 0,0
    result = ""
    for k in range(len(plaintext)):
        i = (i+1) % 256
        j = (j+S[i]) % 256
        S[i], S[j] = S[j], S[i]
        t = (S[i]+S[j]) % 256
        tt = ((S[(i>>3 ^ j<<5)%256] + S[(i<<5 ^ j>>3)%256] % 256) ^
0xAA) % 256
        ttt = (j + S[j]) % 256
        u = (S[t]+S[tt]) ^ S[ttt]
        result += chr(u ^ ord(plaintext[k]))
```

Fig. 7. Implementasi PRGA Algoritma RC4+

III. RANCANGAN SIMULASI KOMPARASI

Untuk menjalankan simulasi komparasi antara algoritma RC4 dan RC4+, diperlukan berbagai batasan dan asumsi agar proses simulasi tidak kemana-mana. Berbagai batasan diantaranya adalah:

1. Proses yang digunakan adalah proses enkripsi dan proses deskripsi pesan dari file .txt yang memiliki bentuk file bertipe string.
2. Teks yang digunakan dapat berisi huruf, angka, maupun simbol termasuk spasi.
3. Algoritma yang digunakan adalah RC4, RC4+ (sebagai modifikasi RC4+), dan RC4+ modifikasi jika ada bentuk perbaikan dari RC4+ diambil dari hasil simulasi.
4. Komparasi yang dilakukan adalah perbandingan waktu enkripsi dan dekripsi antar algoritma.
5. Perhitungan waktu menggunakan modul *time* oleh python dan berbentuk ms.

Selain itu, berikut berbagai asumsi yang akan dipakai:

1. RC4+ sebagai bentuk modifikasi algoritma RC4 tentu melewati proses yang lebih banyak sehingga sewajarnya waktu yang dibutuhkan sedikit lebih banyak. Oleh karena itu, RC4+ dianggap tetap efisien jika lebih lambat kurang dari 1 ms dan dianggap lebih efisien jika lebih lambat kurang dari 0.5 ms.
2. RC4+ juga diasumsikan dapat beroperasi lebih cepat dari algoritma RC4. Jika terjadi, maka RC4+ jelas lebih efisien daripada RC4.
3. Jika terdapat suatu bentuk modifikasi untuk RC4+ yang dapat mengurangi waktu proses maka akan dicek juga dan tidak lupa dilampirkan berdasarkan pembahasannya.

Proses simulasi komparasi yang dilakukan adalah melalui berbagai file python yang berbeda yang masing-masing berisi algoritma terkait. File yang akan dipakai diawali dengan file rc4.py yang berisi implementasi algoritma RC4 sesuai landasan teori sebelumnya. Kemudian, akan digunakan file rc4plus.py yang juga berisi implementasi algoritma RC4+ sesuai landasan teori sebelumnya. Jika asumsi ketiga dapat dilakukan, maka akan dites file selanjutnya yang berisi modifikasi dari RC4+. File .txt yang akan diuji berisikan teks sepanjang 50 kata dengan isinya sebagai berikut:

“Tetapi saya harus menjelaskan kepada Anda bagaimana semua gagasan keliru tentang mencela kesenangan dan memuji rasa sakit ini lahir dan saya akan memberi Anda penjelasan lengkap tentang sistem ini, dan menjelaskan ajaran sebenarnya dari penjelajah besar kebenaran, pembangun utama kebahagiaan manusia. Tidak ada yang menolak, tidak menyukai, atau menghindari kesenangan”.

Karena aplikasi *time.time()* yang sedikit acak atau tidak konsisten, maka tes akan dilakukan hingga diperoleh 3 waktu enkripsi dan 3 waktu dekripsi bagi masing-masing algoritma. Setelah itu, hasil data waktu yang diperoleh akan dibandingkan dan ditarik kesimpulan.

IV. PENGUJIAN, ANALISIS, DAN PEMBAHASAN

Menggunakan kunci “kukuruyuk” yang memiliki 9 karakter, berikut hasil pengujian yang diperoleh.

Enkripsi	Waktu
1	1.00048828125
2	0.999755859375
3	1.0009765625
Rata-rata	1.000406901041667

Fig. 8. Hasil Pengujian Enkripsi Algoritma RC4

Dekripsi	Waktu
1	1.0009765625
2	1.000244140625
3	0.9990234375

Rata-rata	1.000081380208333

Fig. 9. Hasil Pengujian Dekripsi Algoritma RC4

Enkripsi	Waktu
1	0.999755859375
2	1.00146484375
3	1.100341796875
Rata-rata	1.033854166666667

Fig. 10. Hasil Pengujian Enkripsi Algoritma RC4+

Dekripsi	Waktu
1	1.01318359375
2	0.99951171875
3	1.001953125
Rata-rata	1.0048828125

Fig. 11. Hasil Pengujian Dekripsi Algoritma RC4+

Dapat diperhatikan dari berbagai tabel yang mewakili hasil pengujian yang telah dilakukan, sekilas dapat dicerna bahwa algoritma RC4 masih lebih cepat walaupun dengan *margin* yang amat sangat kecil. Diketahui rata-rata kecepatan enkripsi pada algoritma RC4 diperoleh ~1.0004 dan untuk rata-rata kecepatan dekripsi pada algoritma RC4 diperoleh ~1.00008. Sedangkan, rata-rata kecepatan yang diperoleh pada enkripsi algoritma RC4+ adalah ~1.0338 dan untuk rata-rata kecepatan yang diperoleh pada dekripsi algoritma RC4+ adalah ~1.00488.

Hasil sebelumnya dapat dibilang wajar karena algoritma RC4+ secara simpel melalui 4 proses lebih untuk mengenkripsi dan mendekripsi pesan. Namun demikian, menurut asumsi yang telah ditetapkan sebelumnya algoritma RC4+ tetap dianggap lebih efisien karena memiliki *margin* dibawah 0.5 ms dan tentu hasil enkripsinya akan lebih sulit dipecahkan karena proses enkripsinya lebih rumit.

Fase selanjutnya adalah melakukan modifikasi pada algoritma RC4+ untuk mendapatkan hasil yang lebih baik. Salah satu ide untuk mempercepat proses RC4+ adalah dengan membentuk kunci secara mandiri dalam algoritmanya. Menggunakan paper tesis yang ada pada referensi [3] sebagai basis dari ide ini. Modifikasi dilakukan dengan membentuk kunci mandiri dengan cara melakukan input dalam bentuk panjang kunci. Kunci kemudian akan di-*generate* berdasarkan pada *plaintext* dengan cara pembentukannya adalah mengambil elemen *plaintext* yang nilai ASCIInya genap dan pencarian elemen dimulai dari belakang. Berikut merupakan bentuk implementasi *key generator* dari ide tersebut.

```
# Keygen
    key = ""
    for i in range(len(plaintext)-1,-1,-1):
        if (ord(plaintext[i])%2==0):
            key += plaintext[i]
        if len(key) == length:
            break
```

Fig. 12. Hasil Implementasi *Key Generator*

Menggunakan panjang kunci 9 yang panjangnya sama dengan "kukuruyuk" pada pengujian sebelumnya, berikut adalah hasil pengujian RC4+ yang telah dimodifikasi.

Enkripsi	Waktu
1	0.98828125
2	0.999267578125
3	0.98779296875
Rata-rata	0.9851139322916667

Fig. 13. Hasil Pengujian Enkripsi Algoritma RC4+ Modifikasi

Dekripsi	Waktu
1	0.998779296875
2	1.00048828125
3	0.997802734375
Rata-rata	0.9990234375

Fig. 14. Hasil Pengujian Dekripsi Algoritma RC4+ Modifikasi

Dapat diperhatikan dari dua tabel tersebut, sekilas dapat dicerna bahwa pengaplikasian modifikasi pada algoritma RC4+ ini membuat proses enkripsi dan dekripsi menjadi sedikit lebih cepat. Diketahui rata-rata kecepatan yang diperoleh pada enkripsi algoritma RC4+ modifikasi adalah ~0.9851 dan untuk rata-rata kecepatan yang diperoleh pada dekripsi algoritma RC4+ modifikasi adalah ~0.99902. Hasil ini bahkan lebih cepat dibandingkan dengan algoritma RC4 original. Berdasarkan pada asumsi sebelumnya, sudah jelas, algoritma RC4+ modifikasi lebih efisien dari algoritma RC4 ataupun RC4+ biasa.

V. KESIMPULAN

Algoritma RC4 merupakan suatu algoritma *cipher* alir yang sudah lumayan mutakhir. Sayangnya, karena memang sudah umurnya yang cukup tua dan algoritma enkripsinya yang dapat diserang kelemahannya menggunakan komputer maupun program di zaman modern ini membuat algoritma RC4 tidak terlalu *reliable*. Namun demikian, tentu ada

berbagai upaya maupun ide-ide untuk meningkatkan kemampuan dari algoritma RC4 ini.

Terbentuklah algoritma RC4+ yang merupakan suatu jenis varian dari RC4 yang memiliki beberapa lapis lebih dalam proses enkripsinya. Melalui pengujian pada makalah ini, algoritma RC4+ juga terbukti lebih efisien daripada RC4 dalam hal keamanan dan setara dalam hal kecepatan. Bahkan lebih dari itu, RC4+ juga dapat dimodifikasi untuk memiliki kecepatan yang lebih efisien daripada algoritma RC4 biasa walaupun dengan *margin* yang sangat kecil. Hal lain yang harus disorot dari isi algoritma RC4+ dan RC4 adalah pengaruh panjang kunci. Walaupun tidak diuji, panjang kunci akan mempengaruhi RC4+ karena ada proses pembentukan IV yang bergantung pada panjang kunci sedangkan panjang kunci RC4 tidak akan berpengaruh karena hanya diambil saja elemen kuncinya. Maka dari itu, panjang kunci juga merupakan faktor yang harus diperhatikan dalam penggunaan algoritma RC4+.

VI. UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan puji syukur kepada Allah SWT karena berkat kehendak-Nya penulis dapat mendapatkan dorongan secara spiritual untuk terus mengerjakan tugas ini agar penulis dapat menyelesaikan tugas makalah untuk mata kuliah Kriptografi ini dengan baik. Selain itu, penulis juga ingin mengucapkan terima kasih kepada keluarga yang telah senantiasa memberikan *support* di rumah; juga kepada teman-teman yang selalu mengingatkan untuk terus memberikan proses pada makalah khususnya jika telah mulai mendekati *deadline*; serta berbagai pihak lain yang mungkin telah membantu dan tidak dapat disebut. Terakhir, penulis juga senantiasa ingin memberikan terima kasih sebanyak-banyaknya kepada Bapak Dr. Ir. Rinaldi Munir, MT. selaku dosen IF4020 Kriptografi yang telah memberikan ilmu pengetahuan selama perjalanan kuliah pada semester gasal Tahun 2021/2022 ini. Hal tersebut tentunya sangat berkontribusi pada proses pembentukan makalah ini.

VII. REFERENSI

[1] Munir, Rinaldi. "Pengantar Kriptografi". https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2021-2022/Pengantar-Kriptografi-(2021).pdf. Diakses pada 20 Desember 2021.

[2] Munir, Rinaldi. "Stream Cipher". https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2021-2022/Stream-Cipher-2021.pdf. Diakses pada 20 Desember 2021.

[3] Karo Karo, Siddik. "Modifikasi Algoritma RC4+ dalam Pembentukan Kunci Singkat Dinamis Untuk Pengamanan File". https://repositori.usu.ac.id/bitstream/handle/123456789/24681/177038037.pdf?sequence=1&isAllowed=y. Diakses pada 20 Desember 2021.

[4] Paul, G. & Maitra, S. 2012. "RC4 Stream Cipher and Its Variants". CRC Press Taylor & Francis Group: London.

VIII. 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.

Jakarta, 20 Desember 2021

Naufal Yahya Kurnianto 13519141