Jurnal Elektronik Ilmu Komputer Udayana
Volume 14, No 2. November 2025

p-ISSN: 2301-5373
e-ISSN: 2654-5101

The Implementation of the RSA-CRT Algorithm for Securing Banking Customer Data

Ni Wayan Amanda Putri Astawa[a1], I Made Widiartha[a2], Ngurah Agus Sanjaya ER[a3], I Gusti Agung Gede Arya Kadyanan[a4]

[a]Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Udayana
Kuta Selatan, Badung, Bali, Indonesia
[1]amandaputri@student.unud.ac.id
[2]madewidiartha@unud.ac.id
[3]agus_sanjaya@unud.ac.id
[4]gungde@unud.ac.id

Abstract

Data security is a crucial aspect in the digital era, particularly in sectors that handle sensitive information such as banking. This study implements and evaluates the RSA and RSA-CRT algorithms to secure customer data, focusing on their performance in terms of decryption efficiency, memory usage, and resistance to brute-force attacks. Experimental results demonstrate that RSA-CRT is significantly more efficient than standard RSA in decryption processes. Across five test files ranging from 6.53 MB to 33.1 MB and using 2048-bit keys, RSA-CRT achieved an average time efficiency improvement of approximately 71%. In terms of memory usage, RSA-CRT consumed slightly less memory than RSA for smaller files but tended to use more memory for larger files—up to 370.84 MB compared to RSA's 340.86 MB for a 33.1 MB file. Despite the increased memory usage, this trade-off is considered acceptable given the substantial speed gains. Additionally, brute-force testing revealed that decryption time increases exponentially with key length, making attacks on 2048-bit RSA keys highly impractical. These findings affirm that RSA-CRT is a viable modern cryptographic solution for protecting sensitive data, offering both high performance and robust security, provided the system's memory capacity can support the process.

Keywords: *Data Security, RSA Algorithm, Chinese Remainder Theorem (CRT), Cryptography, Banking Data Protection*

1. Pendahuluan

Dalam satu dekade terakhir, kemajuan teknologi informasi telah mengubah cara manusia mengelola dan menyimpan data. Di tengah pesatnya digitalisasi, keamanan data menjadi isu yang semakin krusial. Data sensitif, seperti informasi finansial, kini menjadi sasaran utama serangan siber. Kebocoran data yang masif di berbagai sektor menimbulkan kekhawatiran publik atas potensi penyalahgunaan informasi pribadi. Misalnya, antara Juli hingga September 2022, terjadi kebocoran 108,9 juta akun secara global, dengan 13,3 juta di antaranya berasal dari Indonesia—menjadikannya negara dengan kebocoran tertinggi ketiga di dunia dan mengalami peningkatan 1.370% dari kuartal sebelumnya [1].

Untuk menghadapi ancaman ini, dibutuhkan sistem keamanan yang kuat, efisien, dan praktis. Salah satu metode yang umum digunakan adalah algoritma kriptografi asimetris RSA (Rivest-Shamir-Adleman), yang menggunakan pasangan kunci publik dan privat. Keamanannya bergantung pada sulitnya memfaktorkan bilangan besar menjadi bilangan prima [2]. Meski telah lama digunakan, RSA memiliki keterbatasan dalam efisiensi, terutama saat menangani data besar atau sistem dengan sumber daya terbatas.

Optimalisasi RSA dapat dilakukan dengan *Chinese Remainder Theorem* (CRT), yang membagi proses dekripsi menjadi dua perhitungan modular dengan bilangan lebih kecil. Pendekatan ini

mempercepat komputasi dan menghemat sumber daya, menjadikannya solusi ideal bagi sistem dengan keterbatasan memori dan waktu [3].

Penelitian sebelumnya membuktikan efektivitas RSA dan RSA-CRT dalam mengamankan data digital. Saha dkk. (2025) menggabungkan RSA dan tokenisasi untuk mengamankan transaksi kartu kredit, sehingga mengurangi kebocoran data dan meningkatkan integritas sistem [5]. Erdiansyah dkk. (2020) menunjukkan bahwa Multi-Power RSA berbasis CRT lebih efisien dalam dekripsi dibanding RSA konvensional. Temuan ini membuka peluang eksplorasi lebih lanjut, terutama untuk sistem informasi nasabah perbankan yang membutuhkan keamanan tinggi dan pemrosesan cepat [6].

Penelitian ini bertujuan merancang sistem keamanan data nasabah bank menggunakan RSA-CRT yang dioptimalkan dari segi efisiensi dan ketahanan terhadap serangan. Fokus utamanya adalah meminimalkan risiko kebocoran data tanpa mengurangi integritas informasi, serta memberikan kontribusi terhadap pengembangan sistem keamanan digital, khususnya di sektor perbankan.

2. Metodologi Penelitian

2.1. Analisis Permasalahan

Tahapan awal dari penelitian ini adalah dengan melakukan identifikasi permasalahan. Permasalahan utama yang melatarbelakangi penelitian ini adalah kebutuhan akan sistem keamanan data yang andal untuk melindungi informasi sensitif, khususnya data nasabah bank, dari potensi kebocoran atau penyalahgunaan oleh pihak yang tidak bertanggung jawab. Di tengah meningkatnya ancaman serangan siber di era digital, data pribadi seperti milik nasabah menjadi target yang sangat rentan. Oleh karena itu, diperlukan mekanisme enkripsi yang tidak hanya kuat dari segi kriptografi, tetapi juga efisien secara komputasi, agar dapat diimplementasikan secara efektif pada sistem yang memiliki keterbatasan sumber daya, baik dari sisi waktu pemrosesan maupun kapasitas memori.

2.2. Pengumpulan Data

Tahapan selanjutnya, penulis melakukan pengumpulan data. Dataset yang penulis gunakan didapatkan dari Kaggle. Penulis menggunakan data dari Kaggle untuk mematuhi regulasi privasi dan keamanan data, karena data tersebut bersifat rahasia. Awalnya, data yang diperoleh berformat .csv, kemudian diubah menjadi format .xlsx agar sesuai dengan kebutuhan pengujian. Untuk contoh data yang diambil, dapat dilihat pada Tabel 1.

Tabel 1. Contoh Data Nasabah Bank

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	term_deposit
58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

2.3. Perancangan Sistem

Perancangan sistem yang digunakan dalam penelitian ini dibagi menjadi beberapa bagian yaitu analisis kebutuhan sistem, perancangan alur umum sistem, perancangan alur enkripsi menggunakan RSA, perancangan alur generasi kunci RSA, perancangan alur dekripsi menggunakan RSA-CRT, *Use Case* diagram, activity diagram, perancangan tampilan antar muka, dan pengujian dan evaluasi sistem.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

2.3.1. Analisis Kebutuhan Sistem

Langkah ini dilakukan untuk memperoleh pemahaman mendalam mengenai kebutuhan fungsional dan non-fungsional yang harus dipenuhi oleh sistem yang akan dibangun. Kebutuhan fungsional mencakup fitur dan fungsi utama sistem, seperti penerapan algoritma Rivest-Shamir-Adleman (RSA) dan *Chinese Remainder Theorem* (CRT). Di sisi lain, kebutuhan non-fungsional mencakup elemen pendukung sistem, baik dari aspek perangkat keras maupun perangkat lunak, yang diperlukan selama proses pengembangan dan implementasi. Dengan menjalani tahap ini, pengembangan sistem dapat disesuaikan dengan kebutuhan pengguna serta menjamin pengalaman penggunaan yang optimal [12].

a. Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan fitur atau layanan yang wajib dimiliki oleh sistem. Dalam konteks sistem yang sedang dikembangkan, kebutuhan ini mencakup kemampuan untuk mengolah data dari file, melakukan proses enkripsi dan dekripsi menggunakan algoritma RSA dan RSA-CRT, serta menampilkan hasilnya secara interaktif. Rincian kebutuhan fungsional sistem tersebut disajikan pada Tabel 2.

Tabel 2. Kebutuhan Fungsional

No	Deskripsi Kebutuhan Fungsional
KF1	Sistem dapat menampilkan halaman utama (Home Screen) dan navigasi fitur
KF2	Sistem dapat menerima input file dokumen berformat .xlsx
KF3	Sistem dapat mengenkripsi setiap data dalam file menggunakan RSA manual
KF4	Sistem dapat menyimpan dan mengunduh hasil enkripsi ke folder khusus
KF5	Sistem dapat melakukan dekripsi data dengan metode RSA dan RSA-CRT
KF6	Sistem dapat menyimpan dan mengunduh hasil dekripsi ke folder khusus

b. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan aspek penting yang berfokus pada hal-hal di luar fungsi utama sistem, namun tetap berperan besar dalam mendukung performa, keandalan, dan efisiensi operasional secara keseluruhan. Aspek-aspek ini mencakup, namun tidak terbatas pada, spesifikasi teknis perangkat keras (*hardware*) dan perangkat lunak (*software*) yang digunakan selama proses pengembangan dan pengujian. Dengan kata lain, kebutuhan non-fungsional memastikan sistem dapat beroperasi secara optimal dalam lingkungan yang telah ditentukan. Rincian lebih lanjut mengenai kebutuhan ini dapat dilihat pada Tabel 3 yang memuat spesifikasi perangkat keras, serta Tabel 4 yang menjelaskan perangkat lunak yang digunakan.

Tabel 3. Spesifikasi *Hardware* Pengembangan Sistem

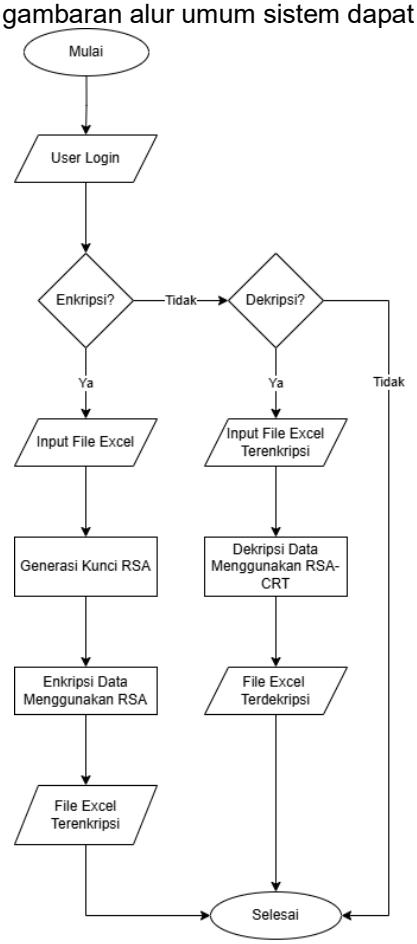
No	Perangkat	Keterangan
1	Laptop	Acer Nitro 5
2	Processor	AMD Ryzen 7-3750H
3	Monitor	15.6 inch
4	RAM	8GB
5	Penyimpanan	512GB SSD

Tabel 4. Spesifikasi *Software* Pengembangan Sistem

No	Perangkat	Keterangan
1	Sistem Operasi	Windows 10/11
2	Platform Aplikasi	Desktop (Python + Kivy)
3	Text Editor	Visual Studio Code untuk pengembangan kode
4	Library Tambahan	openpyxl, gmpy2, json, time, dan lain-lain.

2.3.2. Perancangan Alur Umum Sistem

Secara garis besar, sistem yang dikembangkan memiliki dua fitur utama, yakni enkripsi dan juga dekripsi file. Proses dimulai ketika pengguna berhasil login ke dalam sistem. Setelah masuk, pengguna diberikan dua opsi utama yakni enkripsi dan dekripsi. Jika pengguna memilih enkripsi, sistem akan meminta pengguna untuk mengunggah file Excel yang akan dienkripsi. Selanjutnya, sistem akan melakukan generasi kunci RSA, yang digunakan dalam proses enkripsi. Setelah kunci RSA dibuat, data dalam file Excel akan dienkripsi menggunakan algoritma RSA, menghasilkan file Excel terenkripsi sebagai output akhir. Sebaliknya, jika pengguna memilih dekripsi, sistem akan meminta pengguna untuk mengunggah file Excel terenkripsi. Proses dekripsi dilakukan menggunakan metode RSA-CRT (*Chinese Remainder Theorem*), yang bertujuan untuk meningkatkan efisiensi dalam proses dekripsi. Setelah proses dekripsi selesai, sistem menghasilkan file Excel terdekripsi yang dapat diakses kembali dalam format aslinya. Untuk melihat gambaran alur umum sistem dapat dilihat pada Gambar 1.



Gambar 1. Perancangan Alur Umum Sistem

2.3.3. Perancangan Alur Enkripsi Menggunakan RSA

Penelitian ini menggunakan algoritma RSA (Rivest-Shamir-Adleman) untuk proses enkripsi. Tahap awalnya adalah mengubah plaintext menjadi angka M, lalu menghitung ciphertext C menggunakan kunci publik dengan rumus (1).

$$C = M^e \mod n \quad \text{..... (1)}$$

Keterangan:

C (Ciphertext) : Hasil dari proses enkripsi, yaitu data yang sudah diubah menjadi bentuk tidak terbaca.

M (Plaintext) : Pesan asli atau data awal yang ingin dienkripsi.

e : Eksponen publik. Bagian dari kunci publik yang digunakan dalam proses enkripsi.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

n : Hasil dari perkalian dua bilangan prima besar, dan merupakan bagian dari kunci publik maupun privat.

Algoritma RSA sendiri bekerja dengan prinsip penggunaan dua kunci, yaitu kunci publik untuk enkripsi dan kunci privat untuk dekripsi. Pembuatan kunci menghasilkan sepasang kunci yang terdiri dari kunci publik, yang dapat didistribusikan untuk proses enkripsi, dan kunci privat, yang dirahasiakan untuk proses dekripsi [8].

Dalam implementasinya, pengguna memilih operasi enkripsi dan menentukan file Excel berisi data nasabah. Sistem membaca dan mengenkripsi isi file menggunakan kunci publik RSA, lalu menyimpan hasilnya ke file Excel baru yang telah terlindungi. Proses ini memastikan data aman dari akses tidak sah.

2.3.4. Perancangan Alur Generasi Kunci RSA

Dalam proses pembentukan kunci RSA, terdapat sejumlah tahapan penting yang perlu dilakukan untuk menghasilkan sepasang kunci yang sah dan aman, yakni kunci publik dan kunci privat. Langkah-langkah ini dijelaskan secara rinci berikut ini beserta pengertian dari setiap variabel yang digunakan [7]:

a. Menentukan 2 bilangan prima besar secara acak (p dan q)
   Langkah pertama adalah memilih dua bilangan prima yang besar secara acak dan berbeda satu sama lain. Pemilihan p dan q yang unik ini bertujuan untuk mencegah upaya faktorisasi yang mudah terhadap kunci.

b. Menghitung nilai n
   Setelah mendapatkan p dan q, nilai n dihitung dengan rumus (2).
   $$n = p \cdot q \quad \text{..... (2)}$$
   Keterangan:
   p dan q : Bilangan prima acak yang sudah ditentukan sebelumnya.
   Nilai n digunakan sebagai modulus dalam kunci publik dan kunci privat. Fungsi utamanya adalah membatasi ruang lingkup operasi enkripsi dan dekripsi (modulo n), serta memengaruhi kekuatan kunci RSA. Semakin besar nilai n, semakin sulit untuk memecahkannya menjadi faktor asal (p dan q).

c. Menghitung fungsi totient (m)
   Selanjutnya, dihitung fungsi totient Euler, dilambangkan sebagai m, menggunakan rumus (3).
   $$m = (p-1)(q-1) \quad \text{..... (3)}$$
   m : Totien Euler digunakan untuk mencari kunci privat (d) yang sesuai dengan kunci publik (e).
   Fungsi totient ini merepresentasikan jumlah bilangan bulat yang lebih kecil dari n dan relatif prima terhadap n [4]. Dalam konteks RSA, nilai m sangat penting karena digunakan untuk menentukan pasangan kunci yang valid antara kunci publik e dan kunci privat d. Nilai ini menjadi fondasi matematika yang menjamin bahwa proses enkripsi dan dekripsi dapat saling mengembalikan hasilnya secara akurat.

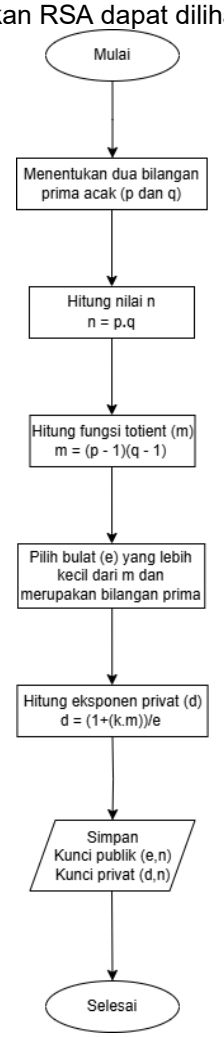
d. Memilih eksponen publik (e)
   Setelah memperoleh m, langkah berikutnya adalah memilih nilai e yang merupakan eksponen publik. Nilai e harus memenuhi dua syarat, yakni harus lebih besar dari 1 dan lebih kecil dari m, serta relatif prima terhadap m. Biasanya dipilih nilai umum seperti e = 65537 karena merupakan bilangan prima yang kecil namun aman dan efisien secara komputasi. Nilai e digunakan dalam proses enkripsi dan menjadi bagian dari kunci publik bersama dengan n.

e. Menghitung eksponen privat (d.)
   Setelah e ditentukan, eksponen privat d. dihitung berdasarkan persamaan (4)
   $$e \cdot d \equiv 1 \ (mod\ m) \quad \text{..... (4)}$$
   Atau dalam bentuk lain:
   $$d = \frac{1+(k \cdot m)}{e} \quad \text{..... (5)}$$

   Keterangan:
   d. : Eksponen publik

≡ : Kongruen, artinya "punya sisa yang sama setelah dibagi m"
k : Bilangan bulat sembarang, digunakan dalam definisi kongruensi
Persamaan ini menunjukkan bahwa d adalah invers modular dari e terhadap m. Dengan mencoba nilai-nilai integer k yang sesuai, diperoleh nilai d yang valid. Nilai d ini sangat krusial karena digunakan untuk mendekripsi data yang telah dienkripsi menggunakan kunci publik. Oleh karena itu, kerahasiaan d harus dijaga dengan ketat agar hanya dapat diakses oleh pihak yang berwenang.

Perancangan alur enkripsi menggunakan RSA dapat dilihat pada Gambar 2.



Gambar 2. Perancangan Alur Generasi Kunci RSA

Melalui serangkaian tahapan tersebut, dihasilkan sepasang kunci dengan fungsi spesifik dalam menjaga keamanan data. Kunci publik, yang terdiri dari pasangan (e, n), dapat disebarkan secara terbuka untuk keperluan enkripsi. Sebaliknya, kunci privat (d, n) harus dirahasiakan karena digunakan untuk proses dekripsi. Keamanan algoritma RSA sangat ditentukan oleh pemilihan bilangan prima yang cukup besar dan penyimpanan kunci privat yang aman. Kombinasi kedua aspek ini menjadikan RSA salah satu algoritma kriptografi paling tepercaya dalam menjaga kerahasiaan dan keutuhan data.

2.3.5. Perancangan Alur Dekripsi Menggunakan RSA-CRT

Dalam metode dekripsi RSA standar, pesan asli (plaintext) M dapat diperoleh kembali dari ciphertext C menggunakan kunci privat melalui persamaan (6) [10].

$$M = C^d \ mod \ n \quad \text{..... (6)}$$

Persamaan ini merupakan inti dari algoritma RSA, di mana C merupakan ciphertext, d adalah eksponen privat yang bersifat rahasia, dan n adalah hasil dari perkalian dua bilangan prima. Mekanisme ini didasarkan pada prinsip eksponensial modular, yaitu perhitungan pangkat dari C

p-ISSN: 2301-5373
e-ISSN: 2654-5101

dengan eksponen d, lalu mengambil hasilnya dalam modulo n. Nilai d sebelumnya telah dihitung sebagai invers dari e (eksponen publik) terhadap fungsi totien m. Hubungan ini memastikan bahwa hasil dekripsi akan sama dengan pesan asli M.
Namun, proses dekripsi dengan nilai d yang sangat besar bisa menjadi sangat lambat, khususnya jika nilai n berukuran besar (misalnya RSA 2048-bit) [11]. Untuk mengatasi hal ini, teknik CRT (*Chinese Remainder Theorem*) digunakan untuk mempercepat proses dekripsi dengan membagi perhitungan menjadi dua bagian lebih kecil, masing-masing dalam modulo p dan q. Tahapannya adalah sebagai berikut [9]:

a. Membagi nilai d menjadi dp dan dq
Untuk mengoptimalkan proses dekripsi, eksponen privat d dibagi menjadi dua bagian, yaitu dp dan dq, yang dihitung dengan rumus (7) dan (8).

$$dp = d \bmod (p-1) \quad \text{..... (7)}$$
$$dq = d \bmod (q-1) \quad \text{..... (8)}$$

Keterangan:
dp : hasil modulus dari kunci privat d terhadap (p-1)
dq : hasil modulus dari kunci privat d terhadap (q-1)
Nilai dp dan dq berfungsi untuk mereduksi eksponen privat d ke dalam ruang bilangan modular yang lebih kecil, yang memungkinkan proses eksponensial dilakukan lebih cepat tanpa harus langsung bekerja pada ruang modulus besar. Ini adalah inti dari optimasi yang dilakukan CRT, karena dua perhitungan kecil lebih cepat dibanding satu perhitungan besar.

b. Menghitung q_{inv}
Langkah selanjutnya adalah menghitung q_{inv}, yakni nilai invers modular dari q terhadap p, menggunakan rumus (9).

$$q_{inv} = q^{-1} \bmod p \quad \text{..... (9)}$$

Keterangan:
q_{inv} = Modular inverse dari q terhadap p
Nilai q_{inv} ini penting karena akan digunakan dalam proses rekonstruksi akhir untuk menggabungkan dua hasil dekripsi yang diperoleh dari modulo p dan q. Nilai ini dihitung sekali saat pembuatan kunci privat dan disimpan untuk mempercepat proses dekripsi.

c. Mendekripsi ciphertext dalam dua bagian (m_1 dan m_2)
Pada tahap ini, ciphertext c didekripsi dengan membagi prosesnya menjadi dua bagian eksponensial yang lebih kecil, yaitu m_1 dan m_2. Perhitungan dilakukan menggunakan rumus (10) dan (11).

$$m_1 = c^{dp} \bmod p \quad \text{..... (10)}$$
$$m_2 = c^{dq} \bmod q \quad \text{..... (11)}$$

Keterangan:
m_1 : Hasil dekripsi sebagian dari ciphertext c menggunakan modulus p
m_2 : Hasil dekripsi sebagian dari ciphertext c menggunakan modulus q
Nilai-nilai ini dihasilkan melalui eksponensial modular pada ruang yang lebih kecil, sehingga secara signifikan mempercepat proses dibandingkan jika dilakukan langsung dengan d dan n. Nilai dp dan dq merupakan bentuk pengurangan eksponen d ke dalam ruang lebih kecil, dan inilah yang membuat metode RSA-CRT lebih efisien dari segi performa komputasi.

d. Menghitung nilai perantara h
Setelah mendapatkan m_1 dan m_2, langkah berikutnya adalah menghitung nilai penyesuaian h, yang digunakan untuk menyatukan hasil dekripsi. erhitungannya dilakukan dengan rumus (12).
h : Nilai sementara yang dihitung untuk membantu menggabungkan hasil dekripsi dari dua modulus p dan q.
Nilai h ini digunakan untuk mengkompensasi selisih antara hasil m_1 dan m_2, agar dapat disatukan secara konsisten. Nilai q_{inv} yang sudah dihitung sebelumnya digunakan dalam proses ini. Fungsi utama dari h adalah sebagai penghubung matematis antara

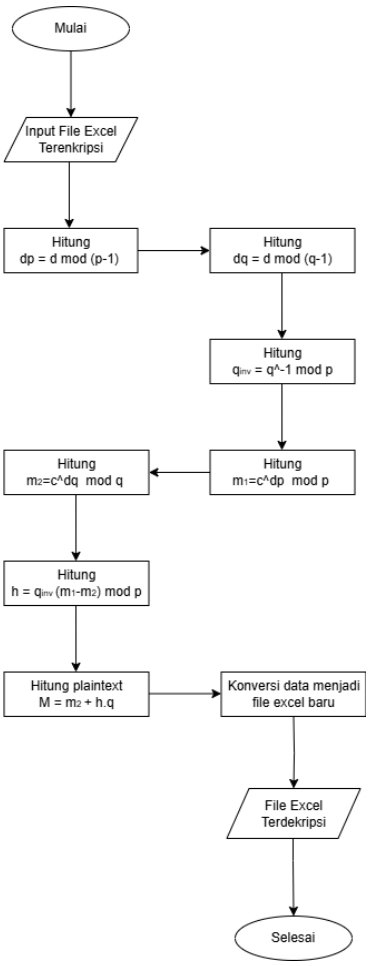
dua hasil dekripsi terpisah agar bisa dikombinasikan menjadi satu pesan yang utuh dan akurat. Nilai ini membantu menyesuaikan perbedaan hasil dekripsi dan menjadi komponen penting dalam menyusun kembali plaintext secara benar.

e. Mendapatkan plaintext (M)
Langkah terakhir dalam proses dekripsi menggunakan CRT adalah menggabungkan dua hasil parsial tersebut untuk membentuk kembali plaintext menggunakan rumus (13).

$$M = m_2 + h \cdot q \quad \text{..... (13)}$$

Persamaan ini digunakan untuk menggabungkan hasil dekripsi dari dua ruang modular (p dan q). Nilai m_2 berasal dari hasil dekripsi pada modulo q, sedangkan $h{\cdot}q$ adalah bagian korektif berdasarkan perhitungan nilai h yang dikalikan dengan q. Hasilnya akan berada dalam ruang modulo n sebagaimana seharusnya.
Penggunaan metode CRT memungkinkan penggabungan dua hasil dekripsi dari ruang bilangan kecil menjadi satu nilai yang benar dalam ruang yang lebih besar. Dengan demikian, efisiensi meningkat tanpa mengorbankan keakuratan hasil dekripsi.

Gambaran dari implementasi proses dekripsi dapat dilihat pada Gambar 3.



Gambar 3. Perancangan Alur Proses Dekripsi

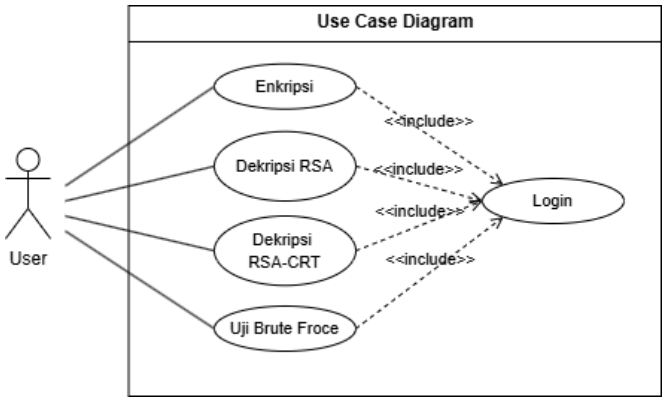
Dalam implementasi dekripsi, proses diawali dengan pengguna memilih tindakan dekripsi dan menentukan file Excel terenkripsi yang akan diproses. Sistem kemudian membaca file tersebut dan menggunakan kunci privat RSA-CRT untuk mendekripsi data. RSA-CRT memungkinkan sistem melakukan dekripsi dengan lebih efisien dibandingkan metode RSA standar, karena memanfaatkan sifat matematika dari *Chinese Remainder Theorem*. Setelah proses dekripsi selesai, data yang telah dikembalikan ke bentuk aslinya dikonversi kembali ke format awalnya sebelum akhirnya ditulis kembali ke dalam file Excel yang baru. Hasil akhir dari proses ini

p-ISSN: 2301-5373
e-ISSN: 2654-5101

adalah file Excel yang telah didekripsi, memungkinkan pengguna untuk mengakses kembali data dalam keadaan tidak terenkripsi.

2.3.6. *Use Case Diagram*

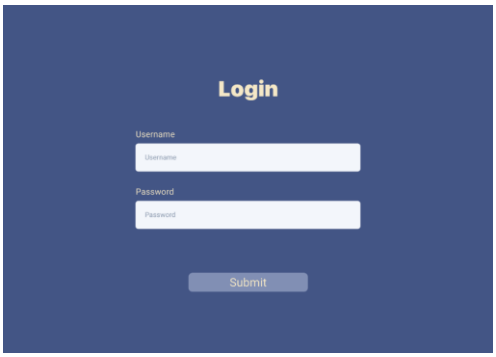
Pada sistem yang dirancang, aktor utama adalah *User* yang memiliki peran sentral dalam menjalankan seluruh proses utama yang berkaitan dengan pengamanan data file dokumen berformat .xlsx. *User* memiliki otoritas penuh terhadap proses enkripsi dan dekripsi data menggunakan algoritma RSA manual maupun RSA-CRT (*Chinese Remainder Theorem*), serta dapat melakukan pengujian ketahanan enkripsi melalui simulasi *Brute Force*. Berdasarkan diagram *Use Case*, seluruh proses utama tersebut mengharuskan *User* untuk terlebih dahulu melakukan autentikasi melalui fitur Login, yang ditunjukkan dengan relasi <<*include*>> dari masing-masing *Use Case* ke *Use Case* Login. Hal ini bertujuan untuk menjamin keamanan sistem dan membatasi akses hanya kepada pengguna yang berwenang. Dalam implementasinya, *User* bertanggung jawab dalam memilih file yang akan diproses, menjalankan proses enkripsi atau dekripsi sesuai kebutuhan, serta mengunduh file hasil pengolahan. Selain itu, *User* juga berkewajiban untuk memastikan bahwa proses yang dilakukan berjalan dengan benar sehingga data yang dihasilkan tetap akurat dan aman. Gambaran dari *Use Case* diagram untuk sistem yang dirancang dapat dilihat pada Gambar 4.



Gambar 4. *Use Case Diagram*

2.3.7. Perancangan Tampilan Antar Muka

Terdapat beberapa rancangan antar muka untuk mendukung sistem yang dibuat. Pada Gambar 7 terdapat halaman *login* agar *User* dapat masuk ke dalam sistem. Kemudian pada Gambar 8 terdapat halaman *home* yang di mana pada halaman tersebut *User* dapat memilih untuk melakukan enkripsi file, dekripsi file, atau uji *Brute Force*. Gambar 9 merupakan tampilan dari halaman enkripsi, dan Gambar 10 merupakan tampilan halaman menu dekripsi. Pada halaman menu dekripsi, *User* dapat memilih dekripsi menggunakan CRT yang ditunjukkan pada Gambar 11, atau memilih dekripsi tanpa menggunakan CRT yang ditunjukkan pada Gambar 12. Selain itu, pada Gambar 13 merupakan halaman uji *Brute Force* agar *User* dapat melakukan simulasi ketahanan sistem.

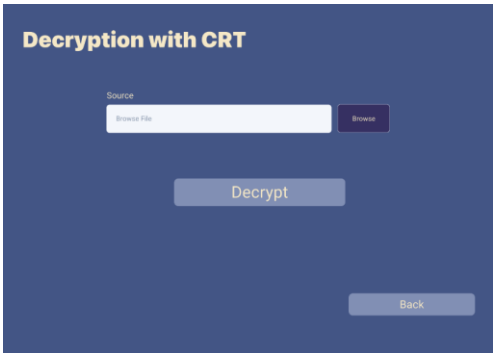


Gambar 5. Rancangan Halaman *Login*

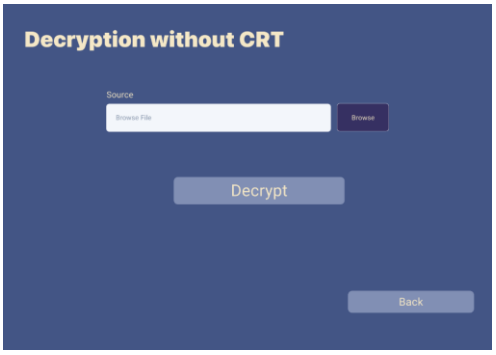
Gambar 6. Rancangan Halaman *Home*

Gambar 7. Rancangan Halaman Enkripsi

Gambar 8. Rancangan Halaman Menu Dekripsi



Gambar 9. Rancangan Halaman Dekripsi RSA-CRT



Gambar 10. Rancangan Halaman Dekripsi RSA

Gambar 11. Rancangan Halaman *Brute Force Test*

2.3.9. Pengujian dan Evaluasi Sistem

Perancangan evaluasi sistem adalah suatu proses yang dibuat untuk menilai kinerja dan kualitas dari sistem yang telah dikembangkan. Tujuannya adalah untuk menilai sejauh mana sistem tersebut mampu mencapai tujuan yang telah ditentukan, serta menemukan kelemahan atau aspek yang masih perlu diperbaiki.

a. Waktu Proses

Penelitian ini bertujuan untuk menguji waktu proses pada tahap dekripsi menggunakan algoritma RSA. Fokus penelitian ini adalah membandingkan kecepatan dekripsi antara metode RSA standar dengan metode RSA-CRT terhadap input data yang sama. Dengan demikian, penelitian ini akan memberikan pemahaman yang lebih mendalam

p-ISSN: 2301-5373
e-ISSN: 2654-5101

mengenai peningkatan efisiensi sistem dalam proses dekripsi menggunakan teknik CRT.

b. Penggunaan Memori (*memory usage*)
Penelitian ini juga bertujuan untuk mengamati penggunaan memori selama proses dekripsi menggunakan algoritma RSA. Fokus pengamatan diarahkan pada perbandingan efisiensi memori antara metode RSA standar dan metode RSA-CRT ketika memproses input data yang identik. Dengan membandingkan konsumsi memori dari kedua pendekatan tersebut, penelitian ini diharapkan dapat memberikan wawasan mengenai dampak optimisasi CRT tidak hanya terhadap kecepatan, tetapi juga terhadap efisiensi sumber daya sistem, khususnya memori yang digunakan selama proses dekripsi.

c. Uji Ketahanan *Brute Force*
Pengujian dalam penelitian ini difokuskan pada evaluasi ketahanan sistem terhadap serangan *Brute Force* menggunakan metode faktorisasi Fermat. Pengujian dilakukan untuk memastikan bahwa sistem mampu menangani upaya dekripsi tanpa izin dengan menguji bagaimana kunci privat dapat diturunkan dari kunci publik.
Dengan pengujian ini, peneliti dapat mengevaluasi sejauh mana sistem mempertahankan keamanannya terhadap serangan *Brute Force* serta memverifikasi bahwa algoritma RSA-CRT yang diterapkan memberikan perlindungan yang andal dalam skenario ancaman tersebut.

3. Hasil dan Diskusi

3.1. Analisis Permasalahan

Sebagai upaya untuk menjawab permasalahan yang telah diidentifikasi, penelitian ini mengusulkan penerapan kombinasi algoritma RSA dengan teknik Chinese Remainder Theorem (CRT). RSA dipilih karena merupakan algoritma kunci publik yang telah terbukti keamanannya dan banyak digunakan dalam berbagai sistem kriptografi. Namun, proses dekripsi pada RSA standar cenderung membutuhkan waktu eksekusi yang cukup lama, terutama ketika digunakan dengan kunci berukuran besar. Untuk mengatasi keterbatasan tersebut, CRT digunakan sebagai metode optimalisasi yang bertujuan untuk meningkatkan efisiensi proses dekripsi tanpa mengurangi tingkat keamanan sistem. Pendekatan ini menjadi dasar dalam perancangan dan pengujian sistem yang dilakukan dalam penelitian.

3.2. Proses Pengumpulan Data

Data yang digunakan dalam penelitian ini diperoleh melalui metode pengumpulan data sekunder. Sumber data berasal dari situs web kaggle.com dengan menggunakan kata kunci “bank customers data”. Awalnya, data yang diperoleh berformat .csv, kemudian diubah menjadi format .xlsx agar sesuai dengan kebutuhan pengujian. File Excel tersebut selanjutnya dibagi menjadi lima ukuran yang berbeda, yaitu file berukuran 156KB, 305KB, 452KB, 601KB, dan 750KB, guna memungkinkan pengujian sistem terhadap berbagai skala data.

3.3. Implementasi Sistem

3.3.1. Implementasi Algoritma RSA-CRT

Penelitian ini mengimplementasikan kombinasi algoritma RSA dengan optimasi Chinese Remainder Theorem (CRT) menggunakan bahasa pemrograman Python. Sistem yang dikembangkan memiliki dua fungsi utama, yaitu enkripsi dan dekripsi data. Fokus utama adalah mengamankan file Excel (*.xlsx) yang berisi data nasabah bank, sehingga hanya dapat dibaca kembali melalui proses dekripsi yang sah.

a. Proses Enkripsi

Proses enkripsi pada sistem ini bertujuan untuk mengubah plainfile data nasabah perbankan menjadi cipherfile menggunakan algoritma RSA, sehingga isi file tidak dapat dibaca secara langsung tanpa proses dekripsi. Proses enkripsi terdiri atas beberapa tahapan, yaitu:

1. Input file

   Pengguna diminta untuk memilih file berformat .xlsx yang akan dienkripsi. File tersebut kemudian dibaca dan diproses lebih lanjut dalam sistem enkripsi berbasis RSA.

2. Pembuatan kunci RSA

   Sebelum enkripsi dapat dilakukan, sistem terlebih dahulu menghasilkan pasangan kunci RSA yang terdiri dari *public key* (n,e) dan *private key* (n,d). Untuk mendukung optimasi dekripsi menggunakan CRT, juga dihitung parameter tambahan berupa p, q, dp, dq, dan q_{inv}. Tahapan pembuatan kunci dimulai dengan:

   a) Menghasilkan dua bilangan prima besar (p dan q) menggunakan fungsi generate_large_prime, yang menggunakan pencarian berbasis bilangan acak dan pengujian keprimaan dengan batas waktu tertentu.

   b) Menghitung nilai-nilai penting, yaitu:
      a) Modulus: n = p × q
      b) Totien Euler (φ(n)): m = (p - 1)(q - 1)
      c) Eksponen publik: e (biasanya 65537, dipilih agar relatif prima terhadap m)
      d) Eksponen privat: d (merupakan invers modular dari e terhadap m)

   c) Parameter CRT:
      a) dp = d mod (p - 1)
      b) dq = d mod (q - 1)
      c) $q_{inv} = q^{-1} \, mod \, p$

   Seluruh parameter kunci kemudian disimpan dalam file berformat JSON untuk memudahkan pemanggilan kembali pada proses enkripsi maupun dekripsi.

3. Proses enkripsi data menggunakan RSA

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Setelah kunci publik tersedia, sistem akan membaca file dalam format string atau byte. Setiap nilai dalam file akan dikonversi menjadi bilangan bulat, lalu dienkripsi satu per satu menggunakan persamaan dasar RSA.

Proses enkripsi dilakukan dengan fungsi encrypt_excel, yang menangani konversi dari string ke byte, lalu ke bilangan bulat panjang. Sebelum dienkripsi, sistem memastikan bahwa setiap nilai *plaintext* lebih kecil dari nilai modulus n. Jika syarat ini tidak terpenuhi, enkripsi dibatalkan untuk nilai tersebut guna menjaga integritas data.

Perhitungan dilakukan menggunakan pustaka gmpy2 yang mampu menangani operasi modular pada bilangan besar secara efisien. Jika terjadi kesalahan, seperti nilai yang melebihi batas modulus atau kegagalan konversi data, sistem akan menampilkan peringatan dan menyimpan nilai asli sebagai fallback.

4. Struktur penyimpanan *ciphertext*
   Hasil enkripsi disimpan dalam format bilangan besar yang ditulis ke dalam file baru. Karena setiap karakter *plaintext* diubah menjadi bilangan besar berdasarkan modulus n, ukuran file terenkripsi akan lebih besar dibandingkan file aslinya. Meski demikian, struktur penyimpanan tetap dirancang efisien dan terorganisir untuk memudahkan proses dekripsi di tahap selanjutnya.
5. Output file enkripsi
   Setelah proses enkripsi selesai, file terenkripsi disimpan dalam direktori khusus (assets/encrypted_files/) dengan penamaan sesuai format encrypted_<nama_file_asli>.xlsx. File ini hanya dapat diakses atau dikembalikan ke bentuk semula melalui proses dekripsi menggunakan private key yang sesuai, sehingga keamanan data sensitif tetap terjamin.

b. Proses Dekripsi
   Proses dekripsi merupakan tahapan untuk mengembalikan data terenkripsi (*cipherfile*) ke bentuk semula (*plainfile*) agar dapat dibaca dan digunakan kembali sebagaimana mestinya. Proses ini dilakukan secara bertahap melalui beberapa langkah sebagai berikut:
   1. Input file
      Langkah awal dekripsi melibatkan pemilihan file .xlsx hasil enkripsi sebelumnya yang akan dikembalikan ke bentuk *plaintext*. File ini telah mengalami modifikasi data melalui proses enkripsi, dan akan diproses lebih lanjut dalam tahap dekripsi.
   2. Pemanggilan kunci privat (private key)
      Sebelum proses dekripsi dimulai, sistem harus memuat kunci privat RSA yang sebelumnya telah disimpan dalam file berformat JSON. Terdapat dua jenis kunci privat dalam sistem ini, yaitu:
      a) Kunci Privat RSA-CRT, yang berisi parameter tambahan seperti p, q, dp, dq, dan q_{inv}. Parameter-parameter ini memungkinkan proses dekripsi dilakukan secara lebih cepat karena operasi eksponensial modular dapat dihitung pada bilangan yang lebih kecil (modulus p dan q) dibandingkan RSA standar.
      b) Kunci Privat RSA Standar, yang hanya terdiri dari nilai d (eksponen privat) dan n (modulus), digunakan dalam proses dekripsi dengan metode RSA biasa tanpa optimasi.

      Kedua jenis kunci ini dimuat dari file masing-masing dan dikonversi ke dalam bentuk bilangan presisi tinggi agar sesuai untuk perhitungan matematis skala besar.
   3. Proses dekripsi menggunakan algoritma RSA-CRT
      Setelah kunci privat RSA-CRT dimuat, proses dekripsi dilakukan terhadap setiap elemen *ciphertext* dalam file. Metode dekripsi RSA-CRT membagi proses menjadi dua bagian dengan menggunakan modulus p dan q, lalu menggabungkannya kembali menggunakan rumus CRT. Teknik ini secara signifikan mempercepat proses dekripsi dibandingkan metode RSA standar karena mengurangi beban komputasi.

      Hasil dari setiap proses dekripsi berupa karakter asli *(plaintext)* yang kemudian akan disusun ulang sesuai format awal file sebelum dilakukan enkripsi.
   4. Dekripsi Data dengan RSA Standar

Jika proses dekripsi dilakukan menggunakan kunci privat RSA biasa, maka setiap ciphertext akan diproses menggunakan operasi eksponensial modular dengan nilai d dan n. Meskipun secara algoritmis lebih sederhana, metode ini memiliki performa yang lebih rendah dibanding RSA-CRT karena operasi dilakukan pada modulus yang jauh lebih besar (n), sehingga memerlukan waktu komputasi yang lebih tinggi.

5. Rekonstruksi file
   Setelah seluruh elemen data berhasil didekripsi, data disusun kembali dalam bentuk file Excel dengan struktur yang menyerupai file asli sebelum proses enkripsi. Proses ini meliputi pengembalian data ke posisi sel semula, pemulihan format, dan tipe data untuk memastikan konsistensi informasi.
6. Output file dekripsi
   Langkah terakhir dari proses dekripsi adalah penyimpanan hasil dekripsi ke dalam file baru dengan format .xlsx. File ini disimpan di direktori khusus dengan penamaan yang menandai bahwa file tersebut merupakan hasil dekripsi. File akhir ini kemudian dapat digunakan sebagaimana file asli sebelum proses enkripsi dilakukan.

3.3.2. Evaluasi Sistem

Tahap ini merupakan proses pengujian terhadap sistem yang telah dirancang untuk menjaga kerahasiaan data nasabah perbankan melalui penerapan teknik kriptografi. Pengujian dilakukan untuk menilai kinerja serta ketahanan sistem dalam berbagai kondisi, sekaligus memastikan bahwa algoritma yang diterapkan mampu memberikan perlindungan data secara efektif. Beberapa aspek krusial yang menjadi fokus dalam pengujian mencakup waktu pemrosesan, yaitu sejauh mana sistem dapat menyelesaikan proses enkripsi dan dekripsi secara efisien; konsumsi memori, yang menunjukkan tingkat efisiensi sistem dalam pemanfaatan sumber daya perangkat keras; serta pengujian terhadap serangan Brute Force, yakni upaya simulasi peretasan dengan mencoba seluruh kemungkinan kombinasi kunci secara berurutan. Ketiga pengujian ini bertujuan untuk memberikan evaluasi komprehensif terhadap performa dan tingkat keamanan sistem yang telah dibangun.

a. Hasil Pengujian Waktu Proses Dekripsi

   Pengujian waktu proses pada sistem ini digunakan untuk mengukur seberapa cepat algoritma RSA-CRT dalam melakukan proses dekripsi, serta melakukan perbandingan waktu antara metode RSA biasa dengan RSA-CRT. Pengujian dilakukan terhadap data terenkripsi dalam berbagai ukuran file menggunakan kunci sepanjang 2048-bit, dengan parameter yang diukur meliputi waktu eksekusi dan efisiensi waktu dekripsi dalam bentuk persentase. Proses dekripsi dilakukan secara bertahap berdasarkan pembagian persentase dari total data, dengan skema distribusi progresif sebesar 5%, 10%, 20%, 25%, dan 40%. Pendekatan ini memungkinkan sistem untuk menyimpan hasil dekripsi secara parsial ke dalam berkas Excel setelah setiap tahap selesai, sehingga dapat meminimalkan risiko kehilangan data apabila terjadi interupsi, serta mempermudah monitoring performa dan penggunaan sumber daya selama proses berlangsung. Untuk mengukur efisiensi dari algoritma RSA-CRT terhadap algoritma RSA, digunakan rumus seperti persamaan (1).

$$Efficiency\ Gain\ (\%) = \left(\frac{T_{lama} - T_{baru}}{T_{lama}}\right) \times 100 \qquad (14)$$

   Hasil dari pengujian yang telah dilakukan terhadap 5 file dengan ukuran yang berbeda yakni 6,53MB, 13,1MB, 19,7MB, 26,3MB, dan 33,1MB, didapat rata-rata waktu dari setiap pengujian yang telah dilakukan. Lebih lanjut, rata-rata tersebut dapat dilihat pada Tabel 5.

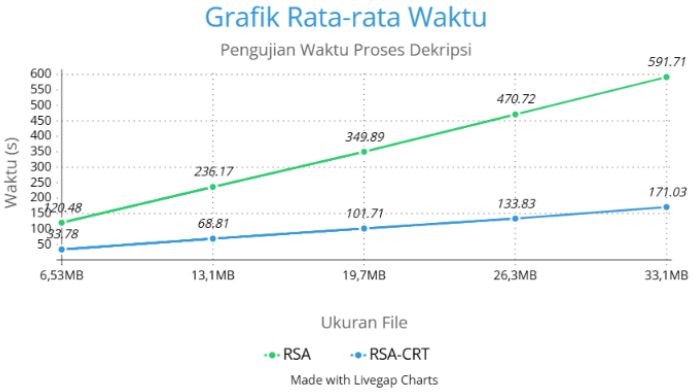
Tabel 5. Rata-rata Hasil Pengujian Waktu Proses Dekripsi

Ukuran File	Rata-rata Waktu Dekripsi RSA	Rata-rata Waktu Dekripsi RSA-CRT	Efficiency Gain
6,53 MB	120.48s	33.78s	71.96%
13,1 MB	236.17s	68.81s	70.85%
19,7 MB	349.89s	101.71s	70.94%

p-ISSN: 2301-5373
e-ISSN: 2654-5101

26,3 MB	470.72s	133.83s	71.57%
33,1 MB	591.71s	171.03s	71.09%

Berdasarkan hasil dari Tabel 5, dapat digambarkan sebuah grafik yang dapat dilihat pada Gambar 14.



Gambar 12. Grafik Rata-rata Pengujian Waktu Proses Dekripsi

Seperti yang ditunjukkan oleh Gambar 12, berdasarkan data rata-rata waktu dekripsi pada lima file dengan ukuran yang berbeda, terlihat tren yang konsisten di mana RSA-CRT secara signifikan mempercepat proses dekripsi dibandingkan RSA standar. Untuk file berukuran 6,53 MB, rata-rata waktu dekripsi menggunakan RSA adalah 120,48 detik, sedangkan RSA-CRT hanya membutuhkan 33,78 detik, menghasilkan efficiency gain sebesar 71,96%. Efisiensi ini tetap tinggi seiring dengan bertambahnya ukuran file. Pada file terbesar, yaitu 33,1 MB, RSA membutuhkan waktu rata-rata 591,71 detik, sementara RSA-CRT hanya memerlukan 171,03 detik, dengan efficiency gain sebesar 71,09%. Secara umum, semua file menunjukkan efficiency gain yang stabil di kisaran 70–72%, membuktikan bahwa RSA-CRT tidak hanya efektif untuk file kecil, tetapi juga sangat efisien dan skalabel untuk file berukuran besar. Hal ini mendukung penggunaan RSA-CRT sebagai metode optimal dalam sistem yang membutuhkan proses dekripsi yang cepat dan efisien terhadap data dalam jumlah besar.

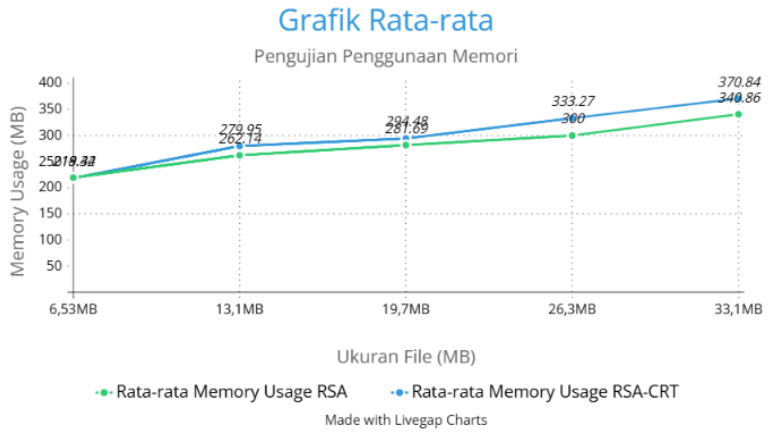
b. Hasil Pengujian Penggunaan Memori (*memory usage*)

Pengujian penggunaan memori pada sistem ini bertujuan untuk mengamati seberapa besar sumber daya memori yang digunakan selama proses dekripsi dengan algoritma RSA dan RSA-CRT. Pengukuran ini penting dilakukan untuk mengevaluasi efisiensi algoritma tidak hanya dari sisi kecepatan, tetapi juga dari aspek konsumsi sumber daya sistem, terutama pada perangkat dengan keterbatasan memori. Pengujian dilakukan terhadap lima file dengan ukuran berbeda, masing-masing dienkripsi menggunakan kunci sepanjang 2048-bit, kemudian didekripsi menggunakan dua pendekatan algoritma. Selama proses dekripsi, sistem mencatat penggunaan memori maksimum pada setiap tahap batch yang dibagi ke dalam skema 5%, 10%, 20%, 25%, dan 40% dari total isi file. Data penggunaan memori dicatat secara otomatis melalui pemantauan sistem, dan hasilnya dikompilasi dalam bentuk tabel serta divisualisasikan dalam grafik untuk memudahkan analisis. Pendekatan ini memungkinkan identifikasi tren penggunaan memori secara progresif seiring bertambahnya ukuran data yang diproses, serta membandingkan efisiensi antara RSA dan RSA-CRT dalam pengelolaan memori. Dengan demikian, dapat diketahui apakah peningkatan kecepatan yang ditawarkan oleh RSA-CRT juga diiringi dengan efisiensi dalam penggunaan memori, atau justru memerlukan sumber daya lebih besar dibanding RSA biasa. Hasil dari pengujian yang telah dilakukan terhadap 5 file dengan ukuran yang berbeda yakni 6,53MB, 13,1MB, 19,7MB, 26,3MB, dan 33,1MB, didapat rata-rata penggunaan memori dari setiap pengujian yang telah dilakukan. Lebih lanjut, rata-rata tersebut dapat dilihat pada Tabel 6.

Tabel 6. Rata-rata Memory Usage

Ukuran File	Rata-rata Memory Usage RSA	Rata-rata Memory Usage RSA-CRT
6,53 MB	219.44MB	218.32MB
13,1 MB	262.14MB	279.95MB
19,7 MB	281.69MB	294.48MB
26,3 MB	300,60MB	333.27MB
33,1 MB	340.86MB	370.84MB

Berdasarkan hasil dari Tabel 6, dapat digambarkan sebuah grafik yang dapat dilihat pada Gambar 15.



Gambar 13. Grafik Rata-rata Pengujian Penggunaan Memori

Grafik pada Gambar 13 menunjukkan rata-rata penggunaan memori dari kedua algoritma menunjukkan tren yang cukup jelas seiring dengan bertambahnya ukuran file yang didekripsi. Pada file terkecil berukuran 6,53 MB, perbedaan penggunaan memori antara RSA (219,44 MB) dan RSA-CRT (218,32 MB) sangat kecil dan hampir tidak signifikan, menunjukkan bahwa untuk file kecil, keduanya memiliki efisiensi memori yang hampir setara. Namun, mulai dari file berukuran 13,1 MB hingga 33,1 MB, RSA-CRT secara konsisten mencatat penggunaan memori yang lebih tinggi dibanding RSA. Sebagai contoh, pada file 13,1 MB, RSA-CRT menggunakan rata-rata 279,95 MB, lebih besar dari RSA yang hanya 262,14 MB. Selisih ini semakin membesar pada file 26,3 MB dan 33,1 MB, dengan RSA-CRT masing-masing mencatat penggunaan memori sebesar 333,27 MB dan 370,84 MB, sementara RSA hanya 300,60 MB dan 340,86 MB. Hal ini mengindikasikan bahwa meskipun RSA-CRT unggul secara signifikan dari segi kecepatan dekripsi, terdapat trade-off dalam bentuk peningkatan konsumsi memori, khususnya pada pengolahan file berukuran besar. Oleh karena itu, dalam penerapannya, RSA-CRT sangat cocok digunakan pada sistem dengan kapasitas memori yang memadai, terutama ketika performa waktu menjadi prioritas utama.

c. Hasil Pengujian Ketahanan *Brute Force*

Pada pengujian ini, dilakukan benchmarking untuk mengukur waktu yang diperlukan untuk melakukan *Brute Force* pada modulus RSA dengan berbagai ukuran kunci (dalam bit). Pengujian dilakukan dengan 5 ukuran kunci berbeda, yakni 32-bit, 34-bit, 36-bit, 40-bit, 44-bit, 48-bit, 52-bit, 56-bit, 60-bit, dan 64-bit. Setiap ukuran kunci diuji menggunakan metode Fermat's Factorization, yang merupakan salah satu teknik faktorisasi yang cukup efisien untuk kunci dengan panjang bit kecil hingga sedang. Hasil pengujian benchmark dapat dilihat pada Tabel 7.

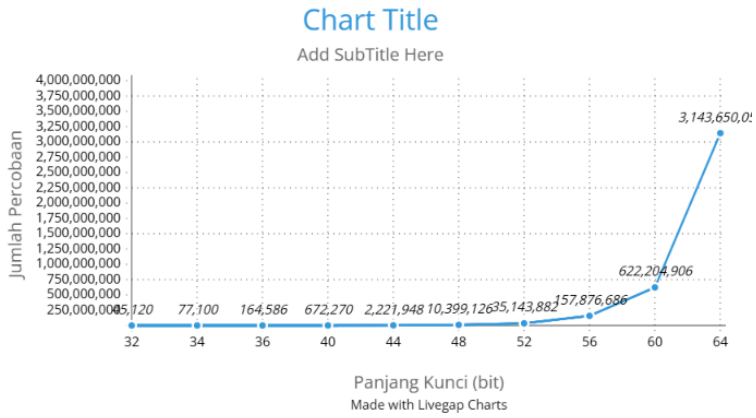
Tabel 7. Hasil Pengujian Simulasi *Brute Force*

Panjang Kunci	Waktu *Brute Force*

p-ISSN: 2301-5373
e-ISSN: 2654-5101

32-bit	45.120
34-bit	77.100
36-bit	164.586
40-bit	672.270
44-bit	2.221.948
48-bit	10.399.126
52-bit	35.143.882
56-bit	157.876.686
60-bit	622.204.906
64-bit	3.143.650.056

Berdasarkan hasil dari Tabel 7, dapat digambarkan sebuah grafik yang dapat dilihat pada Gambar 14.



Gambar 14. Grafik Simulasi Uji *Brute Force*

Berdasarkan hasil pengujian serangan *Brute Force* terhadap algoritma RSA, diperoleh waktu eksekusi yang meningkat secara signifikan seiring bertambahnya panjang kunci RSA, seperti ditunjukkan pada data pengujian dari panjang kunci 32-bit hingga 64-bit. Secara umum, terdapat kecenderungan eksponensial bahwa semakin panjang ukuran kunci, maka semakin besar waktu yang dibutuhkan untuk memfaktorkan nilai modulus n menjadi dua bilangan prima penyusunnya.

Sebagai contoh, pada kunci berukuran 32-bit, proses *Brute Force* hanya memerlukan waktu sekitar 0,01 detik. Namun, ketika panjang kunci meningkat menjadi 48-bit, waktu yang dibutuhkan melonjak menjadi 3,20 detik. Kenaikan waktu ini menjadi semakin drastis pada kunci 56-bit (57,85 detik) dan 64-bit (1304,81 detik), yang menunjukkan bahwa ruang pencarian pasangan bilangan prima p dan q tumbuh secara eksponensial terhadap panjang kunci.

Tidak seperti metode faktorisasi berbasis pola tertentu, pendekatan *Brute Force* yang digunakan dalam penelitian ini tidak mengandalkan karakteristik khusus dari bilangan prima, melainkan memeriksa kemungkinan pasangan faktor satu per satu secara sistematis. Oleh karena itu, hasil waktu yang diperoleh cenderung lebih stabil dan mencerminkan kompl eksitas komputasi yang murni bergantung pada ukuran kunci.

Yang menarik untuk dicermati adalah bahwa waktu *Brute Force* untuk kunci sepanjang 64-bit saja sudah mencapai lebih dari 1300 detik (sekitar 21 menit). Hal ini menunjukkan bahwa meskipun *Brute Force* masih dapat dilakukan pada kunci dengan panjang kecil hingga menengah, metode ini menjadi sangat tidak efisien pada kunci dengan panjang yang lebih besar. Dalam konteks sistem yang dibangun ini, yang menggunakan kunci RSA sepanjang 2048-bit, dapat dibayangkan bahwa waktu *Brute Force* akan meningkat secara eksponensial dan menjadi nyaris mustahil dilakukan dengan sumber daya komputasi biasa.

Dengan demikian, hasil ini memperkuat fakta bahwa RSA dengan panjang kunci 2048-bit menawarkan tingkat keamanan yang sangat tinggi terhadap serangan *Brute Force*,

dan bahwa peningkatan panjang kunci secara langsung berkontribusi signifikan terhadap resistansi algoritma terhadap metode pemecahan sederhana seperti ini.

4. Kesimpulan

Berdasarkan penelitian yang telah dilakukan terkait implementasi dan pengujian algoritma RSA dan RSA-CRT, dapat disimpulkan bahwa:

a. Hasil pengujian menunjukkan bahwa RSA-CRT secara signifikan lebih efisien dibandingkan RSA standar dalam proses dekripsi file, terutama dari segi waktu eksekusi. Pengujian dilakukan pada lima file dengan ukuran berbeda, yaitu 6,53 MB, 13,1 MB, 19,7 MB, 26,3 MB, dan 33,1 MB, menggunakan kunci sepanjang 2048-bit. Rata-rata waktu dekripsi RSA berada pada kisaran 120,48s h ingga 591,71s, sedangkan RSA-CRT hanya memerlukan waktu 33,78s hingga 171,03s, menghasilkan efisiensi waktu rata-rata sekitar 71% di semua ukuran file. Hal ini menunjukkan bahwa penggunaan teknik Chinese Remainder Theorem (CRT) secara konsisten memangkas waktu dekripsi secara substansial, dan cocok digunakan pada sistem dengan kebutuhan performa tinggi.
b. Dari sisi penggunaan memori, hasil pengujian menunjukkan variasi efisiensi yang lebih kompleks. Untuk file kecil (6,53 MB), RSA-CRT menggunakan memori sedikit lebih rendah (rata-rata 218,32 MB) dibanding RSA (rata-rata 219,44 MB), tetapi untuk file yang lebih besar, RSA-CRT cenderung menggunakan memori lebih tinggi, terutama pada file 33,1 MB dengan rata-rata penggunaan sebesar 370,84 MB dibandingkan RSA yang hanya 340,86 MB. Meskipun demikian, peningkatan konsumsi memori ini dapat dianggap sebagai trade-off yang wajar, mengingat peningkatan kecepatan dekripsi yang signifikan. Selain itu, pola memori RSA-CRT cenderung lebih stabil dan tidak fluktuatif secara drastis, yang dapat menguntungkan dalam sistem dengan alokasi memori dinamis.
c. Dari sisi keamanan, hasil pengujian brute force terhadap algoritma RSA menunjukkan bahwa semakin besar ukuran kunci RSA, waktu yang dibutuhkan untuk memfaktorkan modulus n juga meningkat secara signifikan. Peningkatan ini bersifat eksponensial, mencerminkan bahwa ruang pencarian pasangan bilangan prima p dan q tumbuh sangat cepat seiring bertambahnya panjang kunci. Pada pengujian yang dilakukan hingga 64-bit, waktu yang dibutuhkan sudah mencapai lebih dari 1300 detik, dan hal ini memperlihatkan keterbatasan metode brute force dalam menangani kunci dengan panjang besar. Dengan mempertimbangkan bahwa aplikasi ini menggunakan kunci sepanjang 2048-bit, serangan brute force menjadi tidak realistis dilakukan dalam waktu yang wajar. Oleh karena itu, penggunaan kunci RSA dengan panjang yang cukup besar tetap menjadi faktor utama dalam menjaga keamanan sistem dari serangan brute force, meskipun tanpa mempertimbangkan variasi pemilihan bilangan prima p dan q. Dengan implementasi RSA-CRT pada sistem utama menggunakan kunci 2048-bit, serangan Brute Force menjadi sangat tidak praktis, bahkan mustahil dilakukan dalam waktu wajar. Oleh karena itu, RSA-CRT sangat layak digunakan untuk melindungi data sensitif, seperti data nasabah perbankan, karena menawarkan performa dekripsi yang tinggi sekaligus tingkat keamanan yang kuat, selama kapasitas memori sistem dapat mengakomodasi proses tersebut.

Referensi

[1] D. L. Putri and S. Hardiyanto, "Kompas.com," Kompas, 17 11 2022. [Online]. Available: https://www.kompas.com/tren/read/2022/11/17/170500065/indonesia-peringkat-3-kebocoran-data-gara-gara-bjorka-.

[2] R. Imam, M. Areeb, A. S. Alturki and F. Anwer, "Systematic and Critical Review of RSA Based Public Key Cryptographic Schemes: Past and Present Status.," 2021.

[3] M. Campercholi, G. Zigarán and G. Zigarán, "The complexity of the Chinese Remainder Theorem," *,* 2023.

[4] aryasaktiwirasena, "Fungsi Totient Euler (Euler's Totient Function)," Universitas Gadjah

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Mada, 10 January 2021. [Online]. Available: https://aryasaktiwirasena.web.ugm.ac.id/2021/01/10/fungsi-totient-euler-eulers-totient-function/. [Accessed 23 May 2025].

[5] M. Saha, M. T. Basu, A. Gupta, K. Ashrith, C. V. V. Reddy, S. Reddy and R. Reddy, "Enhancing credit card security using RSA encryption and tokenization: a multi-module approach," *International Journal of Informatics and Communication Technology (IJ-ICT),* vol. 14, 2025.

[6] U. Erdiansyah, M. K. M. Nasution and S. Siregar, "Hybrid cryptosystem multi-power RSA with N=P m Q and VMPC," *IOP Conference Series Materials Science and Engineering,* 2020.

[7] W. D. Hartama, I. O. Kirana, S. and I. Gunawan, "Implementasi Algoritma Kriptografi Rivest Shamir Adlemen untuk Mengamankan Data Ijazah pada SMK Swasta Prama Artha Kab. Simalungun," *Jurnal Ilmu Komputer dan Informatika (JIKI),* vol. 2, 2022.

[8] S. and A. Prihanto, "Perbandingan Efisiensi Algoritma RSA dan RSA-CRT Dengan Data Teks Berukuran Besar," *Journal of Informatics and Computer Science (JINACS),* vol. 01, 2019.

[9] Z. Panjaitan, K. Ibnutama and M. G. Suryanata, "Penggunaan Chinese Reminder Theorem (CRT) pada Algoritma RSA," *Sains dan Komputer (SAINTIKOM),* vol. 18, 2019.

[10] M. F. Rafli, Z. Panjaitan and W. Riansah, "Aplikasi Keamanan Sistem Pengiriman Tagihan Pembayaran Online (Invoice) Berbasis Website dengan algoritma RSA-CRT," *SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer),* vol. 23, 2024.

[11] N. C. H. Wibowo, K. Umam, A. M. I. Khaq and F. A. Rizki, "Komparasi Waktu Algoritma RSA dengan RSA-CRT Base On Computer," *Walisongo Journal of Information Technology,* vol. 2, 2020.

[12] S. A. Christie, "Literature Review : Identifikasi Penggunaan Teknik dan Analisis Requirement Engineering," 2022.

This page is intentionally left blank.