Optimasi Enkripsi dan Dekripsi Algoritma RSA Menggunakan Multi Threading

Kevin Fernaldy - 13516109 (*Penulis*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): kevin.fernaldy@gmail.com

***Abstract*—Saat ini, komunikasi digital sudah menjadi hal yang sangat umum di mata masyarakat. Pengiriman dan penerimaan data digital seperti SMS, berkas lagu, telepon, bahkan program komputer dilakukan selama 24 jam dan 7 hari seminggu. Namun, seringkali kita ingin mengirimkan pesan yang bersifat rahasia kepada orang lain tanpa diketahui isinya oleh pihak yang tidak diinginkan. Untuk itu, dibuatlah algoritma enkripsi dan dekripsi untuk mengacak isi pesan ketika dikirimkan. Pesan acak tersebut kemudian dikembalikan ke bentuk aslinya oleh penerima yang sah untuk dibaca isinya. Banyak algoritma enkripsi dan dekripsi yang telah dirilis hingga saat ini. Terdapat sebuah algoritma yang terkenal aman dan mudah untuk diimplementasi, tetapi memiliki kelemahan berupa waktu yang lama untuk melakukan enkripsi dan dekripsi. Apakah terdapat cara untuk melakukan optimasi terdapat algoritma tersebut?**

Keywords—RSA, multi-threading, enkripsi, dekripsi

I. Latar Belakang

Dalam dunia kriptografi, enkripsi dan dekripsi adalah salah satu subjek pembelajaran utama. Enkripsi adalah proses pengacakan pesan sebelum dikirimkan kepeda penerima yang membuat isi pesan tidak dapat dibaca atau dimengerti. Dekripsi adalah proses mengembalikan pesan acak yang diterima oleh pengirim untuk mendapatkan pesan awal yang dapat dibaca dan dimengerti. Proses enkripsi dan dekripsi ini membutuhkan sebuah kunci. Seperti layaknya sebuah kotak surat, pengirim pesan memasukkan surat ke dalam kotak dan mengunci kotak surat tersebut sehingga pesan tidak dapat dibaca oleh pihak yang tidak diinginkan. Hanya penerima surat dengan kunci yang benar yang dapat membuka dan membaca surat tersebut.

Kunci yang digunakan untuk melakukan enkripsi dan dekripsi ini dibangkitkan dengan sebuah algoritma. Algoritma ini terdiri dari dua buah jenis, yaitu algoritma pembangkitan kunci simetri, dan algoritma pembangkitan kunci simetri. Pada algoritma pembangkitan kunci simeteri, kunci yang digunakan untuk enkripsi dan dekripsi sama. Karena proses enkripsi dan dekripsi hanya dilakukan dengan satu buah kunci, maka proses tersebut cenderung dapat dilakukan dengan mudah, namun memiliki kelemahan berupa bentuk komunikasi unutuk mengirimkan kunci dari pengirim ke penerima. Pada algoritma pembangkitan kunci asimetri, terdapat dua buah kunci yang dibangkitkan. Dua buah kunci ini biasa disebut sebagai kunci puiblik dan kunci privat. Seperti pada namanya, kunci publik dapat diketahui oleh semua orang, sedangkan kunci privat harus dijaga kerahasiannya. Pada proses enkripsi, kunci publik digunakan untuk melakukan enkripsi sebelum dikirim ke penerima. Pesan yang telah dienkripsi ini hanya dapat dikembalikan oleh bentuk aslinya dengan kunci privat yang dimiliki oleh penerima, sehingga pihak lain yang memiliki kunci publik yang sama tidak dapat membaca isi pesan tersebut. Ketika pesan diterima oleh penerima yang sah, penerima kemudian menggunakan kunci privatnya untuk mengembalikan isi pesan ke bentuk semula. Algoritma pembangkitan kunci asimetri cukup sulit diaplikasikan ke dunia nyata karena algoritma pembangkitannya kompleks, namun memiliki keuntungan berupa tidak diperlukannya pengiriman kunci dari pengirim ke penerima. Penerima cukup memberikan kunci publik kepada orang-orang yang ingin mengirimkan pesan, karena pada dasarnya kunci publik memang dapat diketahui oleh publik.

Salah satu contoh algoritma pembangkitan kunci asimetri yang terkenal adalah algoritma Rivest-Shamir-Adleman (RSA). Algoritma RSA dikembangkan oleh R.L. Rivest, A. Shamir, dan L. Adleman, seperti nama algoritmanya. Algoritma RSA menggunakan faktorisasi dan bilangan prima untuk menentukan kunci publik dan privatnya, kemudian memanfaatkan modulo eksponensial untuk melakukan enkripsi dan dekripsi pesan. Kemanan dari algoritma RSA terletak pada ketidakmampuan manusia untuk melakukan faktorisasi biiangan sangat besar dengan efektif[1].

II. Implementasi Program

A. *Implementasi Algoritma RSA*

Algoritma RSA saya diimplementasi sesuai dengan spesifikasi algoritma RSA versi 2.2 dari *Internet Engineering Task Force* (IETF). Untuk menghindari permasalahan konkatenasi pesan dari algoritma PCKS#1, saya menggunakan algoritma *Optimal Asymmetric Encryption Padding* (OAEP) untuk melakukan kontatenasi blok pesan terakhir. Untuk fungsi *hash* pada OAEP, saya menggunakan fungsi *Secure Hash Algorithm* dengan panjang kunci 256-bit, atau lebih dikenal sebagai algoritma SHA256.

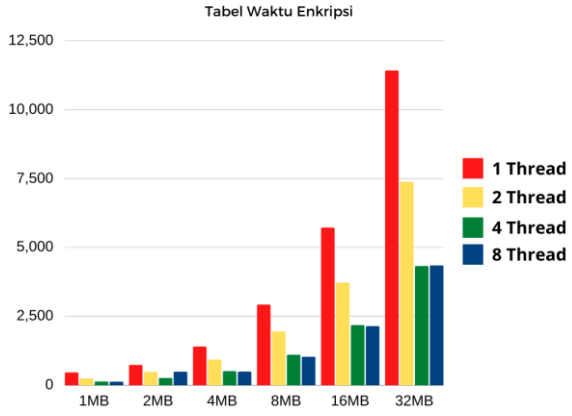
III. Pengetesan Implementasi Program

Pengetesan dilakukan dengan *Cenral Processing Unit* (CPU) Intel Core i5-8250U, dengan 4 *physical core*, dan 8

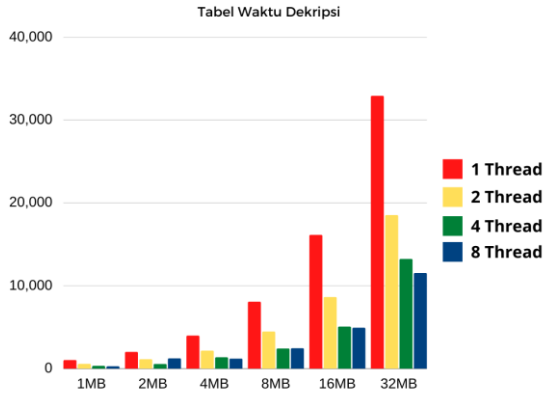
logical core. Jumlah *thread* yang digunakan adalah 1, 2, 4, dan 8 *thread*. Pengetesan dilakukan sebanyak dua belas kali dengan membuang hasil tertinggi dan terendah pada setiap komponen tes untuk membersihkan rata-rata dari nilai yang terlalu ekstrim. Variabel tes yang diukur adalah lama waktu eksekusi dalam *milisecond* (ms) untuk proses enkripsi dan dekripsi pada enam buah berkas berukuran 1MB, 2MB, 4MB, 8MB, 16MB, dan 32MB. Pengetesan pada jumlah *thread* 1 digunakan sebakai variabel kontrol.

Untuk bahasa pemograman yang dilakukan, saya menggunakan bahasa C++, dengan bantuan *GNU Multiple Precision Library* (GMP *library*) untuk melakukan perhitungan angka yang besar. Algoritma SHA256 dan RSA saya implementasikan sendiri sesuai dengan spesifikasi SHA256 pada dokumen IETF RFC-6234, dan spesifikasi RSA pada dokumen IETF RFC-8017.

A. *Hasil Pengetesan*



Gambar 1. Bagan Waktu Enkripsi



Gambar 2. Bagan Waktu Dekripsi

B. *Analisis*

Berdasarkan bagan hasil pengetesan waktu enkripsi dan dekripsi, didapat bahwa terdapat penurunan waktu yang signifikan dari implementasi *multithreading* terhadap algoritma RSA. Penurunan waktu signifikan terjadi pada penambahan jumlah *thread* dari 1 *thread* hingga 2 *thread*. Namun, terdapat penurunan waktu yang sangat kecil yaitu pada penambahan jumlah *thread* dari 2 *thread* hingga 4 *thread*. Penurunan waktu cenderung sangat sedikit, bahkan terdapat waktu eksekusi yang bertambah, seperti pada proses dekripsi berkas 2MB.

Perkiraan dari saya adalah, hal ini disebabkan oleh waktu *pembuatan thread* yang lebih lama dibandingkan waktu eksekusi, sehingga waktu pada 8 *thread* tidak memliki pengurangan waktu yang signifikan. Perkiraan kedua dari saya adalah terdapat *context-switching* pada eksekusi. *Context-switching* artinya proses pemindahan eksekusi program dari CPU ke memori untuk menjalankan program lain ke CPU. Hal ini dapat memperpanjang waku eksekusi, terlebih pada algoritma RSA yang cenderung berat.

IV. KESIMPULAN

Berdasarkan hasil percobaan dan analisis yang telah dilakukan, saya menentukan bahwa optimasi algoritma RSA berhasil dilakukan. Penambahan waktu yang signifikan terjadi pada penambahan jumlah *thread* dari 1 *thread* ke *2 thread*. Untuk performa rata-rata yang paling baik, 4 buah *thread* memberikan waktu enkripsi dan waktu dekripsi rata-rata yang terbaik.

ACKNOWLEDGMENT

Saya berterima kasih kepada Bapak Rinaldi Munir, Dosen Mata Kuliah Kriptografi yang telah memberikan ilmu dan pengetahuannya kepada saya dan mahasiswa kelas Kriptografi 2021-2022. Saya juga berterima kasih kepada keluarga saya yang selalu memberikan semangat dan kasih sayang dalam melakukan penulisan makalah ini.

REFERENCES

[1] M. Shandd and J. Vuillemin, "Fast Implementations of RSA Cryptography," Digital Equipment Corp., France: Paris Research Laboratory, 1993.

PERNYATAAN

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

Bandung, 20 Desember 2021