

**PERBANDINGAN ALGORITMA *BRUTE FORCE* DAN *KRAITCHIK* DALAM  
KRIPTANALISIS ALGORITMA KUNCI PUBLIK *ENHANCED AA<sub>β</sub>***

**SKRIPSI**

**AKHDAN FADHILAH**

**211401094**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2025**

**PERBANDINGAN ALGORITMA *BRUTE FORCE* DAN *KRAITCHIK* DALAM  
KRIPTANALISIS ALGORITMA KUNCI PUBLIK *ENHANCED AA<sub>β</sub>***

**SKRIPSI**

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah  
Sarjana Komputer

AKHDAN FADHILAH

211401094



PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2025

**PERSETUJUAN**

Judul : PERBANDINGAN ALGORITMA *BRUTE FORCE* DAN  
*KRAITCHIK* DALAM KRIPTANALISIS ALGORITMA  
KUNCI PUBLIK *ENHANCED AA <sub>$\beta$</sub>*

Kategori : SKRIPSI

Nama : AKHDAN FADHILAH

Nomor Induk Mahasiswa : 211401094

: SARJANA (S-1) ILMU KOMPUTER


: ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

Tanggal Sidang : 9 JANUARI 2025


Komisi Pembimbing :

Pembimbing II

Pembimbing I



Dr. Ir. Elviawaty Muisa Zamzami  
S.T., M.T., M.M., IPU  
NIP. 197812212014042001



Dian Rachmawati S.Si., M.Kom.  
NIP. 198307232009122004

Diketahui/Disetujui Oleh  
Program Studi Ilmu Komputer  
Ketua



Dr. Amalia, S.T., M.T.  
NIP. 197812212014042001

**PERNYATAAN**

PERBANDINGAN ALGORITMA *BRUTE FORCE* DAN *KRAITCHIK* DALAM  
KRIPTANALISIS ALGORITMA KUNCI PUBLIK *ENHANCED  $AA_\beta$*

**SKRIPSI**

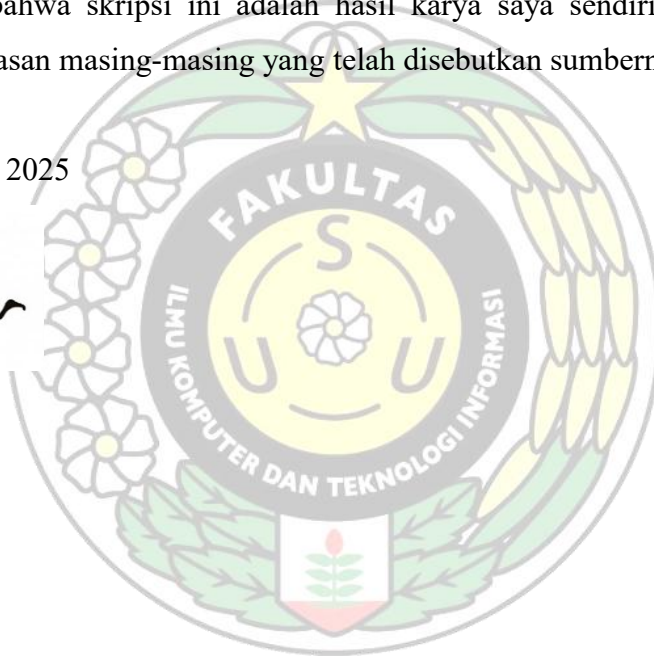
Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan masing-masing yang telah disebutkan sumbernya.

Medan, 9 Januari 2025



Akhdan Fadhillah

211401094



## PENGHARGAAN

*Bismillahirrahmanirrahim*, segala puji dan syukur senantiasa dipanjatkan kepada Allah *Subhanahu Wa Ta'ala* atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat sampai pada tahap penyusunan skripsi ini sebagai bagian dari persyaratan untuk meraih gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, USU. Shalawat dan salam juga tak lupa limpahkan kepada junjungan Nabi Muhammad *Shalallaahu 'Alayhi Wasallam*, yang telah membawa umat manusia dari masa kegelapan menuju era penuh cahaya seperti sekarang ini.

Selama penulis menyusun skripsi ini, tentunya tidak lepas dari bimbingan dan dukungan berbagai pihak. Oleh sebab itu, penulis ingin menyampaikan tanda terima kasih kepada:

1. Bapak Prof. Dr. Muryanto Amin S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yang telah memberikan banyak dukungan kepada penulis sepanjang masa studi.
3. Bapak Dr. Mohammad Andri Budiman S.T., M.Comp.Sc., M.E.M. selaku Wakil Dekan I Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara sekaligus Dosen Penguji yang telah memberikan banyak masukan, motivasi, serta dukungan kepada penulis sepanjang masa studi.
4. Ibu Dr. Amalia, S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yang telah telah mendedikasikan waktu untuk memberikan masukan, motivasi, serta dukungan kepada penulis sepanjang masa studi.
5. Ibu Sri Melvani Hardi, S.Kom., M.Kom. selaku Sekretaris Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yang telah mendedikasikan waktu untuk memberikan banyak masukan, motivasi, serta dukungan kepada penulis sepanjang masa studi.
6. Ibu Dian Rachmawati S.Si., M.Kom. selaku Dosen Pembimbing I yang senantiasa meluangkan waktu untuk memberikan bimbingan dan saran kepada penulis dalam proses penyelesaian skripsi ini.

7. Ibu Dr. Ir. Elviawaty Muisa Zamzami S.T., M.T., M.M., IPU. selaku Dosen Pembimbing II senantiasa meluangkan waktu untuk memberikan bimbingan dan saran kepada penulis dalam proses penyelesaian skripsi ini.
8. Ibu Anandhini Medianty Nababan S.Kom., M.T selaku Dosen Penguji yang telah memberikan masukan dan saran yang membangun kepada penulis.
9. Bapak Prof. Dr. Syahril Efendi S.Si., M.I.T. selaku Dosen Pembimbing Akademik yang telah membimbing dan mengarahkan penulis dalam masa studi.
10. Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer, yang telah membimbing penulis sepanjang masa studi.
11. Seluruh Staff, Pegawai, Satpam, dan CS Program Studi S-1 Ilmu Komputer, yang telah memberikan dukungan dan kenangan sepanjang masa studi.
12. Teristimewa kepada Orang Tua terkasih, tersayang penulis Mama Sandiyah, atas segala bentuk pengorbanan, kasih sayang, serta doa-doa tulus yang senantiasa dipanjatkan untuk penulis. Ucapan terima kasih juga penulis sampaikan kepada Papa, Alm. Budi Asprianto, yang telah memberikan cinta dan dukungan semasa hidupnya, serta kepada Papa Yushar Efendi, ayah sambung penulis, atas kasih sayang dan dukungannya yang selalu menemani di setiap langkah perjalanan ini.
13. Abang kandung tersayang Asyraful Rizal serta adik kandung Adri Maulana dan Arbiansyah Habibi yang selalu mendukung dan mendoakan penulis.
14. Kepada seluruh anggota keluarga besar Alm. H. Ganie serta Alm. Juned & Saniah, khususnya Mamaji, Alm Wak E, Wak Umi, Wak Sum, Wak Bas, Pakde Udin, Nyaknis, Pakboy, Pakde Tamam, Ayahwa Marun, Ayahwa Abidin, Kak Maya, Kak Nunung, Bang Salman, Bang Kiki, Cut nyak, dan Iqbal penulis menyampaikan rasa terima kasih yang mendalam atas dukungan moral dan material yang telah diberikan. Dukungan yang tak ternilai ini telah menjadi sumber motivasi bagi penulis untuk menuntaskan studi, dengan harapan dapat mengangkat derajat dan membawa pencerahan bagi keluarga.
15. Pengurus IMILKOM Tahun 2023/2024, terkhusus BPH yaitu Belvin, Fransisca, Presidium Khususnya Adi, Nurul, Poji, Riji, Saffa, Shinta, Stephen, Addina, Elin, Haikal, Muthia, Nadya, Sea, Ichak, Yusuf, Ivanny, dan Elizabeth



yang telah banyak membantu penulis melalui berbagai aktivitas dan bekerja sama dengan baik selama menjalankan satu periode kepengurusan.

16. Kepada Dewan Penasehat IMILKOM Tahun 2023/2024 yaitu, Bang Zaman, Bang Yudha, Bang Aduy, Bang Kevin, dan Kak Saras yang telah memberikan memberikan motivasi dan mendukung selama masa studi.
17. Sahabat penulis Alex, Angela, Andhika, Dea, Dian, Ghalib, Higen, Hanafi, Hiskia, Karin, Kimsang, Krisna, Naftaly, Miranda, Nayata, Naufal, Noel, Nicho, Putri, Popot, Rizky Maulana, Revi, Reysa, Riyan, Samuel, Said, Yusuf, Yakin, Zaki, Zidane, dan Zwingli serta seluruh sahabat stambuk 2021 dan ‘dak dak futsal’, serta sahabat yang membersamai selama masa studi yang penuh kegembiraan dan kenangan. Terima kasih atas semua pembelajaran dan waktu berharga yang telah diberikan, baik dalam bidang akademis maupun sebagai bekal untuk menjalani kehidupan di dunia dan akhirat.
18. Abang-kakak senior terkhusus stambuk 2020 dan 2019 yang telah memberi banyak motivasi, masukan, dan doa kepada penulis selama masa studi.
19. Untuk adik-adik dari stambuk 2022 dan 2023 yang telah berkontribusi tenaga dalam kerja sama selama kegiatan kepanitiaan dan perkuliahan, serta memberikan doa dan dukungan yang sangat berarti bagi penulis.
20. Kepada Kantor Xtend Indonesia terkhusus kepada Bapak Alfian, Ibu Devi, Bang Farhan, Bang Khairul, Bang Faiz, Kak Fitri, Bang wawan, Bang Andra, Bang Rizky serta Abang dan Kakak yang memberikan semangat.
21. Dan untuk setiap orang yang mengenal penulis, setiap orang yang menyebutkan nama penulis dalam doa-doanya, Terimakasih yang sebesar-besarnya.

Sebagai penutup, penulis berharap semoga skripsi ini dapat memberikan manfaat bagi para pembaca. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah memberikan bantuan dalam proses penyelesaian skripsi ini, serta memohon maaf atas segala kekurangan dalam penulisannya, kepada Allah SWT saya mohon ampun.

Medan, 9 Januari 2025

Penulis,



Akhdan Fadhillah

## ABSTRAK

Penelitian ini membahas perbandingan algoritma *Brute Force* dan *Kraitchik* dalam kriptanalisis algoritma kunci publik *Enhanced  $AA_\beta$* , yang diimplementasikan dan diuji pada perangkat berbasis Android menggunakan bahasa pemrograman Dart pada platform Flutter. Hasil analisis menunjukkan bahwa kedua algoritma mampu memfaktorkan kunci publik  $A_1$  dan  $A_2$  untuk menghasilkan faktor  $p$ ,  $q$ , dan  $d$ . Namun, algoritma *Kraitchik* menunjukkan efisiensi yang lebih rendah dibandingkan *Brute Force* pada panjang kunci  $k$  yang besar. *Kraitchik* menunjukkan performa yang lebih cepat dalam memfaktorkan  $A_2$  ketika nilai  $p$  lebih kecil dari  $q$ , terutama jika jarak antara  $p$  dan  $q$  berdekatan. Hal ini disebabkan oleh perbedaan kuadrat ( $x^2 - kn$ ) yang lebih kecil, sehingga mempercepat proses faktorisasi. Namun, algoritma *Kraitchik* mengalami peningkatan waktu eksekusi yang lebih signifikan dibandingkan *Brute Force*, terutama untuk  $k \geq 9$ . Pada panjang kunci yang lebih kecil ( $k = 3$  hingga  $k=5$ ), kedua algoritma menunjukkan waktu eksekusi yang relatif cepat, tetapi peningkatan panjang kunci membuat *Kraitchik* menjadi kurang efisien dibandingkan *Brute Force*.

**Kata Kunci:** Kriptografi, *Enhanced  $AA_\beta$* , *Brute Force*, *Kraitchik*, Kriptanalisis, Keamanan Data.



**COMPARISON OF THE BRUTE FORCE AND KRAITCHIK ALGORITHMS IN  
THE CRYPTANALYSIS OF THE ENHANCED  $AA_\beta$  PUBLIC KEY ALGORITHM**

**ABSTRACT**

*This study examines the comparison between the Brute Force and Kraitchik algorithms in the cryptanalysis of the Enhanced  $AA_\beta$  public key algorithm, implemented and tested on Android devices using the Dart programming language on the Flutter platform. The analysis reveals that both algorithms can factorize the public keys  $A_1$  and  $A_2$  to produce the factors  $p$ ,  $q$ , and  $d$ . However, the Kraitchik algorithm demonstrates lower efficiency than Brute Force for larger key lengths  $k$ . Kraitchik performs faster in factorizing  $A_2$  when  $p$  is smaller than  $q$ , especially when  $p$  and  $q$  are close in value. This efficiency is due to the smaller quadratic difference ( $x^2 - kn$ ), which accelerates the factorization process. Nevertheless, Kraitchik experiences a sharper increase in execution time compared to Brute Force, particularly for  $k \geq 9$ . For smaller key lengths ( $k=3$ ), both algorithms exhibit relatively fast execution times. However, as key length increases, Kraitchik becomes less efficient compared to Brute Force.*

**Keywords:** *Cryptography, Enhanced  $AA_\beta$ , Brute Force, Kraitchik, Cryptanalysis, Data Security.*

## DAFTAR ISI

<b>PERNYATAAN .....</b>	<b>iii</b>
<b>PENGHARGAAN .....</b>	<b>iv</b>
<b>ABSTRAK .....</b>	<b>vi</b>
<b>DAFTAR ISI .....</b>	<b>ix</b>
<b>DAFTAR GAMBAR .....</b>	<b>xi</b>
<b>DAFTAR TABEL .....</b>	<b>xii</b>
<b>BAB I PENDAHULUAN .....</b>	<b>1</b>
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	3
1.3. Batasan Masalah .....	3
1.4. Tujuan Penelitian .....	4
1.5. Manfaat Penelitian .....	4
1.6. Metodologi Penelitian .....	5
1.7. Sistematika Penulisan .....	6
<b>BAB II LANDASAN TEORI .....</b>	<b>8</b>
2.1. Kriptografi .....	8
2.1.1. GCD ( <i>Greatest Common Divisor</i> ) .....	9
2.1.2. Modulo Eksponensial .....	10
2.1.3. Invers Modulo .....	12
2.1.4. Pembangkit Bilangan Prima .....	14
2.2. Kriptografi Simetris .....	16
2.3. Kriptografi Asimetris .....	17
2.4. File Teks .....	17
2.5. Kompleksitas Algoritma .....	18
2.6. Kriptanalisis .....	19
2.7. Kunci Publik <i>Enhanced AA<sub><math>\beta</math></sub></i> .....	19
2.8. Algoritma <i>Brute Force</i> .....	21
2.9. Algoritma <i>Kraitchik</i> .....	23
<b>BAB III ANALISIS DAN PERANCANGAN SISTEM .....</b>	<b>25</b>
3.1. Analisis .....	25

3.1.1.	Analisis Masalah .....	25
3.1.2.	Analisis Sistem .....	26
3.2.	Perancangan Sistem .....	28
3.2.1.	Diagram Umum Pembangkitan Kunci .....	28
3.2.2.	Diagram Umum Kriptanalisis .....	29
3.2.3.	<i>Use Case Diagram</i> .....	30
3.2.4.	<i>Activity Diagram</i> .....	31
3.2.5.	<i>Sequence Diagram</i> .....	36
3.2.6.	<i>Flowchart</i> (Diagram Alir) .....	37
3.2.7.	Perancangan <i>User Interface</i> .....	42
<b>BAB IV IMPLEMENTASI DAN PENGUJIAN .....</b>		<b>46</b>
4.1.	Implementasi .....	46
4.1.1.	Halaman Beranda .....	46
4.1.2.	Halaman Enkripsi .....	46
4.1.3.	Halaman Dekripsi .....	47
4.1.4.	Halaman Kriptanalisis .....	47
4.2.	Pengujian .....	48
4.2.1.	Pengujian Pembangkitan Kunci Publik dan Enkripsi .....	48
4.2.2.	Pengujian Dekripsi Kunci Publik dan Privat .....	49
4.2.3.	Pengujian Pemfaktoran Kunci Publik .....	49
4.2.4.	Kompleksitas Algoritma .....	55
<b>BAB V KESIMPULAN DAN SARAN .....</b>		<b>60</b>
5.1.	Kesimpulan .....	60
5.2.	Saran .....	61
<b>DAFTAR PUSTAKA .....</b>		<b>62</b>

## DAFTAR GAMBAR

<b>Gambar 2.1</b> Algoritma Simetris .....	16
<b>Gambar 2.2</b> Algoritma Asimetris .....	17
<b>Gambar 3.1</b> Diagram Ishikawa Masalah Penelitian .....	26
<b>Gambar 3.2</b> Diagram Umum Pembangkitan Kunci .....	29
<b>Gambar 3.3</b> Diagram Umum Kriptanalisis .....	29
<b>Gambar 3.4</b> Use Case Diagram .....	31
<b>Gambar 3.5</b> Activity Diagram Pembangkit Kunci dan Enkripsi .....	32
<b>Gambar 3.6</b> Activity Diagram Dekripsi .....	34
<b>Gambar 3.7</b> Activity Diagram Kriptanalisis .....	35
<b>Gambar 3.8</b> <i>Sequence</i> Diagram Pembangkitan Kunci .....	37
<b>Gambar 3.9</b> <i>Sequence</i> Diagram Kriptanalisis .....	37
<b>Gambar 3.10</b> Flowchart Key Generation .....	38
<b>Gambar 3.11</b> Flowchart Enkripsi .....	39
<b>Gambar 3.12</b> Flowchart Dekripsi .....	40
<b>Gambar 3.13</b> Flowchart Kriptanalisis .....	41
<b>Gambar 3.14</b> Rancangan Halaman Beranda .....	42
<b>Gambar 3.15</b> Rancangan Halaman Enkripsi .....	43
<b>Gambar 3.16</b> Rancangan Halaman Dekripsi .....	44
<b>Gambar 3.17</b> Rancangan Halaman Kriptanalisis .....	45
<b>Gambar 4.1</b> Halaman Beranda .....	46
<b>Gambar 4.2</b> Halaman Enkripsi .....	47
<b>Gambar 4.3</b> Halaman Dekripsi .....	47
<b>Gambar 4.4</b> Halaman Kriptanalisis .....	48
<b>Gambar 4.5</b> Pengujian Pembangkitan Kunci Publik .....	49
<b>Gambar 4.6</b> Pengujian Dekripsi Kunci Publik dan Privat .....	49
<b>Gambar 4.7</b> Pengujian Pemfaktoran Kunci Publik .....	50
<b>Gambar 4.8</b> Grafik Waktu Eksekusi Program .....	51
<b>Gambar 4.9</b> Grafik Perbandingan $p$ dan $q$ dengan <i>Kraitichik</i> .....	54

## DAFTAR TABEL

<b>Tabel 4.1</b> Tabel Perbandingan Waktu Eksekusi .....	50
<b>Tabel 4.2</b> Perbandingan $p$ dan $q$ pada Algoritma <i>Kraitichik</i> .....	53
<b>Tabel 4.3</b> Kompleksitas Pembangkitan Kunci .....	55
<b>Tabel 4.4</b> Kompleksitas Enkripsi .....	56
<b>Tabel 4.5</b> Kompleksitas Dekripsi .....	57
<b>Tabel 4.6</b> Kompleksitas <i>Brute Force</i> .....	58
<b>Tabel 4.7</b> Kompleksitas <i>Kraitichik</i> .....	58



## BAB I PENDAHULUAN

### 1.1. Latar Belakang

Dalam era digital yang semakin terintegrasi dengan perangkat *mobile*, perlindungan data sensitif dari akses tidak sah menjadi prioritas utama bagi berbagai sektor, termasuk pemerintahan, perusahaan, dan individu. Perangkat *mobile*, yang digunakan secara luas untuk transaksi dan komunikasi, menghadapi risiko keamanan yang signifikan karena konektivitasnya yang terus-menerus dengan jaringan (Ghafar & Ariffin, 2014). Oleh karena itu, kebutuhan akan algoritma kriptografi yang kuat dan efisien sangat penting untuk menjaga kerahasiaan dan integritas data (Asbullah & Ariffin, 2016).

Salah satu algoritma yang dikembangkan untuk memenuhi kebutuhan ini adalah *Enhanced AA<sub>β</sub>*. Algoritma ini merupakan pengembangan dari kriptosistem *AA<sub>β</sub>* yang sebelumnya telah digunakan dalam berbagai aplikasi kriptografi. Penelitian oleh Asbullah et al. (2018) menunjukkan bahwa algoritma *Enhanced AA<sub>β</sub>* memiliki desain yang cepat dan efisien, serta mampu menangani volume data yang besar dengan lebih optimal dibandingkan algoritma sebelumnya. Selain itu, *Enhanced AA<sub>β</sub>* dirancang untuk mengurangi kelemahan yang ditemukan pada versi *AA<sub>β</sub>* terdahulu terhadap serangan kriptanalisis, seperti serangan timing (Ghafar & Ariffin, 2014). Meskipun demikian, masih terdapat tantangan terkait keamanan algoritma ini, terutama dalam menghadapi serangan kriptanalisis yang lebih kompleks (Mahad et al., 2019).

Algoritma *Brute Force* juga merupakan pendekatan penting dalam menguji ketahanan algoritma kriptografi. Metode ini beroperasi dengan mencoba semua kemungkinan kombinasi kunci hingga kunci yang benar ditemukan (Rechberger, 2013). Kelebihan utama metode ini adalah kepastian bahwa kunci yang benar pasti akan ditemukan jika diberikan waktu dan sumber daya yang cukup (Irawan & Pratama, 2020). Namun, kekurangan dari metode ini adalah kecepatan prosesnya yang lambat dan tidak efisien, terutama pada sistem dengan kunci yang panjang. Seperti yang dijelaskan dalam penelitian, "Serangan *Brute Force* adalah salah satu cara praktis untuk memecahkan teknik kriptografi dengan mencoba setiap kemungkinan kunci" (Talenta Conference Series, 2019).



Meski lambat, metode ini memberikan wawasan penting tentang batas kekuatan kunci yang digunakan dalam suatu algoritma (Gohr, 2022). Dalam konteks ini, (Patterson & Henessy 2013) menyebutkan bahwa meskipun serangan *Brute Force* terlihat sederhana, penting untuk mempertimbangkan batasan waktu dan sumber daya yang dibutuhkan untuk menjalankannya, yang dapat mempengaruhi efektivitas serangan terhadap algoritma kriptografi.

Algoritma *Kraitchik* merupakan metode kriptanalisis yang difokuskan pada pemecahan masalah faktorisasi bilangan bulat, terutama dalam sistem kunci publik seperti RSA. Algoritma ini dirancang untuk menangani modulus kecil, sehingga memungkinkan penyerang mengekstrak kunci privat lebih cepat dibandingkan metode lain (Muchlis, Budiman, & Rachmawati, 2017). Keunggulan utama *Kraitchik* adalah kecepatannya dan efisiensinya dalam menyerang algoritma dengan modulus yang kecil, menjadikannya efektif dalam situasi tertentu. Namun, kelemahan algoritma ini terletak pada kurangnya efektivitas dalam menangani modulus yang lebih besar, sehingga efektivitasnya bergantung pada ukuran modulus yang digunakan (Lydia, Budiman, & Rachmawati, 2021). Memahami batasan ini sangat penting dalam mengamankan algoritma *Enhanced AA <sub>$\beta$</sub>*  (Mahad, Asbullah, & Kamel, 2019). Algoritma *Enhanced AA <sub>$\beta$</sub>* , yang merupakan pengembangan dari kriptografi modern, menawarkan efisiensi dan keamanan yang lebih tinggi dibandingkan algoritma sebelumnya. Namun, ia tetap memiliki kerentanan terhadap serangan kriptanalisis yang lebih canggih, seperti yang terlihat dalam analisis perbandingan (Mahad et al., 2019).

Dalam literatur, berbagai pendekatan hibrida telah diajukan untuk meningkatkan keamanan kriptografi. Budiman dan Irvida (2023), misalnya, menggabungkan sistem kunci publik Rabin-p dengan algoritma Spritz dalam sebuah sistem kriptografi hibrida untuk memperkuat perlindungan data. Pendekatan serupa dapat diterapkan pada *Enhanced AA <sub>$\beta$</sub>* , dengan tujuan mengurangi kerentanannya terhadap serangan kriptanalisis yang lebih kompleks (Mahad et al., 2019).

Penelitian ini bertujuan untuk menganalisis kelemahan dari algoritma *Enhanced AA <sub>$\beta$</sub>*  dengan menggunakan pendekatan kriptanalisis, yaitu algoritma

*Brute Force* dan *Kraitichik*. Hasil dari penelitian ini diharapkan dapat mengevaluasi efektivitas algoritma *Enhanced AA<sub>β</sub>* serta mengidentifikasi potensi perbaikan yang dapat diterapkan untuk meningkatkan keamanan kriptografi secara keseluruhan. Analisis ini tidak hanya penting untuk memahami kelemahan yang ada, tetapi juga sebagai panduan dalam pengembangan algoritma kriptografi yang lebih tahan terhadap serangan di masa depan.

## 1.2. Rumusan Masalah

Berdasarkan latar belakang diatas, seiring dengan semakin maraknya ancaman serangan siber yang terus berkembang, berbagai algoritma kunci publik kini menghadapi tantangan besar dalam menjaga keamanan informasi. Algoritma kunci publik seperti Algoritma Rabin, meskipun awalnya dianggap aman, mulai menunjukkan kelemahan ketika diuji dengan teknik kriptanalisis yang lebih canggih. Algoritma *AA<sub>β</sub>* hadir sebagai perbaikan dari Rabin, namun masih belum sepenuhnya mampu mengatasi potensi celah keamanan. Untuk menghadapi ancaman yang semakin kompleks, *Enhanced AA<sub>β</sub>* dirancang sebagai penyempurnaan lebih lanjut. Namun, algoritma ini juga tidak kebal terhadap serangan, terutama dengan adanya kemajuan teknologi modern. Oleh karena itu, diperlukan evaluasi mendalam menggunakan algoritma *Brute Force* dan *Kraitichik* untuk mengungkap kelemahan yang tersisa dan mengusulkan perbaikan agar sistem keamanan kriptografis dapat lebih tangguh di masa depan.

## 1.3. Batasan Masalah

Dalam melakukan penelitian, peneliti membatasi masalah yang akan diteliti. Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini hanya berfokus pada analisis kerentanan algoritma kunci publik *Enhanced AA<sub>β</sub>* menggunakan algoritma *Brute Force* dan *Kraitichik*, tidak mencakup algoritma atau metode lain di luar ruang lingkup ini.
2. Hanya serangan kriptanalisis terhadap kunci publik yang diuji dalam penelitian ini, tidak termasuk serangan lain seperti serangan pada implementasi atau saluran samping (*side-channel attacks*).
3. Penelitian ini membandingkan waktu eksekusi dan kompleksitas dari kedua metode kriptanalisis (*Brute Force* dan *Kraitichik*) dalam upaya

mendekripsi *ciphertext*, untuk menilai efektivitas masing-masing algoritma.

4. Pengujian kriptanalisis dilakukan pada skenario simulasi dan contoh-contoh kunci publik *Enhanced AA <sub>$\beta$</sub>*  yang digunakan dalam literatur penelitian, bukan pada data atau sistem kriptografi aktual di lapangan.
5. Implementasi kriptanalisis dilakukan menggunakan bahasa pemrograman Dart dalam *platform* Flutter, terbatas pada pengembangan aplikasi berbasis *mobile*.

#### 1.4. Tujuan Penelitian

Penelitian ini bertujuan untuk mengidentifikasi kelemahan algoritma kunci publik *Enhanced AA <sub>$\beta$</sub>*  melalui pendekatan kriptanalisis berbasis *mobile*, menggunakan algoritma *Brute Force* dan *Kraitchik*. Dengan memanfaatkan *platform* pengembangan aplikasi *mobile*, penelitian ini akan mengevaluasi celah keamanan yang berpotensi dimanfaatkan oleh penyerang, serta seberapa efektif algoritma tersebut dalam melindungi data dari serangan berbasis *Brute Force* dan *Kraitchik*. Melalui evaluasi ini, diharapkan dapat ditemukan cara-cara untuk memperkuat keamanan algoritma *Enhanced AA <sub>$\beta$</sub>*  agar lebih tahan terhadap serangan yang dilakukan pada perangkat *mobile*. Hasil akhir dari penelitian ini diharapkan dapat memberikan wawasan lebih dalam mengenai efektivitas algoritma *Enhanced AA <sub>$\beta$</sub>*  dalam skenario *mobile*, serta menjadi fondasi untuk mengembangkan algoritma yang lebih aman dan sesuai untuk perlindungan data di era perangkat *mobile*.

#### 1.5. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah memberikan pemahaman yang lebih mendalam mengenai kelemahan algoritma kunci publik *Enhanced AA <sub>$\beta$</sub>*  dalam konteks *platform mobile*, melalui analisis kriptanalisis algoritma *Brute Force* dan *Kraitchik*. Penelitian ini diharapkan dapat memperkuat keamanan data yang diakses melalui perangkat *mobile*, baik untuk pemerintah, perusahaan, maupun individu dengan mengidentifikasi potensi perbaikan pada algoritma *Enhanced AA <sub>$\beta$</sub>*  agar lebih tangguh terhadap serangan. Selain itu, penelitian ini juga diharapkan berkontribusi untuk pengembangan algoritma

kriptografi yang lebih kuat dan efektif pada perangkat *mobile* serta meningkatkan perlindungan data dan privasi dalam komunikasi digital di masa depan.

## 1.6. Metodologi Penelitian

Metodologi penelitian yang digunakan adalah sebagai berikut:

### 1. Studi Pustaka

Pada tahap ini, peneliti akan mengumpulkan referensi yang relevan mengenai algoritma *Enhanced AA<sub>β</sub>*, algoritma *Brute Force*, dan *Kraitichik* dari berbagai sumber, termasuk jurnal ilmiah, artikel konferensi, dan publikasi terkait. Studi ini akan membantu memahami teori dasar, prinsip kerja, dan kerentanan algoritma *Enhanced AA<sub>β</sub>*, serta bagaimana algoritma *Brute Force* dan *Kraitichik* dapat digunakan untuk menganalisis kelemahan tersebut.

### 2. Analisis dan Perancangan Sistem

Pada tahap ini, peneliti akan menganalisis kelemahan dari algoritma kunci publik *Enhanced AA<sub>β</sub>* menggunakan pendekatan kriptanalisis dengan algoritma *Brute Force* dan *Kraitichik*. Penelitian ini bertujuan untuk memahami seberapa rentan algoritma tersebut terhadap serangan dan merancang sistem yang dapat mengidentifikasi kelemahan tersebut. Peneliti akan membuat diagram alir (*flowchart*) untuk merencanakan langkah-langkah dalam analisis dan perancangan sistem.

### 3. Implementasi Sistem

Pada Pada tahap ini, peneliti akan mengimplementasikan analisis kriptanalisis dengan membuat program berbasis algoritma *Brute Force* dan *Kraitichik*. Program ini akan dirancang untuk menguji keefektifan kedua algoritma dalam mengevaluasi kekuatan algoritma *Enhanced AA<sub>β</sub>*. Pengembangan akan dilakukan menggunakan bahasa pemrograman Dart dalam *platform* Flutter, sehingga dapat diterapkan dalam konteks aplikasi *mobile*.

### 4. Pengujian Sistem

Pada tahap ini, peneliti akan melakukan pengujian untuk mengevaluasi efektivitas algoritma *Brute Force* dan *Kraitichik* dalam mengidentifikasi

kelemahan algoritma *Enhanced AA<sub>β</sub>*. Uji coba ini akan mencakup berbagai skenario untuk mengukur seberapa baik algoritma tersebut dapat melindungi data dan seberapa efisien dalam menghadapi serangan. Hasil dari pengujian akan dibandingkan dengan penelitian sebelumnya untuk mengevaluasi peningkatan keamanan yang dicapai.

#### 5. Dokumentasi

Pada tahap ini, peneliti akan melakukan dokumentasi pada setiap tahap yang terjadi dalam penelitian dan membuat kesimpulan akhir dalam bentuk skripsi.

### 1.7. Sistematika Penulisan

Adapun sistematika penulisan dari penelitian ini terdiri atas:

#### **BAB 1 PENDAHULUAN**

Bagian pendahuluan ini terdiri atas latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

#### **BAB 2 LANDASAN TEORI**

Bagian ini membahas dasar-dasar teori yang menjadi acuan dalam penelitian ini, mencakup konsep yang berkaitan dengan analisis kriptanalisis pada algoritma kunci publik *Enhanced AA<sub>β</sub>* menggunakan algoritma *Brute Force* dan *Kraitchik*. Teori ini mencakup pengenalan dasar tentang algoritma kriptografi, konsep serangan kriptanalisis, serta penjelasan mendalam mengenai kedua algoritma yang digunakan dalam penelitian ini, yaitu *Brute Force* dan *Kraitchik*.

#### **BAB 3 ANALISIS DAN PERANCANGAN**

Bagian ini berisi tentang analisis untuk menyelesaikan masalah menggunakan algoritma yang telah ditentukan dan perancangan terhadap sistem yang akan dibuat.

#### **BAB 4 IMPLEMENTASI DAN PENGUJIAN**



Bagian ini berisi tentang mengimplementasikan rancangan yang telah dibuat serta melakukan pengujian terhadap sistem berdasarkan hasil dari analisis dan perancangan.

## **BAB 5 KESIMPULAN DAN SARAN**

Bagian ini berisi mengenai ringkasan penelitian yang telah dilakukan dan masukan dari penulis yang membangun untuk penelitian selanjutnya





## BAB II

### LANDASAN TEORI

#### 2.1. Kriptografi

Kriptografi adalah disiplin ilmu yang mempelajari teknik pengamanan informasi agar hanya dapat diakses oleh pihak yang berwenang. Tujuan utama dari kriptografi adalah untuk memastikan kerahasiaan, integritas, dan keaslian informasi. Kerahasiaan menjamin bahwa informasi tidak dapat diakses oleh pihak yang tidak berwenang, integritas memastikan bahwa data tidak diubah selama transmisi, dan keaslian memastikan bahwa sumber data dapat dipercaya (Stallings, 2017). Dalam era digital saat ini, kriptografi berperan penting dalam menjaga privasi dan keamanan data, terutama dengan meningkatnya ancaman keamanan siber.

Kriptografi bekerja melalui dua proses utama, yaitu enkripsi dan dekripsi. Enkripsi adalah proses mengubah data asli (*plaintext*) menjadi format yang tidak dapat dipahami (*ciphertext*) dengan menggunakan algoritma dan kunci enkripsi. Sebaliknya, dekripsi adalah proses mengembalikan *ciphertext* menjadi *plaintext* dengan memanfaatkan kunci dekripsi. Algoritma enkripsi dapat dibedakan menjadi dua jenis utama: simetris dan asimetris. Algoritma simetris menggunakan kunci yang sama untuk kedua proses enkripsi dan dekripsi, sedangkan algoritma asimetris mengandalkan pasangan kunci publik dan privat. Algoritma asimetris, seperti RSA, memungkinkan pertukaran kunci yang aman dan banyak digunakan dalam komunikasi yang memerlukan tingkat keamanan yang tinggi (Menezes et al., 1996).

Dalam konteks penelitian ini, kriptografi sangat relevan dengan latar belakang dan tujuan penelitian yang berfokus pada analisis keamanan algoritma *Enhanced AA<sub>β</sub>*. Algoritma ini adalah contoh dari algoritma kunci publik yang dirancang untuk meningkatkan keamanan komunikasi digital. Namun, meskipun dirancang dengan kompleksitas matematis yang tinggi, *Enhanced AA<sub>β</sub>* tetap rentan terhadap serangan kriptanalisis. Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi kelemahan algoritma tersebut menggunakan algoritma *Brute Force* dan *Kraitichik*, serta mencari cara untuk meningkatkan ketahanannya terhadap serangan (Katz & Lindell, 2007).

Dengan demikian, penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam meningkatkan ketahanan kriptografi terhadap ancaman keamanan yang berkembang.

### 2.1.1. GCD (*Greatest Common Divisor*)

FPB (Faktor Persekutuan Terbesar) atau GCD (*Greatest Common Divisor*) adalah bilangan bulat terbesar yang dapat membagi dua atau lebih bilangan bulat tanpa menyisakan sisa, dan memiliki peran yang sangat penting dalam meningkatkan efisiensi algoritma kriptografi, khususnya pada sistem yang menggunakan kunci publik seperti RSA dan *Enhanced AA<sub>β</sub>*. Sistem-sistem ini memanfaatkan sifat GCD untuk menjaga keamanan kunci dan mempersulit serangan kriptanalisis. Algoritma Euclidean adalah metode efisien untuk menghitung GCD, yang dilakukan dengan membagi dua bilangan secara berulang hingga sisa pembagian menjadi nol, dan menggantikan bilangan yang lebih kecil dengan yang lebih besar. Proses ini diulang hingga menemukan GCD, yang merupakan bilangan terakhir sebelum sisa menjadi nol, sehingga sangat berguna dalam aplikasi kriptografi modern (Safar et al., 2017).

#### Rumus Algoritma Euclidean

Rumus untuk menghitung GCD dari dua bilangan  $a$  dan  $b$  dengan Algoritma Euclidean adalah:

$$\text{GCD}(a,b) = \text{GCD}(b, a \bmod b)$$

Menghasilkan sisa dari pembagian  $a$  dengan  $b$ . Algoritma ini diulangi sampai sisa pembagian adalah nol, dan nilai terakhir dari  $b$  adalah GCD. Proses menghitung GCD menggunakan algoritma euclidean adalah sebagai berikut:

1. Tentukan dua bilangan bulat  $a$  dan  $b$  yang akan digunakan untuk menghitung GCD. Jika  $a < b$ , tukar nilai  $a$  dengan  $b$  agar  $a > b$ .
2. Bagi  $a$  dengan  $b$ , dan catat hasil bagi dan sisanya. Hasil bagi tidak diperlukan untuk langkah selanjutnya, namun sisa sangat penting.
3. Mengganti nilai  $a$  dengan  $b$ , dan nilai  $b$  dengan sisa pembagian yang telah dihitung.

4. Ulangi langkah 2 dan 3, lakukan pembagian lagi hingga sisa menjadi nol.
5. Ketika sisa pembagian adalah nol, maka GCD adalah bilangan terakhir yang bukan nol.

**Contoh :** Misalkan kita ingin menghitung GCD dari dua bilangan  $a = 1071$  dan  $b = 462$  menggunakan Algoritma Euclidean :

1. Pilih  $a = 1071$  dan  $b = 462$
2. Bagi  $1071 \div 462$ , hasilnya adalah 2  
dengan sisa  $1071 - (462 \times 2) = 147$   
$$1071 = 462 \times 2 + 147$$
3. Gantilah  $a = 462$  dan  $b = 147$ , lalu ulangi pembagian.
4. Bagi  $462 \div 147$ , hasilnya adalah 3  
dengan sisa  $462 - (147 \times 3) = 21$ .  
$$462 = 147 \times 3 + 21$$
5. Gantilah  $a = 147$  dan  $b = 21$  lalu ulangi pembagian.
6. Bagi  $147 \div 21$ , hasilnya adalah 7 dengan sisa nol.  
$$147 = 21 \times 7 + 0$$
7. Karena sisa adalah nol, maka GCD dari 1071 dan 462 adalah 21.

### 2.1.2. Modulo Eksponensial

Modulo eksponensial adalah operasi untuk menghitung hasil pemangkatan suatu bilangan, di mana hasil tersebut diambil sisanya setelah dibagi dengan bilangan modulus tertentu. Operasi ini sangat penting dalam kriptografi, khususnya dalam algoritma kunci publik seperti RSA, di mana keamanan enkripsi bergantung pada kesulitan memecahkan operasi modulo eksponensial dengan bilangan besar. Untuk meningkatkan efisiensi, *fast modular exponentiation* (eksponensial modular cepat) dikembangkan, yang menggunakan teknik eksponensial berulang (*repeated squaring*) untuk mengurangi jumlah perkalian yang diperlukan, sehingga mempercepat perhitungan. Dalam konteks kriptanalisis, teknik ini juga berguna untuk mengoptimalkan analisis terhadap serangan *Brute Force* atau *Kraitchik* dengan memanfaatkan kecepatan komputasi (Egecioglu & Koç, 1990).

**Rumus Algoritma *Fast modular exponentiation*:**

$$c = (a^e) \bmod n$$

Dengan algoritma *fast modular exponentiation*, perhitungan dilakukan dengan pendekatan pemangkatan berulang (*exponentiation by squaring*), yang berarti nilai eksponen  $e$  dapat dipecah menjadi beberapa perkalian berulang untuk meminimalkan jumlah operasi. Proses *fast modular exponentiation* adalah sebagai berikut :

1. Pilih nilai dasar  $a$ , eksponen  $e$ , dan modulus  $n$ .
2. Ubah eksponen  $e$  menjadi representasi biner untuk memudahkan perhitungan dengan metode eksponensial berulang.
3. Eksponensiasi berulang:
  - Mulai dengan hasil = 1.
  - Untuk setiap bit eksponen (dari bit paling signifikan ke paling rendah):
    - Jika bit bernilai 1, kalikan hasil dengan basis  $a$ , lalu lakukan operasi modulus  $n$ :  

$$\text{hasil} = (\text{hasil} \times a) \bmod n$$
    - Kuadratkan basis  $a$ , lalu lakukan operasi modulus  $n$ :  

$$a = (a^2) \bmod n$$
4. Lanjutkan iterasi hingga seluruh bit eksponen diproses untuk mendapatkan hasil akhir.
5. Setelah semua bit dievaluasi, nilai akhir dari variabel hasil akan menjadi  $(a^e) \bmod n$ .

**Contoh :** Misalkan kita ingin menghitung  $7^{13} \bmod 5$  menggunakan *Fast modular exponentiation* :

1. Pilih  $a = 7$ ,  $e = 13$ , dan  $n = 5$ .
2. Konversikan eksponen 13 ke biner:  
 $13_{10} = 1101_2$  (yang sesuai dengan  $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ ).
3. Mulai dari inisiasi hasil 1 dan basis  $7 \bmod 5 = 2$ , kemudian lakukan iterasi untuk setiap bit:
  - Iterasi 1:  $e = 13$  (biner: 1101)
    - Bit paling rendah adalah 1:

$$hasil = (1 \times 2) \bmod 5 = 2.$$

- Perbarui basis:

$$basis = (2 \times 2) \bmod 5 = 4.$$

- Geser  $e$ :

$$e = 6 \text{ (biner: 110)}$$

- Iterasi 2:  $e = 6$  (biner: 110)

- Bit paling rendah adalah 0: (tidak ada perubahan pada hasil)

- Perbarui basis:

$$basis = (4 \times 4) \bmod 5 = 1.$$

- Geser  $e$ :

$$e = 3 \text{ (biner: 11)}$$

- Iterasi 3:  $e = 3$  (biner: 11):

- Bit paling rendah adalah 1:

$$hasil = (2 \times 1) \bmod 5 = 2.$$

- Perbarui basis:

$$basis = (1 \times 1) \bmod 5 = 1.$$

- Geser  $e$ :

$$e = 1 \text{ (biner: 1)}$$

- Iterasi 4:  $e = 1$  (biner: 1)

- Bit paling rendah adalah 1:

$$hasil = (2 \times 1) \bmod 5 = 2.$$

- Perbarui basis:

$$basis = (1 \times 1) \bmod 5 = 1.$$

- Geser  $e$ :

$$e = 0 \text{ (biner: 0)}$$

4. Setelah iterasi selesai, maka hasil akhir  $7^{13} \bmod 5 = 2$ .

### 2.1.3. Invers Modulo

Invers modulo adalah konsep penting dalam teori bilangan dan kriptografi yang memungkinkan kita menemukan bilangan  $b$  sehingga  $a \cdot b \equiv 1 \bmod n$ , sehingga memungkinkan operasi pembagian dalam aritmatika modular. Konsep ini sangat relevan dalam algoritma kunci publik, di mana enkripsi dan dekripsi sering melibatkan pembagian



dalam sistem modular. Salah satu algoritma untuk menemukan invers modulo adalah Algoritma *Extended Euclidean* (EEA), yang tidak hanya menghitung *Greatest Common Divisor* (GCD) tetapi juga memberikan koefisien untuk mengekspresikan GCD sebagai kombinasi linear dari dua bilangan. EEA memungkinkan kita menemukan invers modulo dengan cepat, yang sangat penting dalam aplikasi kriptografi. Menurut Mrabet et al. (2017), "Algoritma *Extended Euclidean* menyediakan cara yang efisien untuk menghitung invers modulo, yang sangat penting untuk aplikasi kriptografi" (hal. 1), menunjukkan bahwa EEA meningkatkan efisiensi perhitungan dan kontribusi pada keamanan sistem kriptografi.

### Rumus Invers Modulo

$$a \cdot b \equiv \text{mod } n$$

di mana  $a$  adalah bilangan yang ingin dicari inversnya, dan  $n$  adalah modulus. Dengan kata lain,  $b$  adalah bilangan yang, ketika dikalikan dengan  $a$ , menghasilkan sisa 1 ketika dibagi oleh  $n$ . Konsep ini sangat penting dalam kriptografi, terutama dalam algoritma kunci publik, di mana invers modulo digunakan untuk melakukan operasi pembagian dalam sistem modular. Proses invers modulo adalah sebagai berikut:

1. Pilih bilangan  $a$  yang ingin dicari inversnya dan modulus  $n$ .
2. Periksa bahwa  $\text{gcd}(a, n) = 1$ . (Jika tidak, invers modulo tidak ada)
3. Gunakan Algoritma *Extended Euclidean* (EEA):

EEA digunakan untuk menemukan bilangan  $x$  dan  $y$  yang memenuhi persamaan:

$$a \cdot x + n \cdot y = 1$$

Di sini,  $x$  adalah invers dari  $a$  modulo  $n$ .

4. Memastikan Nilai Positif. (Jika nilai  $x$  negatif, tambahkan  $n$  untuk mendapatkan nilai positif)

**Contoh:** Misalkan kita hitung invers dari 7 modulo 26

1. Tentukan Bilangan dan Modulus:

- $a = 7$
- $n = 26$



2. Pastikan  $\text{GCD} = 1$ :

Hitung  $\text{gcd}(7, 26)$ :

- $26 = 3 \cdot 7 + 5$
- $7 = 1 \cdot 5 + 2$
- $5 = 2 \cdot 2 + 1$
- $2 = 2 \cdot 1 + 0$

Jadi,  $\text{gcd}(7, 26) = 1$ . (Invers modulo ada)

3. Gunakan Algoritma *Extended Euclidean*:

Kembali ke langkah sebelumnya untuk mengekspresikan 1 sebagai kombinasi linear dari 7 dan 26:

- $1 = 5 - 2 \cdot 2$
- $2 = 7 - 5$

Substitusi:

- $1 = 5 - 2(7 - 5)$
- $1 = 3 \cdot 5 - 2 \cdot 7$

Sekarang kita substitusi kembali  $5 = 26 - 3 \cdot 7$ :

- $1 = 3(26 - 3 \cdot 7) - 2 \cdot 7$
- $1 = 3 \cdot 26 - 11 \cdot 7$

Jadi, kita mendapatkan:

$$-11 \cdot 7 + 3 \cdot 26 = 1$$

Di mana  $x = -11$  dan  $y = 3$ .

4. Pastikan Nilai Positif:

Karena  $-11$  negatif, kita tambahkan 26:

$$x = -11 + 26 = 15$$

Sehingga invers dari 7 modulo 26 adalah 15.

#### 2.1.4. Pembangkit Bilangan Prima

Pembangkit bilangan prima merupakan algoritma krusial dalam kriptografi modern, berfungsi untuk menemukan bilangan bulat yang hanya bisa dibagi oleh 1 dan dirinya sendiri. Bilangan prima besar sangat penting dalam algoritma kunci publik seperti RSA dan *Enhanced AA<sub>β</sub>*, karena digunakan untuk menghasilkan kunci publik dan privat yang sulit untuk difaktorkan, memberikan tingkat keamanan

tinggi terhadap serangan kriptanalisis serta menjaga integritas data dalam transaksi digital. Salah satu algoritma yang banyak digunakan untuk menguji keprimaan adalah Miller-Rabin *Primality Test*, yang merupakan metode probabilistik (Conrad, 2011). Algoritma ini memungkinkan verifikasi keprimaan bilangan dengan risiko kesalahan yang dapat diminimalkan melalui pengulangan pengujian.

### Rumus Miller-Rabin *Primality Test*

Miller-Rabin *Primality Test* didasarkan pada prinsip pembagian modular dan sifat eksponensial dari bilangan prima. Jika  $p$  adalah bilangan prima, maka dapat dinyatakan bahwa:

$$a^d \bmod p = 1 \text{ atau } a^{2^r \cdot d} \bmod p = p - 1$$

dengan  $p - 1 = 2^r \cdot d$ , di mana  $d$  ganjil. Algoritma ini mengevaluasi keprimaan  $p$  dengan memanfaatkan properti tersebut melalui pengujian dengan nilai acak  $a$ . Proses untuk menghitung Miller-Rabin *Primality* yaitu adalah sebagai berikut :

1. Representasikan  $p - 1 = 2^r \cdot d$ , di mana  $d$  ganjil.
2. Pilih bilangan acak  $a$  di mana  $2 \leq a \leq p - 2$ .
3. Hitung  $x = a^d \bmod p$ :
  - a. Jika  $x = 1$  atau  $x = p - 1$ , lanjutkan ke langkah berikutnya.
4. Untuk  $i = 1$  hingga  $r - 1$ :
  - a. Hitung  $x = x^2 \bmod p$ .
  - b. Jika  $x = p - 1$ , lanjutkan pengujian berikutnya.
  - c. Jika  $x = 1$ , maka  $p$  bukan bilangan prima.
5. Jika pengujian gagal untuk semua nilai  $a$ ,  $p$  bukan bilangan prima.
6. Ulangi langkah-langkah tersebut beberapa kali dengan nilai  $a$  berbeda untuk meningkatkan probabilitas keprimaan  $p$ .

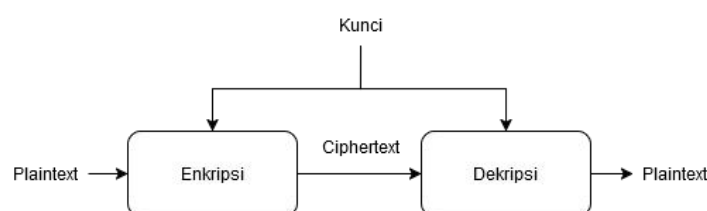
**Contoh :** Misalkan kita ingin menguji apakah  $p = 17$  adalah bilangan prima menggunakan Miller-Rabin *Primality Test*:

1. Representasikan  $p - 1 = 16 = 2^4 \cdot 1$ , sehingga  $r = 4$  dan  $d = 1$
2. Pilih  $a = 2$ .
  - a. Hitung  $x = 2^1 \bmod 17 = 2$ :
  - b. Periksa  $x$  :
    - i.  $x^2 \bmod 17 = 4$ .

- ii.  $x^4 \bmod 17 = 16$ , maka  $x = p - 1$ , dan pengujian berhasil.
  - 3. Pilih  $a = 3$ .
    - a. Hitung  $x = 3^1 \bmod 17 = 3$ :
    - b. Periksa  $x$  :
      - i.  $x^2 \bmod 17 = 9$ .
      - ii.  $x^4 \bmod 17 = 13$
      - iii.  $x^8 \bmod 17 = 16$ , maka  $x = p - 1$ , dan pengujian berhasil.
- Ulangi langkah-langkah tersebut beberapa kali dengan nilai  $a$  berbeda seperti  $a = 5$  atau  $a = 7$ . Jika semua pengujian berhasil,  $p$  dapat dianggap sebagai bilangan prima.

## 2.2. Kriptografi Simetris

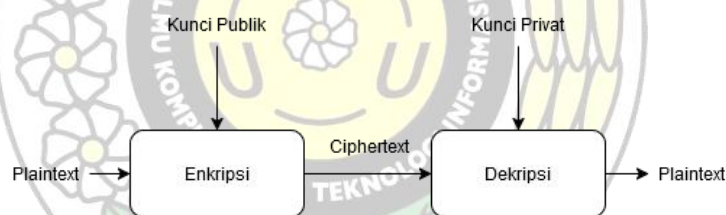
Kriptografi simetris adalah teknik enkripsi di mana kunci yang digunakan untuk enkripsi dan dekripsi adalah identik. Dalam sistem ini, baik pengirim maupun penerima harus memiliki kunci yang sama dan menjaga kerahasiaannya agar informasi yang dienkripsi tidak bisa diakses oleh pihak ketiga. Salah satu keunggulan utama dari kriptografi simetris adalah kecepatannya yang lebih tinggi dibandingkan dengan algoritma asimetris, menjadikannya sangat cocok untuk aplikasi yang membutuhkan pemrosesan data secara langsung. Namun, masalah utama yang dihadapi oleh kriptografi simetris adalah distribusi kunci, karena setiap pasangan pengguna memerlukan kunci yang berbeda untuk komunikasi yang aman. Simmons (1979) menyatakan, "Kriptografi simetris menawarkan efisiensi tinggi tetapi menghadapi tantangan dalam pengelolaan dan distribusi kunci" (hal. 305). Beberapa algoritma kriptografi simetris yang sering digunakan antara lain AES (*Advanced Encryption Standard*), DES (*Data Encryption Standard*), *Blowfish*, *TwoFish*, dan *RC4*, yang masing-masing memiliki karakteristik dan tingkat keamanan yang berbeda.



**Gambar 2.21** Algoritma Simetris

### 2.3. Kriptografi Asimetris

Kriptografi asimetris adalah metode enkripsi yang menggunakan sepasang kunci berbeda: kunci publik dan kunci privat. Kunci publik dapat dibagikan secara luas, sementara kunci privat disimpan dengan aman oleh pemiliknya. Proses enkripsi dilakukan dengan menggunakan kunci publik, dan hanya pemegang kunci privat yang dapat mendekripsi pesan yang telah dienkripsi. Keunggulan utama dari kriptografi asimetris adalah kemudahan dalam pertukaran kunci, yang menghilangkan kebutuhan untuk mendistribusikan kunci rahasia secara aman, sehingga lebih aman dibandingkan dengan kriptografi simetris. Namun, algoritma ini cenderung lebih lambat dalam proses enkripsi dan dekripsi. Contoh algoritma kriptografi asimetris yang terkenal adalah RSA (*Rivest-Shamir-Adleman*), ElGamal, dan algoritma *Enhanced AA<sub>β</sub>*. Menurut Simmons (1979), "Kriptografi asimetris menawarkan solusi untuk masalah distribusi kunci yang dihadapi oleh sistem kriptografi simetris" (hal. 306), menunjukkan bahwa algoritma ini memiliki potensi besar dalam meningkatkan keamanan komunikasi digital.



**Gambar 2.2** Algoritma Asimetris

### 2.4. File Teks

File teks adalah jenis file komputer yang menyimpan data dan informasi dalam bentuk karakter. File ini berisi berbagai karakter seperti huruf, angka, simbol, dan tanda baca. Proses input dan output dari file teks dilakukan menggunakan kode yang dikenali oleh sistem komputer, yang dikenal sebagai character set. Salah satu contoh character set yang umum digunakan adalah ASCII (*American Standard Code for Information Interchange*).

Ada beberapa ekstensi file teks yang sering digunakan, antara lain:

1. txt

Ekstensi ini merupakan format file teks yang paling sederhana dan menggunakan skema character set ASCII. File .txt biasanya berisi karakter-karakter dasar seperti angka, simbol, dan spasi.

## 2. doc dan docx

Ekstensi ini dikenal luas dalam perangkat lunak pengolah kata. Format .doc adalah versi lama, sedangkan .docx adalah versi terbaru yang lebih fleksibel untuk diubah ke format lain.

## 2.5. Kompleksitas Algoritma

Sebuah masalah bisa diselesaikan dengan berbagai algoritma. Namun, algoritma yang digunakan tidak hanya harus tepat, tetapi juga harus efisien. Efisiensi sebuah algoritma dapat diukur berdasarkan waktu eksekusi dan penggunaan memori. Algoritma yang efisien adalah algoritma yang meminimalkan waktu dan ruang yang dibutuhkan. Dengan menganalisis berbagai algoritma untuk suatu masalah, kita dapat menemukan algoritma yang paling efisien. Ukuran yang digunakan untuk menggambarkan pengukuran waktu dan ruang ini adalah kompleksitas algoritma.

Kompleksitas algoritma mengacu pada seberapa banyak sumber daya komputasi yang diperlukan untuk menyelesaikan masalah. Secara umum, algoritma yang menyelesaikan masalah dalam waktu singkat memiliki kompleksitas rendah, sementara algoritma yang memerlukan waktu lebih lama untuk menyelesaikan masalah memiliki kompleksitas yang lebih tinggi (Ulfah Nur Azizah, 2013). Kompleksitas algoritma terbagi menjadi dua jenis, yaitu kompleksitas waktu dan kompleksitas ruang.

### 1. *Time Complexity* (Kompleksitas Waktu)

Kompleksitas Waktu ( $T(n)$ ) diukur dari jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan  $n$ , di mana ukuran masukan merupakan jumlah data yang diproses oleh sebuah algoritma. Rumus dasar untuk kompleksitas waktu sering dinyatakan dalam notasi  $O$  besar, seperti  $O(1)$  untuk waktu konstan,  $O(n)$  untuk waktu linier, dan  $O(n^2)$  untuk waktu kuadratik.

### 2. *Space Complexity* (Kompleksitas Ruang)

Kompleksitas Ruang ( $S(n)$ ) diukur dari memori yang digunakan oleh struktur data dalam algoritma sebagai fungsi dari ukuran masukan  $n$ . Ini mencakup semua ruang yang diperlukan untuk menyimpan variabel, struktur data, dan informasi lainnya selama eksekusi algoritma. Rumus



dasar untuk kompleksitas ruang juga dinyatakan dalam notasi  $O$  besar, misalnya  $O(1)$  untuk penggunaan memori konstan dan  $O(n)$  untuk penggunaan memori linier.

## 2.6. Kriptanalisis

Kriptanalisis adalah disiplin ilmu yang berfokus pada pengujian dan evaluasi keamanan algoritma kriptografi, bertujuan untuk mengidentifikasi dan mengeksploitasi kelemahan dalam sistem kriptografi. Proses ini melibatkan analisis mendalam terhadap algoritma dan penerapan berbagai teknik untuk mengungkap informasi yang dilindungi, seperti memecahkan kunci atau mendapatkan informasi rahasia tanpa mengetahui kunci tersebut. Sebagai contoh, dalam enkripsi simetris, kriptanalisis dapat mencakup teknik untuk menemukan kunci dengan mempelajari pola dalam *ciphertext* dan *plaintext* (Stinson, 2006).

Berbagai jenis serangan kriptanalisis dapat diterapkan dalam evaluasi keamanan algoritma, di antaranya adalah *Brute Force attack*. Serangan *Brute Force* adalah algoritma yang mencoba setiap kemungkinan kombinasi kunci hingga menemukan kunci yang benar, yang dapat diilustrasikan dengan rumus:

$$\text{Waktu} = \text{Jumlah kemungkinan} \times \text{Waktu untuk mencoba satu kunci}$$

Dalam serangan ini, jika panjang kunci  $n$  dalam bit, maka jumlah kemungkinan kunci adalah  $2^n$ . Misalnya, untuk kunci 128 bit, ada  $2^{128}$  kemungkinan yang harus dicoba. Serangan ini dianggap tidak efisien untuk kunci yang panjangnya cukup besar, tetapi tetap menjadi ancaman serius jika kunci tersebut tidak dirancang dengan baik (Menezes et al., 1996). Keterkaitan antara kriptanalisis dan penelitian ini sangat penting, karena dengan memahami dan menerapkan teknik kriptanalisis seperti algoritma *Brute Force* dan *Kraitchik*, penelitian ini dapat mengevaluasi kelemahan pada algoritma kunci publik *Enhanced AA <sub>$\beta$</sub>*  dan memberikan rekomendasi untuk perbaikan keamanan.

## 2.7. Kunci Publik *Enhanced AA <sub>$\beta$</sub>*

Algoritma *Enhanced AA <sub>$\beta$</sub>*  adalah sistem kriptografi kunci publik yang dirancang untuk meningkatkan keamanan komunikasi melalui algoritma kriptografi asimetris. Dalam algoritma ini, terdapat dua kunci: kunci publik



dan kunci privat, yang saling terkait namun tidak dapat dengan mudah ditemukan satu sama lain. Proses ini bertujuan untuk menjamin kerahasiaan dan integritas data selama transmisi, sehingga hanya pihak yang berwenang yang dapat mengakses informasi tersebut. Algoritma ini memanfaatkan bilangan besar dan operasi matematika kompleks untuk melindungi data dari serangan kriptanalisis. Menurut Mahad et al. (2019), "Sistem kriptografi *Enhanced AA<sub>β</sub>* menunjukkan peningkatan yang signifikan dalam kecepatan dekripsi dan efisiensi memori dibandingkan dengan pendahulunya" (hal. 1).

### **Key Generation (Pembangkit Kunci)**

Proses *key generation* dalam algoritma *Enhanced AA<sub>β</sub>*, yaitu di mana kunci publik dan kunci privat dihasilkan. Proses ini adalah sebagai berikut :

- Pilih dua bilangan prima acak dan berbeda  $p$  dan  $q$  sehingga  $2^k < p, q < 2^{k+1}$  dan memenuhi  $p, q \equiv 3 \pmod{4}$ .
- Hitung  $A_2 = p^2 q$
- Pilih integer acak  $A_1$  sehingga  $2^{3k+4} < A_1 < 2^{3k+6}$ .
- Hitung integer  $d \equiv A_1^{-1} \pmod{p^2}$ .
- Kunci publik terdiri dari  $A_1$  dan  $A_2$ , sedangkan kunci privat terdiri dari  $d, p, q$ .

### **Encryption (Enkripsi)**

Proses enkripsi dalam algoritma *Enhanced AA<sub>β</sub>* sebagai berikut:

- Pilih *plaintext*  $m$  sehingga  $2^{2k-2} < m < 2^{2k-1}$ .
- Pilih integer  $t$  sehingga  $2^{4k} < t < 2^{4k+1}$ .
- Hitung *ciphertext*  $c$  menggunakan rumus:  $c = A_1 m^2 + A_2 t$
- Mengembalikan *ciphertext*  $c$ .

### **Decryption (Dekripsi)**

Proses dekripsi dalam algoritma *Enhanced AA<sub>β</sub>* dilakukan sebagai berikut:

- Terima *ciphertext*  $c$  dan kunci privat  $(d, p)$ .
- Hitung  $w \equiv cd \pmod{p^2}$
- Hitung  $m_p \equiv w^{\frac{p+1}{4}} \pmod{p}$
- Hitung  $i = \frac{w - m_p^2}{p}$
- Hitung  $j \equiv \frac{i}{2m_p} \pmod{p}$

- Hitung  $m_1 = m_p + jp$ 
  1. Jika  $m_1 < 2^{2k-1}$  maka  $m = m_1$
  2. Jika tidak,  $m = p^2 - m_1$
- Hitung  $t = \frac{c - A_1 m^2}{A_2}$
- Kembalikan *plaintext*  $m, t$ .

## 2.8. Algoritma *Brute Force*

Algoritma *Brute Force* dalam kriptografi adalah teknik yang mencoba semua kemungkinan kunci secara sistematis hingga menemukan kunci yang benar untuk mendekripsi pesan. Meskipun algoritma ini pasti menemukan kunci yang tepat, efisiensinya sangat dipengaruhi oleh panjang kunci dan sumber daya komputasi yang tersedia. Semakin panjang kunci, semakin banyak kombinasi yang harus diuji, membuat algoritma ini tidak praktis untuk kunci yang lebih besar, seperti dalam algoritma kunci publik *Enhanced AA<sub>β</sub>*, di mana kompleksitas *Brute Force* meningkat signifikan. Kelebihannya adalah kepastian dalam menemukan kunci, namun kekurangannya adalah kebutuhan waktu dan daya komputasi yang sangat besar, terutama dengan panjang kunci yang mencapai 128 bit atau lebih, yang membuatnya hampir mustahil dipecahkan oleh komputer modern dalam waktu yang wajar (Gohr, 2022). Algoritma ini beroperasi dengan cara berikut :

1. Tentukan  $A_2$ , bilangan komposit yang ingin difaktorkan.
2. Tentukan ruang pencarian faktor
3. Inisialisasi nilai awal faktor  $p_{min}$  dan menggunakan penghitung untuk melacak kunci yang dicoba.
4. Periksa kelipatan  $q = \frac{A_2}{p}$
5. Periksa validitas  $p$ , apakah  $p \times q = A_2$  dan  $p \neq q$ .
6. Jika  $p$  tidak valid, tambahkan nilai  $p$  dan ulangi proses.
7. Lanjutkan hingga kunci valid *ditemukan* atau  $p$  melebihi  $\sqrt{A_2}$ .
8. Setelah kunci valid ditemukan, catat nilai  $p$  dan  $q$  sebagai hasil faktorisasi.

**Contoh :** Misalkan kita ingin mencari kunci privat dari kunci publik  $A_2$  yaitu 75 menggunakan *Brute Force* :

1. Nilai awal  $A_2 = 75$ .
2. Tentukan ruang pencarian:  
 $\sqrt{A_2} = \sqrt{75} \approx 8.66$ .
3. Mulai dari inisiasi faktor  $p$  dari 2, kemudian lakukan iterasi sampai kelipatan  $q = \frac{A_2}{p}$ :
  - Iterasi 1:  $p = 2$ 
    - Hitung  $p^2$ :  

$$p^2 = 2^2 = 4.$$
    - Hitung  $q$ :  

$$q = \frac{75}{4} = 18.75 \text{ (tidak valid, } q \text{ harus bilangan bulat)}$$
  - Iterasi 2:  $p = 3$ 
    - Hitung  $p^2$ :  

$$3^2 = 3^2 = 9.$$
    - Hitung  $q$ :  

$$q = \frac{75}{9} = 8.33 \text{ (tidak valid, } q \text{ harus bilangan bulat)}$$
  - Iterasi 3:  $p = 5$ 
    - Hitung  $p^2$ :  

$$p^2 = 5^2 = 25.$$
    - Hitung  $q$ :  

$$q = \frac{75}{25} = 3 \text{ (tidak valid, } q \text{ harus bilangan bulat)}$$
  - Iterasi 5:  $p = 5$ 
    - Hitung  $p^2$ :  

$$p^2 = 5 = 25.$$
    - Hitung  $q$ :  

$$q = \frac{75}{25} = 3 \text{ (tidak valid, } q \text{ harus bilangan bulat)}$$
4. Setelah iterasi selesai, maka validasi faktor dengan periksa apakah  $p^2 \cdot q = A_2$  dan  $p \neq q$ , maka hasil akhir  $p = 5$  dan  $q = 3$ .

## 2.9. Algoritma *Kraitchik*

Algoritma *Kraitchik* merupakan salah satu algoritma pemfaktoran yang didasarkan pada prinsip perbedaan kuadrat Fermat. Dikembangkan oleh Maurice *Kraitchik*, algoritma ini bertujuan untuk memecahkan bilangan komposit menjadi faktor-faktornya menggunakan persamaan kuadrat yang terkait. Algoritma *Kraitchik* bekerja dengan memanfaatkan hubungan  $x^2 - y^2 = kn$ , di mana  $n$  adalah bilangan yang ingin difaktorkan dan  $k$  adalah bilangan bulat positif. Meskipun algoritma ini tidak secepat teknik modern lainnya seperti *Quadratic Sieve*, algoritma ini sangat efektif dalam konteks tertentu, termasuk pada pemfaktoran modulus RSA yang lebih kecil (Lydia, Budiman, & Rachmawati, 2021).

### Rumus Pemfaktoran *Kraitchik*

$$k \times n = x^2 - y^2$$

Keterangan:

- $n$  merupakan bilangan yang ingin difaktorkan,
- $x$  dan  $y$  merupakan bilangan bulat,
- $k$  merupakan bilangan bulat positif.

Persamaan ini menggambarkan yaitu  $x^2 - y^2$  adalah kelipatan dari  $A_2$ . Dengan menemukan  $x$  dan  $y$  yang memenuhi persamaan ini, faktor dari  $A_2$  dapat ditemukan melalui relasi  $x - y$  dan  $x + y$ . Langkah-langkah Pemfaktoran dengan Algoritma *Kraitchik*:

1. Tentukan  $A_2$ , bilangan komposit yang ingin difaktorkan.
2. Hitung  $x$  sebagai nilai yang mendekati akar kuadrat dari  $n$ , yaitu  $x = \lfloor \sqrt{A_2} \rfloor$
3. Inisialisasi kunci awal  $K_{min}$
4. Cari nilai  $y^2 = x^2 - kA_2$  untuk beberapa nilai  $k$ , mulai dari  $k = 1$  hingga  $y^2$  adalah kuadrat sempurna.
  - Periksa  $y^2$  adalah akar sempurna, maka  $y = \lfloor \sqrt{y^2} \rfloor$
5. Setelah menemukan  $y$ , faktorkan  $A_2$  menggunakan persamaan  $A_2 = (x - y)(x + y)$ .
6. Ulangi langkah ini hingga faktor  $A_2$  ditemukan.

**Contoh :** Misalkan kita ingin memfaktorkan  $A_2 = 45$  menggunakan algoritma *Kraitchik* dengan  $k = 1$ .

1. Hitung  $x_0 = \lceil \sqrt{45} \rceil \approx 6.7 = 6$
2. Gunakan persamaan  $k \times A_2 = x^2 - y^2$ , mulai dari  $k = 1$ , dan hitung:
  - Iterasi 1 ( $x = 6, k = 1$ ):
    - Hitung  $kn = k \cdot A_2 = 1 \cdot 45 = 45$ .
    - Hitung  $y^2 = x^2 - kn = 6^2 - 45 = 36 - 45 = -9$ .
    - $y^2$  negatif tidak valid
    - Tingkatkan  $x$ , reset  $k = 1$ .
  - Iterasi 2 ( $x = 7, k = 1$ ):
    - Hitung  $kn = k \cdot A_2 = 1 \cdot 45 = 45$ .
    - Hitung  $y^2 = x^2 - kn = 7^2 - 45 = 49 - 45 = 4$ .
    - $y^2$  adalah akar sempurna
    - $y = \sqrt{4} = 2$ .
3. Faktorkan  $A_2$  yaitu  $A_2 = p^2 q$  menjadi:
  - $A_2 = (x - y)(x + y)$
  - $p^2 = x + y = 7 + 2 = 9$
  - $p = \sqrt{9} = 3$
  - $q = x - y = 7 - 2 = 5$

Dengan demikian,  $A_2 = 45$  berhasil difaktorkan menjadi  $p = 3$  dan  $q = 5$ .

## BAB III

### ANALISIS DAN PERANCANGAN SISTEM

#### 3.1. Analisis

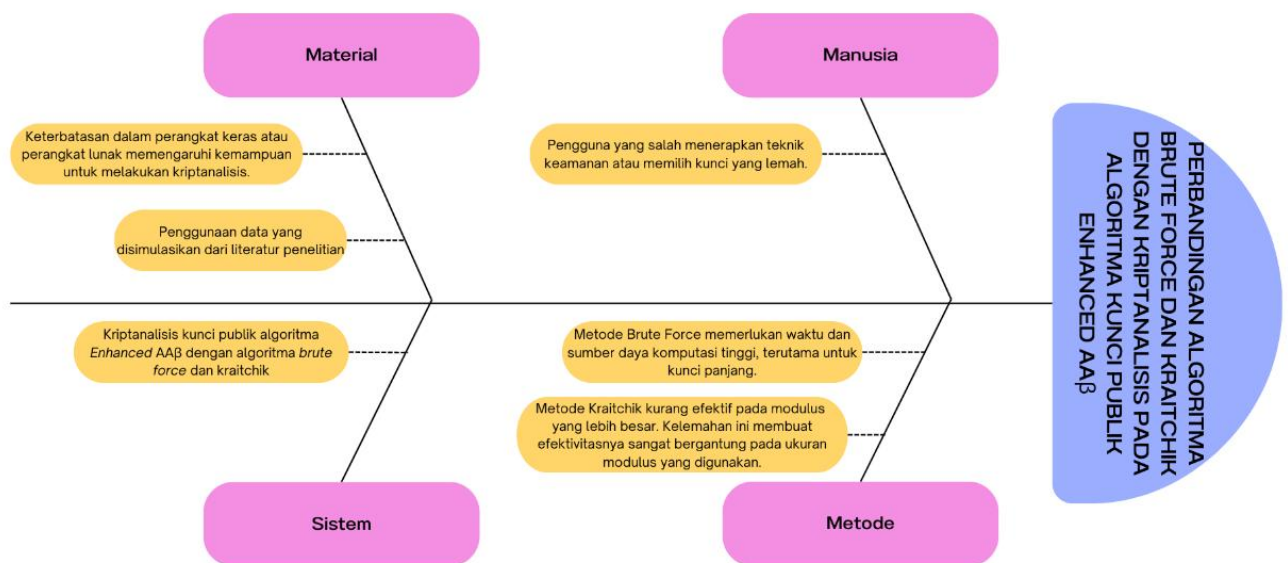
Analisis merupakan proses memecah dan menguraikan informasi menjadi komponen yang lebih kecil agar lebih mudah dipahami. Proses analisis merupakan landasan awal sebelum melakukan perancangan dan pengembangan suatu sistem, agar sesuai dengan kebutuhan dan lebih terstruktur dalam mencapai tujuan akhir. Analisis sistem terdiri dari analisis masalah dan analisis kebutuhan.

##### 3.1.1. Analisis Masalah

Algoritma kriptografi memainkan peran vital dalam melindungi data, terutama dalam era digital saat ini yang didominasi oleh perangkat *mobile* yang berfungsi untuk komunikasi dan transaksi. Namun, algoritma kriptografi memerlukan evaluasi ketahanan terhadap serangan kriptanalisis guna memastikan keamanan data dari pihak yang tidak berwenang. Dalam penelitian ini, analisis akan difokuskan pada kelemahan algoritma *Enhanced AA<sub>β</sub>*, yang merupakan pengembangan dari algoritma *AA<sub>β</sub>*, menggunakan algoritma pemecahan kunci *Brute Force* dan *Kraitchik*. Penelitian ini akan membandingkan efektivitas kedua algoritma ini dalam melakukan kriptanalisis terhadap algoritma *Enhanced AA<sub>β</sub>* untuk memahami kelemahan dan potensi perbaikan yang dapat diterapkan di masa depan.

Pada Gambar 3.1 di bawah ini, terlihat bahwa masalah utama dalam penelitian ini berada pada bagian kanan (kepala ikan), yaitu melakukan kriptanalisis terhadap kunci publik dari algoritma *Enhanced AA<sub>β</sub>*. Terdapat empat penyebab utama yang ditunjukkan pada tulang utama berbentuk horizontal, yaitu: manusia, sistem, metode, dan material. Masing-masing faktor penyebab ditunjukkan dengan tanda panah yang mengarah ke kategori utama penyebab tersebut.





**Gambar 3.1** Diagram Ishikawa Masalah Penelitian

### 3.1.2. Analisis Sistem

Analisis kebutuhan merupakan tahapan penting yang berfokus pada pengenalan dan pemahaman kebutuhan yang diperlukan untuk merancang sistem dalam memenuhi tujuan. Analisis kebutuhan dibagi dalam dua bagian utama, yaitu fungsional dan non-fungsional.

#### 1. Kebutuhan fungsional

Kebutuhan fungsional merupakan spesifikasi fungsionalitas yang dapat dilakukan dan harus ada pada suatu sistem dalam mencapai tujuan utama. Penelitian ini memiliki kebutuhan fungsional utama, yaitu:

- a. Membangkitkan kunci publik dan kunci privat sesuai dengan algoritma *Enhanced AAB*.
- b. Menerima masukan *plaintext* dalam format file berjenis \*.docx atau \*.txt, yang dapat dimasukkan oleh pengguna untuk diproses melalui enkripsi
- c. Melakukan proses enkripsi, yaitu mengenkripsi *plaintext* menjadi *ciphertext* dengan menggunakan kunci publik dan privat *Enhanced AAB*.
- d. Menghasilkan *ciphertext* yang tidak terbaca dan bermakna dari proses enkripsi yang telah dilakukan.

- e. Menerima input kunci publik yang telah dibangkitkan sebelumnya menggunakan algoritma *Enhanced AA <sub>$\beta$</sub>*  untuk dianalisis.
- f. Melakukan proses dekripsi menggunakan kunci privat untuk mengembalikan *ciphertext* ke *plaintext* yang dapat dibaca.
- g. Melakukan proses kriptanalisis menggunakan algoritma *Enhanced AA <sub>$\beta$</sub>*  dengan sistem melakukan faktorisasi kunci publik menggunakan algoritma *Brute Force* dan *Kraitchik* untuk menemukan faktor kunci.
- h. Menyimpan hasil waktu eksekusi untuk setiap algoritma kriptanalisis (*Brute Force* dan *Kraitchik*), yang akan ditampilkan untuk membantu perbandingan efektivitas kedua algoritma tersebut.

## 2. Kebutuhan non-fungsional

Kebutuhan non-fungsional merupakan spesifikasi tambahan yang mendukung sistem, dapat berupa kinerja, keamanan, serta batasan dari sistem. Penelitian ini memiliki beberapa kebutuhan non-fungsional, yaitu:

### a. Kinerja (*Performance*) :

Sistem harus dapat melakukan enkripsi dan dekripsi secara efisien menggunakan algoritma *Enhanced AA <sub>$\beta$</sub>* . Penggunaan algoritma *Brute Force* dan *Kraitchik* untuk kriptanalisis juga perlu dioptimalkan agar dapat memproses data dalam waktu yang layak pada perangkat yang spesifikasinya terbatas.

### b. Informasi (*information*) :

Menampilkan informasi status proses secara jelas, seperti notifikasi mengenai enkripsi, dekripsi, atau kriptanalisis yang sedang berlangsung, hasil yang sudah selesai, serta estimasi waktu yang tersisa untuk setiap algoritma kriptanalisis yang dijalankan.

### c. Kontrol (*Control*) :

Menangani masalah yang tidak sesuai atau salah prosedur dengan menampilkan pesan kesalahan yang informatif. Misalnya, jika input *plaintext* tidak dalam format yang didukung atau jika kunci yang dimasukkan salah, sistem harus memberikan pemberitahuan dan meminta input yang sesuai.

d. Layanan (*Service*) :

Memproses data secara akurat dan memberikan hasil enkripsi atau kriptanalisis yang konsisten. Kemampuan untuk menangani *input* dan *output* yang beragam serta kelengkapan fitur untuk penanganan kesalahan

e. Perangkat Keras (*Hardware*) :

Pengujian aplikasi dilakukan dengan perangkat yang memiliki spesifikasi sebagai berikut:

HP Android :

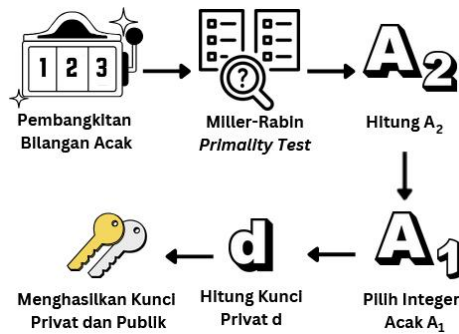
- Chipset : Mediatek Helio P35
- Ram & Rom : 4GB/128GB
- CPU dan GPU : Octa-core 2.3GHz

### 3.2. Perancangan Sistem

Perancangan sistem merupakan suatu proses yang bertujuan untuk mengembangkan sistem agar dapat beroperasi secara optimal, dengan fokus pada peningkatan efektivitas dan efisiensi. Dalam penelitian ini, perancangan sistem akan melibatkan penggunaan berbagai jenis diagram, termasuk diagram umum, diagram *use case*, diagram aktivitas, diagram urutan, dan diagram alir (*flowchart*).

#### 3.2.1. Diagram Umum Pembangkitan Kunci

Pada Gambar 3.2 menunjukkan diagram umum untuk tahap pembangkitan kunci.



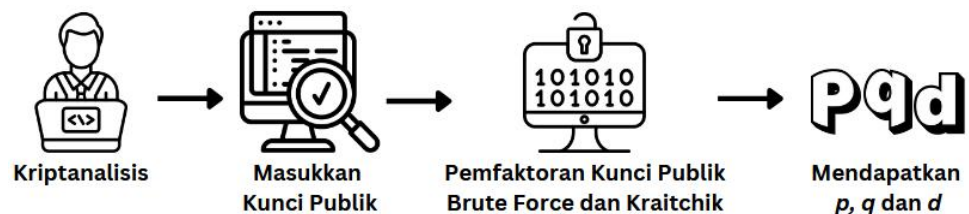
**Gambar 3.2** Diagram Umum Pembangkitan Kunci

Pada diagram tersebut, terlihat proses pembangkitan bilangan  $p$  dan  $q$  yang akan dilaksanakan dalam sistem yang sedang dikembangkan. Berikut adalah langkah-langkah yang ditunjukkan dalam diagram tersebut:

1. Pembangkitan bilangan acak yang akan digunakan sebagai dasar untuk pembangkitan kunci.
2. Menguji Keprimaan menggunakan Miller-Rabin. Uji ini digunakan untuk menentukan apakah bilangan yang dihasilkan adalah bilangan prima.
3. Menghitung  $A_2$  untuk proses pembangkitan kunci dan berkontribusi pada kompleksitas kunci yang dihasilkan.
4. Pilih sebuah bilangan integer acak  $A_1$  dalam rentang tertentu.
5. Menghitung kunci  $d$  adalah untuk bagian dari kunci privat yang akan digunakan dalam proses enkripsi.
6. Sistem mengeluarkan kunci privat dan publik.

### 3.2.2. Diagram Umum Kriptanalisis

Pada Gambar 3.3 menunjukkan diagram umum untuk tahap kriptanalisis.



**Gambar 3.3** Diagram Umum Kriptanalisis

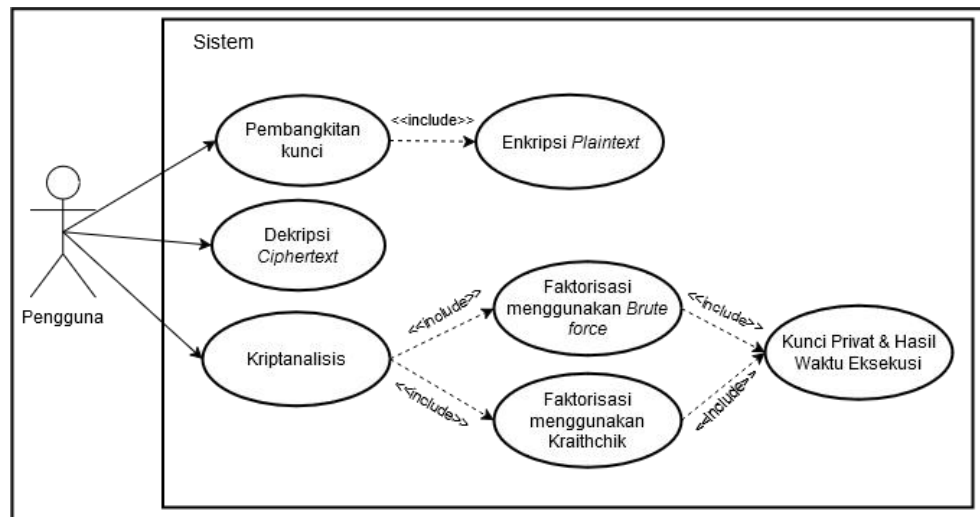
Pada Gambar 3.3, ditampilkan langkah-langkah yang akan diambil dalam proses kriptanalisis pada sistem yang sedang dikembangkan. Berikut adalah urutan langkah-langkah yang terlihat pada gambar tersebut:

1. Kriptanalisis bertujuan untuk mendapatkan informasi mengenai kunci publik.
2. Kunci publik tersebut akan dimasukkan ke dalam sistem, yang kemudian akan memanfaatkan kunci ini sebagai *input* untuk proses kriptanalisis.
3. Sistem akan melakukan pemfaktoran terhadap kunci publik dengan menggunakan dua algoritma kriptanalisis, yaitu algoritma *Brute Force* dan *Kraitichik*. Kedua algoritma ini akan diterapkan untuk mencoba menemukan faktor-faktor  $p, q$  dan kemudian melakukan invers modulo dari  $p$  untuk mendapatkan  $d$ .
4. Setelah tahap pemfaktoran selesai, sistem akan memperoleh dua faktor ( $p$  dan  $q$ ) yang selanjutnya akan digunakan untuk menghasilkan kunci privat  $d$ .

### 3.2.3. Use Case Diagram

*Use case* diagram adalah jenis grafik dalam pemodelan perangkat lunak yang digunakan untuk memvisualisasikan hubungan antara sistem perangkat lunak dengan berbagai aktor, baik itu pengguna maupun entitas eksternal lainnya, yang berinteraksi dengan sistem. Diagram ini berfungsi untuk mengidentifikasi, menjelaskan, dan memahami cara kerja sistem yang dirancang dalam konteks situasi nyata. *Use case* diagram yang menggambarkan sistem pada penelitian ini dapat dilihat pada Gambar 3.4.





**Gambar 3.4** Use Case Diagram

Pada Gambar 3.4, seorang aktor Pengguna dapat menjalankan beberapa fungsi utama pada sistem yang dirancang. Pengguna dapat melakukan pembangkitan kunci publik dan privat melalui proses uji bilangan prima dengan Miller-Rabin, dilanjutkan dengan enkripsi *plaintext* yang diambil dari direktori penyimpanan. Proses ini menghasilkan *ciphertext* yang akan disimpan untuk penggunaan selanjutnya.

Kemudian pengguna juga dapat melakukan dekripsi *ciphertext*. Proses ini mencakup memasukkan kunci publik  $(A_1, A_2)$  dan kunci privat  $(p, q, d)$ ,  $k$  serta *ciphertext*, validasi data yang dimasukkan, kemudian lakukan dekripsi sehingga menghasilkan *plaintext* yang nantinya bisa disimpan. Selain itu pengguna dapat melakukan kriptanalisis dengan memasukkan kunci publik  $(A_1, A_2)$  serta melakukan validasi. Jika sudah maka menggunakan algoritma *Brute Force* dan *Kraitchik*. Sistem menghitung waktu eksekusi untuk masing-masing algoritma, memberikan perbandingan efektivitas keduanya. Setelah proses kriptanalisis selesai, kunci privat  $(p, q, d)$  dihasilkan dan disimpan bersama hasil waktu eksekusi kedua algoritma.

#### 3.2.4. Activity Diagram

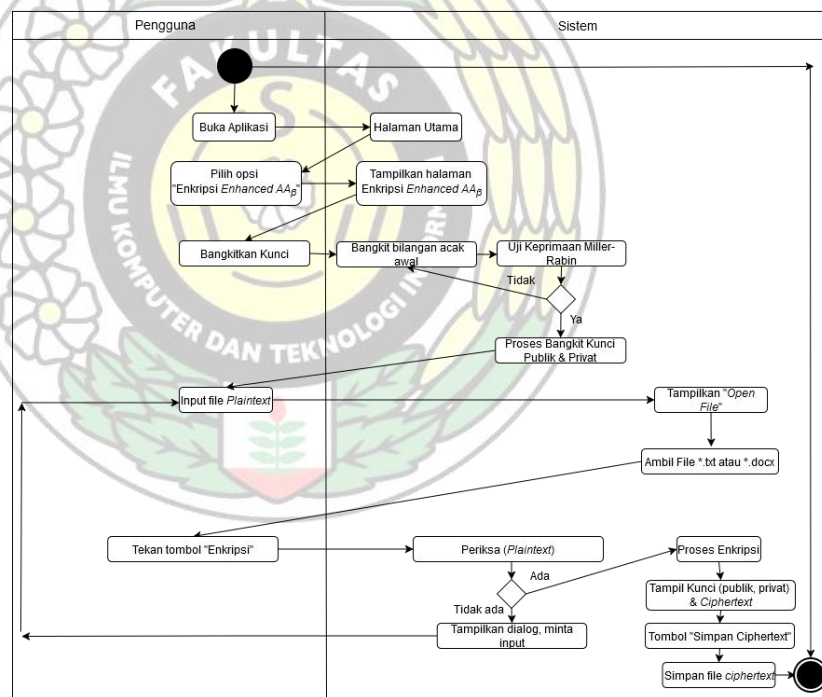
*Activity* diagram adalah representasi grafis yang digunakan untuk menggambarkan alur aktivitas dalam sebuah sistem, memberikan



pemahaman yang jelas mengenai urutan langkah-langkah yang dilakukan dalam suatu proses. Dalam penelitian ini, terdapat tiga diagram untuk sistem yang dikembangkan, yakni *activity* diagram untuk pembangkitan kunci, *activity* diagram untuk proses kriptanalisis kunci publik, dan *activity* diagram untuk proses dekripsi *ciphertext*.

### 3.2.4.1. Activity Diagram Pembangkit Kunci dan Enkripsi

Pada Gambar 3.5 menggambarkan graf aktivitas pada fase pembangkitan kunci yang digunakan di sistem yang dibuat pada penelitian ini. *Activity* diagram di fase ini memiliki dua buah kotak, yakni kotak paling kiri menunjukkan aktivitas yang dilakukan pengguna pada sistem, lalu kotak tengah menunjukkan respon apa yang saja yang dilakukan oleh sistem.



**Gambar 3.5** Activity Diagram Pembangkit Kunci dan Enkripsi

Proses pembangkitan kunci dimulai ketika pengguna membuka aplikasi dan memilih opsi "Enkripsi *Enhanced AA<sub>β</sub>*" pada halaman utama. Sistem akan menampilkan halaman enkripsi *Enhanced AA<sub>β</sub>*, dimana pengguna dapat memulai proses dengan membangkitkan kunci. Pada tahap ini, pengguna diminta untuk

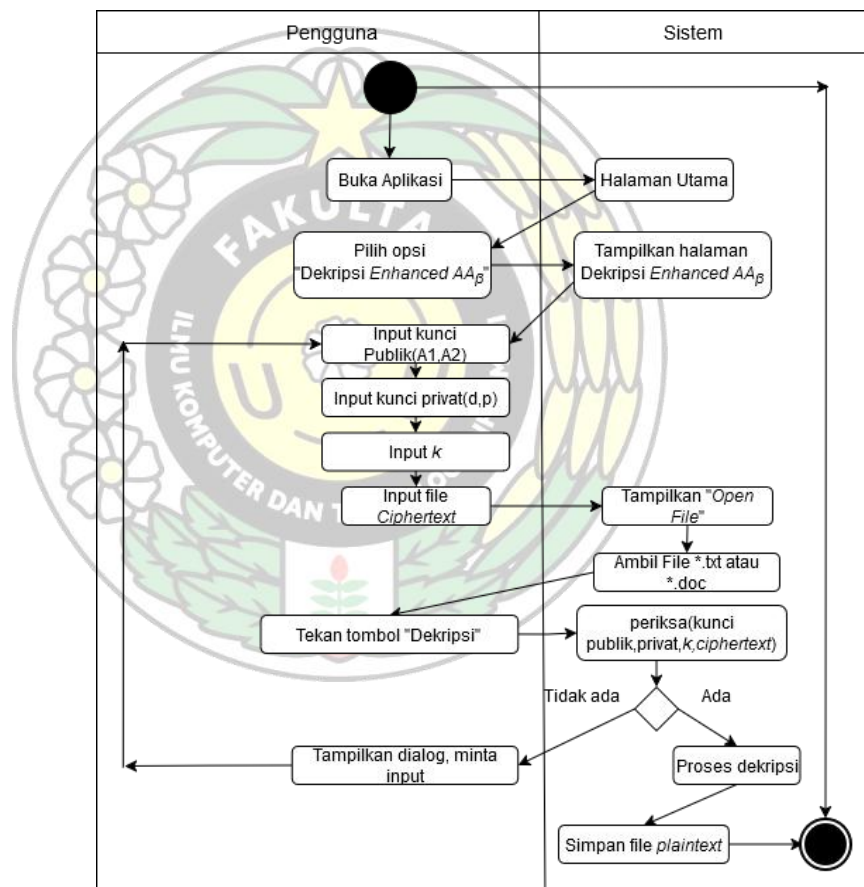
memasukkan panjang kunci  $k$ , kemudian jika ditekan bangkitkan kunci maka sistem akan menghasilkan bilangan acak awal yang kemudian diuji keprimaannya menggunakan algoritma Miller-Rabin. Jika bilangan tersebut tidak prima, proses pembangkitan bilangan acak akan diulang hingga ditemukan bilangan prima yang valid. Setelah validasi keprimaan selesai, sistem melanjutkan ke proses pembangkitan kunci publik  $(A_1, A_2)$  dan kunci privat  $(p, q, d)$ .

Pengguna kemudian dapat memasukkan file *plaintext* dengan memilih opsi "Open File" untuk mengunggah file berformat \*.txt atau \*.docx. Setelah file dipilih, sistem akan memeriksa validitas isi file. Jika file tidak valid, sistem akan menampilkan dialog untuk meminta *input* ulang dari pengguna. Jika valid, pengguna dapat melanjutkan proses enkripsi dengan menekan tombol "Enkripsi". Sistem akan mengenkripsi *plaintext* menggunakan kunci publik yang telah dihasilkan sebelumnya. Hasil enkripsi berupa *ciphertext*, bersama dengan kunci publik dan privat, akan ditampilkan kepada pengguna. Sebagai langkah akhir, pengguna dapat menyimpan file *ciphertext* dengan menekan tombol "Simpan Ciphertext". Proses ini memastikan pengguna dapat melakukan enkripsi secara efektif dan aman menggunakan algoritma *Enhanced AA<sub>β</sub>*.

#### 3.2.4.2. Activity Diagram Dekripsi

Pada gambar 3.7 menunjukkan proses dekripsi pesan dalam sistem. Proses dimulai ketika pengguna membuka aplikasi dan memilih opsi "Dekripsi *Enhanced AA<sub>β</sub>*" dari halaman utama. Sistem kemudian menampilkan halaman dekripsi, dimana pengguna dapat memasukkan kunci privat ( $d$ ) dan memilih file *ciphertext* yang ingin didekripsi. File *ciphertext* dapat diambil melalui dialog "open file" dengan format \*.txt atau \*.doc.

Setelah data berupa kunci privat dan file *ciphertext* dimasukkan, pengguna menekan tombol "Dekripsi & Simpan." Sistem akan memeriksa kelengkapan dan validitas data yang dimasukkan kunci publik ( $A_1, A_2$ ), kunci privat ( $p, q, d$ ) dan *ciphertext*. Jika data valid, sistem melanjutkan proses dekripsi menggunakan algoritma *Enhanced AA<sub>β</sub>* untuk mengubah *ciphertext* menjadi *plaintext*. Setelah proses dekripsi selesai, hasil *plaintext* disimpan ke direktori yang ditentukan oleh pengguna. Diagram ini menggambarkan alur lengkap dari pengambilan input hingga penyimpanan hasil dekripsi dalam sistem.

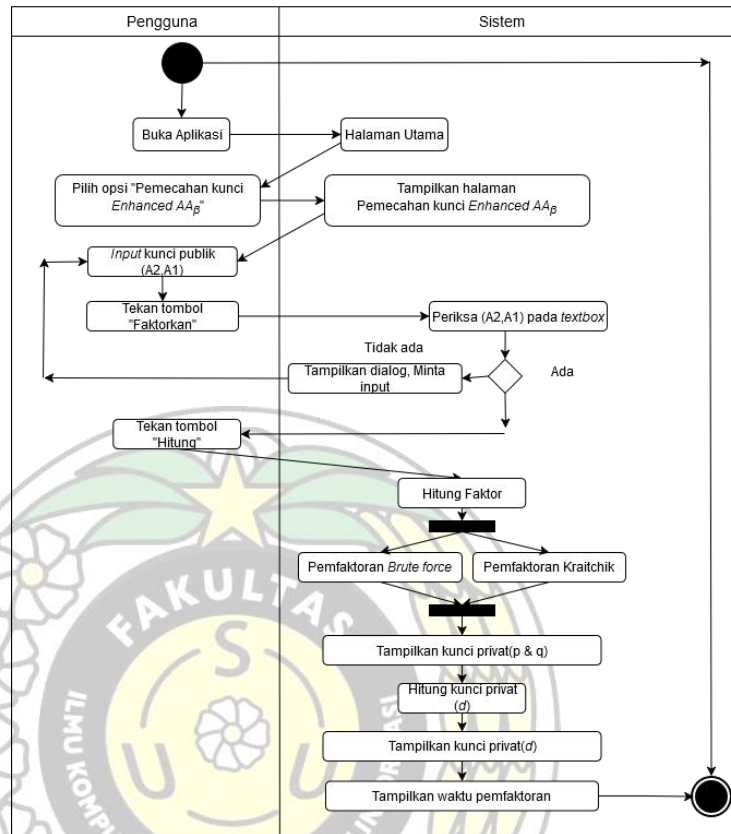


**Gambar 3.6** Activity Diagram Dekripsi

### 3.2.4.3. Activity Diagram Kriptanalisis

Pada gambar 3.7 menggambarkan graf aktivitas pada fase kriptanalisis yang digunakan di sistem yang dibuat pada penelitian ini. Activity diagram di fase ini memiliki dua buah

kotak, yakni kotak paling kiri menunjukkan aktivitas yang dilakukan pengguna pada sistem, lalu kotak tengah menunjukkan respon apa yang saja yang dilakukan oleh sistem.



**Gambar 3.7** Activity Diagram Kriptanalisis

Proses kriptanalisis dimulai ketika pengguna membuka aplikasi dan memilih opsi "Pemecahan Kunci *Enhanced AA<sub>β</sub>*" di halaman utama. Sistem akan menampilkan halaman pemecahan kunci dimana pengguna dapat memasukkan kunci publik  $A_1, A_2$  ke dalam kolom input. Setelah memasukkan nilai  $A_1, A_2$ , pengguna menekan tombol "Hitung" untuk memulai proses. Sistem terlebih dahulu memeriksa validitas input  $A_1, A_2$ . Jika input tidak valid, sistem akan menampilkan dialog untuk meminta input ulang. Namun, jika valid, proses akan dilanjutkan ke tahap faktorisasi. Tahap faktorisasi dilakukan menggunakan dua algoritma, yaitu algoritma *Brute Force* dan algoritma *Kraitichik*. Pada algoritma *Brute Force*, sistem mencoba semua kemungkinan faktor dari  $A_2$  hingga ditemukan nilai  $p$  dan  $q$ . Sedangkan pada algoritma

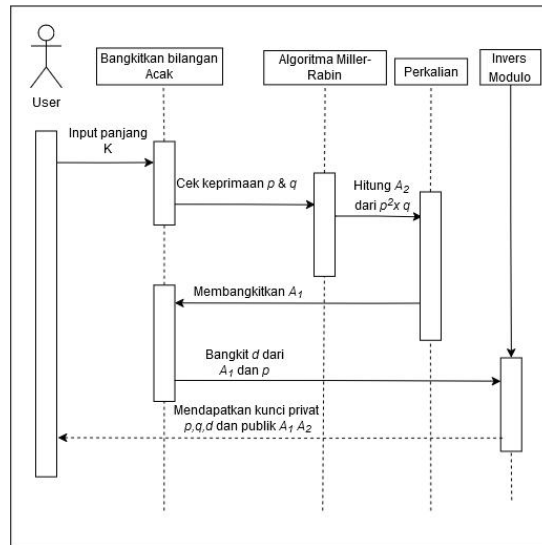
*Kraitichik*, sistem menggunakan teknik faktorisasi berbasis pola untuk menemukan nilai  $p$  dan  $q$ . Setelah kedua metode selesai, hasil berupa nilai  $d$ ,  $p$  dan  $q$ , beserta waktu eksekusi masing-masing metode, akan ditampilkan kepada pengguna. Selanjutnya, sistem menghitung kunci privat  $d$  berdasarkan nilai  $p$  dan  $A_1$  yang telah ditemukan. Sebagai langkah akhir, sistem menampilkan hasil pembangkitan kunci berupa kunci privat  $(p,q,d)$  kepada pengguna. Proses ini memberikan gambaran lengkap bagaimana pengguna dapat memanfaatkan sistem untuk melakukan pembangkitan kunci sekaligus membandingkan efisiensi dua algoritma faktorisasi yang digunakan.

### 3.2.5. Sequence Diagram

*Sequence* diagram atau diagram urutan adalah grafik pemodelan perangkat lunak yang berfungsi untuk menggambarkan hubungan antara objek dalam suatu sistem berdasarkan urutan waktu tertentu. Diagram ini menunjukkan bagaimana objek saling terhubung dan berinteraksi. Dalam penelitian ini, sistem yang akan dikembangkan mencakup sebuah diagram urutan, yaitu diagram urutan pembangkitan kunci dan kriptanalisis.

#### 3.2.5.1. Sequence Diagram Pembangkitan Kunci

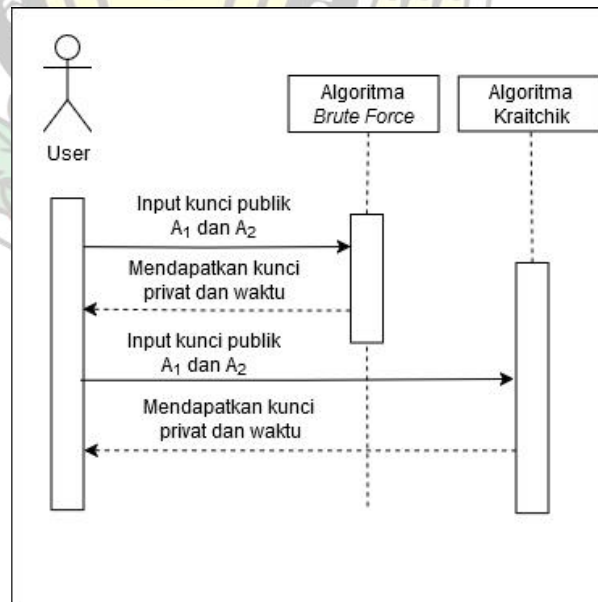
Pada Gambar 3.8 menunjukkan bagaimana *sequence* diagram untuk interaksi antara user dan sistem pada fase pembangkitan kunci untuk mendapatkan kunci privat  $p,q,d$  dan publik  $A_1 A_2$ .



**Gambar 3.8** *Sequence Diagram* Pembangkitan Kunci

### 3.2.5.2. *Sequence Diagram* Kriptanalisis

Pada Gambar 3.9 menunjukkan bagaimana *sequence diagram* untuk interaksi antara user dan sistem pada fase kriptanalisis untuk mendapatkan kunci privat  $p, q, d$  dan waktu eksekusi program.



**Gambar 3.9** *Sequence Diagram* Kriptanalisis

### 3.2.6. *Flowchart* (Diagram Alir)

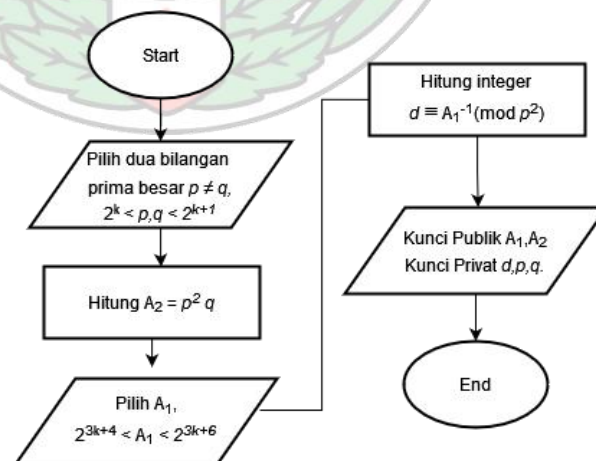
*Flowchart* adalah representasi visual dari rangkaian proses, keputusan, dan alur logika dalam sebuah sistem, yang digambarkan menggunakan



simbol-simbol standar yang ditetapkan oleh American National Standards Institute (ANSI). Simbol-simbol tersebut meliputi panah, persegi panjang, hexagon, dan lainnya. Dalam penelitian ini, terdapat beberapa *flowchart* yang menjelaskan alur algoritma *Enhanced AA<sub>β</sub>*, termasuk proses-proses penting seperti pembangkitan kunci, enkripsi, dekripsi, dan kriptanalisis. *Flowchart* ini membantu memvisualisasikan langkah-langkah secara sistematis dan terstruktur.

### 3.2.6.1. Flowchart Pembangkit Kunci

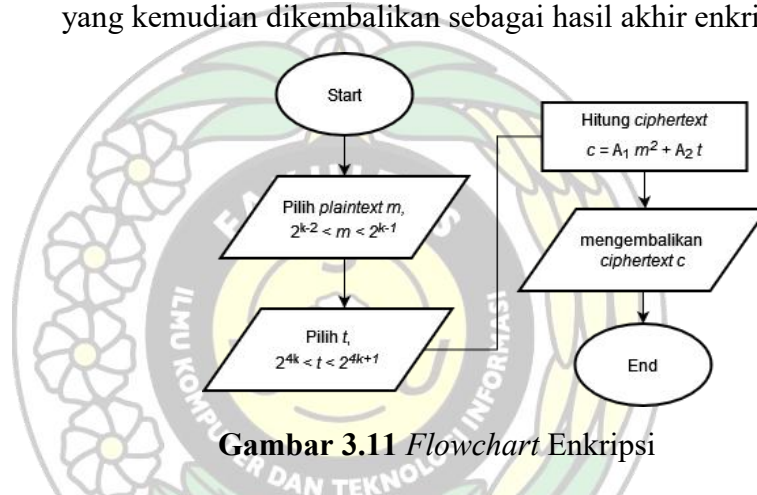
*Flowchart key generation* yang ditunjukkan pada Gambar 3.10. Proses pembangkitan kunci dalam sistem menggunakan algoritma *Enhanced AA<sub>β</sub>*. Proses diawali dengan memilih dua bilangan prima besar  $p$  dan  $q$  dengan ketentuan  $2^k < p, q < 2^{k+1}$ . Setelah itu, sistem menghitung nilai  $A_2 = p^2 \cdot q$  sebagai bagian dari parameter kunci. Selanjutnya, dipilih nilai  $A_1$  yang memenuhi syarat  $2^{3k+4} < A_1 < 2^{3k+6}$ , kemudian dihitung nilai integer  $d$  dengan rumus  $d \equiv A_1^{-1} \pmod{p^2}$ . Hasil dari fase ini adalah kunci publik berupa pasangan  $A_1$  dan  $A_2$ , serta kunci privat berupa  $d$ ,  $p$ , dan  $q$ , yang nantinya digunakan dalam proses enkripsi dan dekripsi pada sistem.



**Gambar 3.10** *Flowchart Key Generation*

### 3.2.6.2. Flowchart Enkripsi

Flowchart enkripsi yang ditunjukkan pada Gambar 3.11 menggunakan algoritma *Enhanced AA<sub>β</sub>* yang dimulai dengan memilih *plaintext*  $m$  yang akan dienkripsi, di mana *plaintext* harus memenuhi syarat  $2^{k-2} < m < 2^{k-1}$ . Selanjutnya, memilih nilai  $t$  dalam rentang  $2^{4k} < t < 2^{4k+1}$ , yang berfungsi sebagai bilangan acak dalam proses enkripsi. Setelah nilai  $m$  dan  $t$  dipilih, sistem menghitung *ciphertext*  $c$  menggunakan formula  $c = A_1 m^2 + A_2 t$ , dengan  $A_1$  dan  $A_2$  adalah parameter kunci publik yang telah dihasilkan sebelumnya. Dari proses ini membentuk *ciphertext*  $c$  yang kemudian dikembalikan sebagai hasil akhir enkripsi.

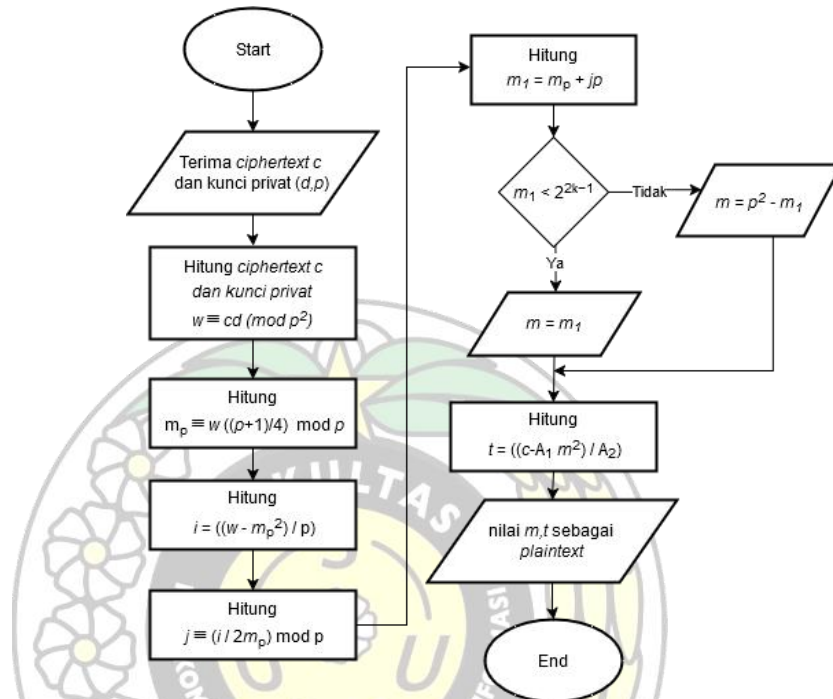


Gambar 3.11 Flowchart Enkripsi

### 3.2.6.3. Flowchart Dekripsi

Flowchart dekripsi yang ditunjukkan pada Gambar 3.12. Proses dekripsi algoritma *Enhanced AA<sub>β</sub>* diawali dengan penerimaan *ciphertext* dan kunci privat oleh penerima pesan. Langkah pertama adalah menghitung nilai awal dekripsi berdasarkan *ciphertext* dan kunci privat. Selanjutnya, hasil perhitungan tersebut digunakan untuk mendapatkan nilai antara yang merepresentasikan bagian *plaintext*. Pada tahap berikutnya, dilakukan serangkaian perhitungan tambahan untuk menentukan nilai final *plaintext*. Proses ini melibatkan pengujian kondisi tertentu untuk memastikan hasil akhir sesuai dengan batasan yang ditentukan. Jika kondisi terpenuhi, nilai *plaintext* langsung ditentukan, sedangkan jika tidak, dilakukan penyesuaian nilai

untuk mendapatkan hasil yang valid. Tahap terakhir adalah menghitung nilai pendukung tambahan untuk melengkapi hasil dekripsi. Proses ini akhirnya menghasilkan *plaintext* yang telah terenkripsi sebelumnya, yang kemudian dikembalikan sebagai hasil akhir dekripsi.

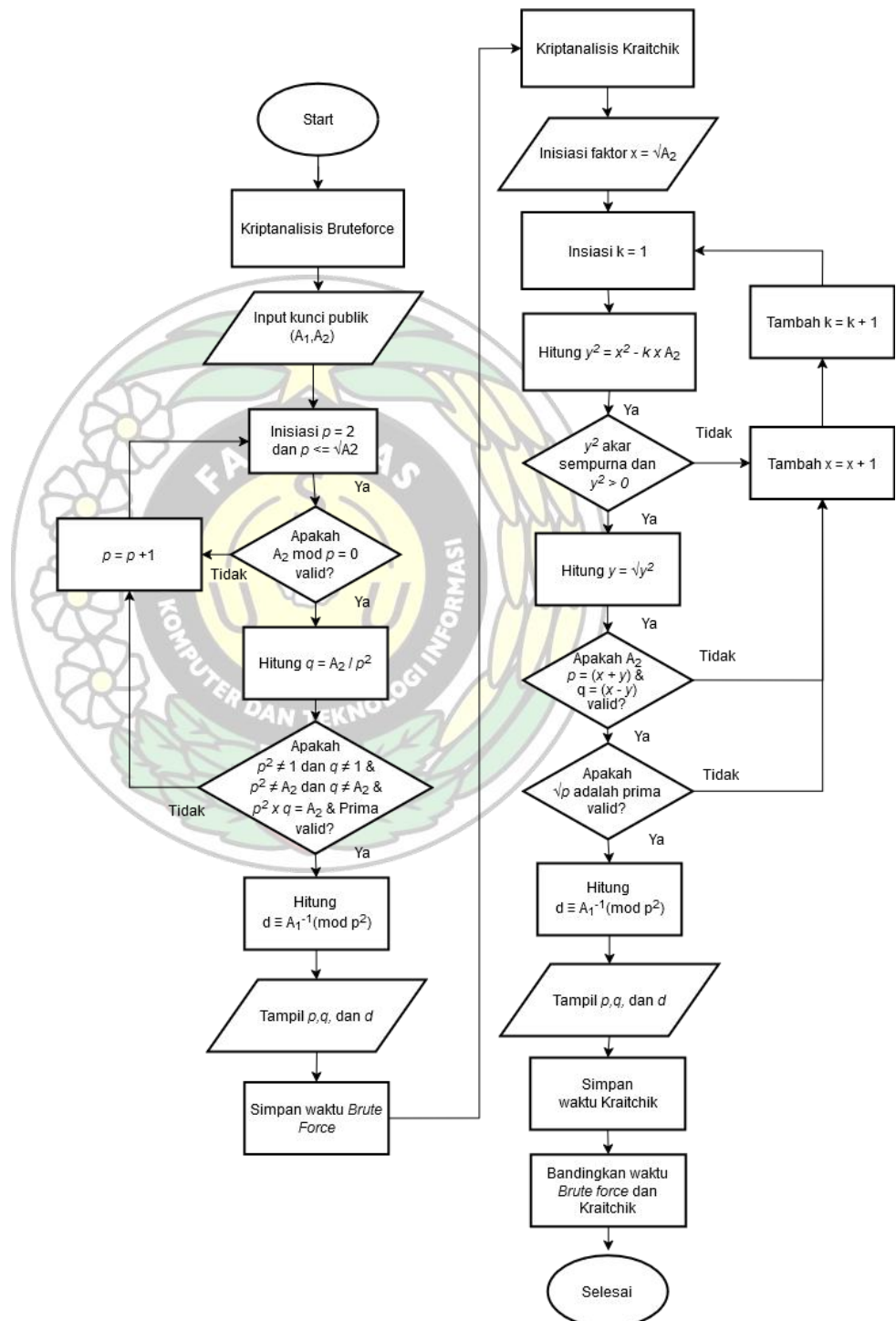


**Gambar 3.12** Flowchart Dekripsi

#### 3.2.6.4. Flowchart Kriptanalisis

Flowchart Kriptanalisis yang ditunjukkan pada Gambar 3.13. Proses kriptanalisis terhadap algoritma kunci publik *Enhanced AA<sub>β</sub>* menggunakan metode *Brute Force* dan *Kraitichik* secara paralel. Proses dimulai dengan memasukkan kunci publik  $A_1$  dan  $A_2$ . Pada jalur *Brute Force*, faktor-faktor minimal dari di inialisasi  $p$  dimulai dari 2, kemudian dilakukan proses iteratif untuk memeriksa validitas faktor  $p$  adalah  $\sqrt{A_2}$  sampai memenuhi persamaan  $p^2 \times q = A_2$ . Jika valid, kunci privat  $d$  dihitung, dan waktu eksekusi metode *Brute Force* dicatat. Pada jalur *Kraitichik*, proses dimulai dengan inialisasi faktor minimal dari  $A_2$ , diikuti oleh langkah iteratif untuk memvalidasi faktor  $p$  dan  $q$ . Jika ditemukan pasangan faktor yang valid, kunci privat  $d$  dihitung,

dan waktu eksekusi metode *Kraitichik* juga dicatat. Tahap terakhir adalah perbandingan waktu eksekusi antara kedua metode untuk menilai efisiensi dan performa masing-masing algoritma dalam melakukan kriptanalisis terhadap algoritma *Enhanced AA<sub>β</sub>*. Proses berakhir setelah semua langkah selesai.



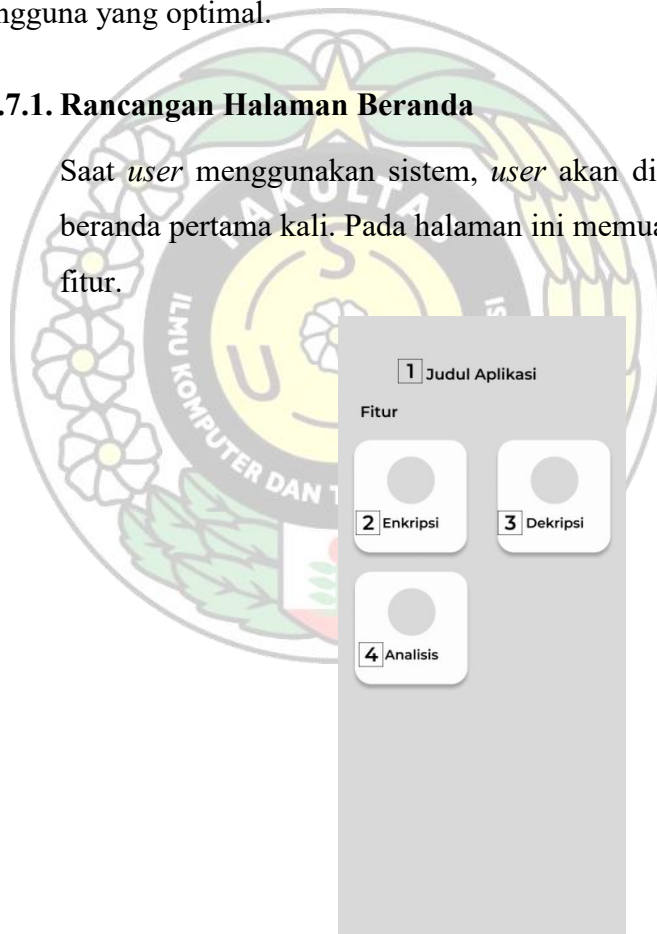
**Gambar 3.13** Flowchart Kriptanalisis

### 3.2.7. Perancangan *User Interface*

Perancangan *User Interface* (UI) adalah proses yang bertujuan untuk menciptakan antarmuka perangkat lunak yang efektif, sehingga pengguna dapat berinteraksi dengan sistem dengan mudah dan efisien. UI berfungsi sebagai penghubung antara pengguna dan komputer, memungkinkan interaksi melalui tampilan yang ditampilkan di layar. Dalam proses perancangan UI, fokus utama adalah menciptakan antarmuka yang intuitif, responsif, dan sesuai dengan kebutuhan pengguna. Desain UI harus mempertimbangkan berbagai elemen seperti tata letak, warna, tipografi, dan navigasi untuk memastikan pengalaman pengguna yang optimal.

#### 3.2.7.1. Rancangan Halaman Beranda

Saat *user* menggunakan sistem, *user* akan ditampilkan halaman beranda pertama kali. Pada halaman ini memuat tentang judul dan fitur.



**Gambar 3.14** Rancangan Halaman Beranda

Keterangan pada Gambar 3.14 ialah sebagai berikut :

1. Judul Aplikasi
2. Pilihan halaman yang ingin digunakan
3. Pilihan halaman yang ingin digunakan

4. Pilihan halaman yang ingin digunakan

### 3.2.7.2. Rancangan Halaman Enkripsi

Pada halaman enkripsi menampilkan proses yang dilakukan yaitu *user* memasukkan panjang kunci  $k$  yang akan dipakai untuk membangkitkan kunci publik. Serta menampilkan proses enkripsi yang dilakukan dengan memasukkan *file* dan memproses dari kunci publik dan privat kemudian menghasilkan *ciphertext*.

**Gambar 3.15** Rancangan Halaman Enkripsi

Keterangan pada Gambar 3.15 ialah sebagai berikut :

1. Teks Halaman
2. Kotak untuk menginputkan besar digit  $k$
3. Tombol untuk membangkitkan bilangan kunci privat  $p, q, d$  dan kunci publik  $A_1, A_2$
4. Teks untuk menampilkan bilangan kunci privat  $p, q, d$  dan kunci publik  $A_1, A_2$
5. Tombol untuk memasukkan *file plaintext*
6. Tombol untuk memproses *ciphertext*
7. Teks untuk menampilkan *ciphertext*
8. Tombol untuk simpan *ciphertext*



### 3.2.7.3. Rancangan Halaman Dekripsi

Pada halaman dekripsi menampilkan proses yang dilakukan yaitu *user* memasukkan kunci publik ( $A_1$   $A_2$ ) dan kunci privat ( $d, p, q$ ) yang akan dipakai untuk mendapatkan *plaintext*.

The image shows a mobile application interface for decryption. It features a vertical list of input fields and buttons. The elements are numbered as follows: 1. Title bar 'Judul Halaman'; 2. Input field for 'Kunci Publik A1'; 3. Input field for 'Kunci Publik A2'; 4. Input field for 'Kunci Privat d'; 5. Input field for 'Kunci Privat p'; 6. Input field for 'Panjang Kunci k'; 7. Button with a file icon and text 'Pilih File'; 8. Button with text 'Dekripsi'; 9. Text area labeled 'Plaintext:'; 10. Button with text 'Simpan Plaintext'.

**Gambar 3.16** Rancangan Halaman Dekripsi

Keterangan pada Gambar 3.16 ialah sebagai berikut :

1. Teks Halaman
2. Kotak untuk menginputkan kunci publik  $A_1$
3. Kotak untuk menginputkan kunci publik  $A_2$
4. Kotak untuk menginputkan kunci publik  $d$
5. Kotak untuk menginputkan kunci publik  $p$
6. Kotak untuk menginputkan kunci publik  $k$
7. Tombol untuk memasukkan file *plaintext*
8. Tombol untuk proses dekripsi
9. Kotak untuk menampilkan *plaintext*
10. Tombol untuk simpan *plaintext*

### 3.2.7.4. Rancangan Halaman Kriptanalisis

Pada halaman Kriptanalisis menampilkan proses yang dilakukan yaitu *user* memasukkan kunci publik ( $A_1$   $A_2$ ) dan akan dipakai untuk mendapatkan kunci privat  $p, q$ , dan  $d$ .

1 Judul Halaman

2 Kunci Publik A1

3 Kunci Publik A2

4 ☐ Hitung

Brute Force :

5

Kraitchik :

6

Waktu program (ms) :

Brute Force

Kraitchik

7

**Gambar 3.17** Rancangan Halaman Kriptanalisis

Keterangan pada Gambar 3.17 ialah sebagai berikut :

1. Teks Halaman
2. Kotak untuk memasukkan kunci publik  $A_1$
3. Kotak untuk memasukkan kunci publik  $A_2$
4. Tombol untuk proses Kriptanalisis
5. Kotak untuk menampilkan  $d$ ,  $p$ , dan  $q$ , dengan *Brute Force*.
6. Kotak untuk menampilkan  $d$ ,  $p$ , dan  $q$ , dengan *Kraitchik*.
7. Kotak untuk menampilkan waktu program dari algoritma *Brute Force* dan *Kraitchik*.

## BAB IV

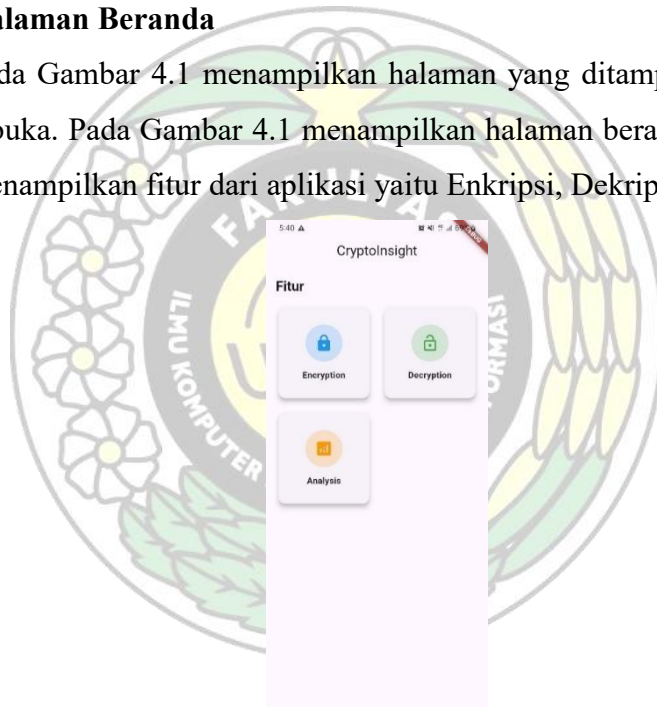
### IMPLEMENTASI DAN PENGUJIAN

#### 4.1. Implementasi

Setelah melakukan analisis dan perancangan, tahap selanjutnya adalah implementasi dan pengujian. Pada tahap ini, penelitian menggunakan perangkat lunak Android Studio sebagai *Integrated Development Environment* (IDE). Bahasa pemrograman yang digunakan dalam penelitian ini adalah Dart, dengan memanfaatkan *framework* Flutter untuk membangun dan menjalankan aplikasi yang dikembangkan.

##### 4.1.1. Halaman Beranda

Pada Gambar 4.1 menampilkan halaman yang ditampilkan saat aplikasi dibuka. Pada Gambar 4.1 menampilkan halaman beranda yang berfungsi menampilkan fitur dari aplikasi yaitu Enkripsi, Dekripsi serta Analisis.



**Gambar 4.1** Halaman Beranda

##### 4.1.2. Halaman Enkripsi

Pada Gambar 4.2 menampilkan halaman Enkripsi. Halaman ini berfungsi untuk membangkitkan kunci publik dan kunci privat. Setelah pembangkitan kunci berhasil, pengguna dapat memasukkan file *plaintext* dalam format \*.docx atau \*.txt. Setelah dimasukkan pengguna bisa langsung mengenkripsi *plaintext* untuk menghasilkan *ciphertext*.

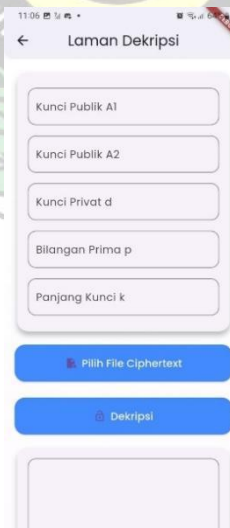
Setelah berhasil, pengguna dapat menyimpan hasil dari *ciphertext* ke dalam perangkat.



**Gambar 4.2** Halaman Enkripsi

#### 4.1.3. Halaman Dekripsi

Pada Gambar 4.3 menampilkan halaman Dekripsi. Halaman ini berfungsi untuk melakukan dekripsi *ciphertext* untuk mendapatkan *plaintext* dari kunci publik dan privat yang telah dibangkitkan sebelumnya serta panjang kunci  $k$ .

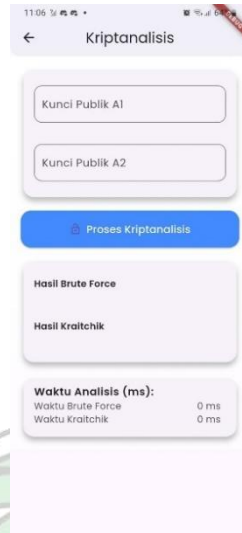


**Gambar 4.3** Halaman Dekripsi

#### 4.1.4. Halaman Kriptanalisis

Pada Gambar 4.3 menampilkan halaman Analisis. Halaman ini berfungsi untuk melakukan pemfaktoran terhadap kunci publik dengan algoritma

*Brute Force* dan *Kraitichik* dan mendapatkan hasil faktor dari kunci privat yaitu  $p, q$ , dan  $d$ , serta mendapatkan lama waktu dari pemfaktoran kedua algoritma.



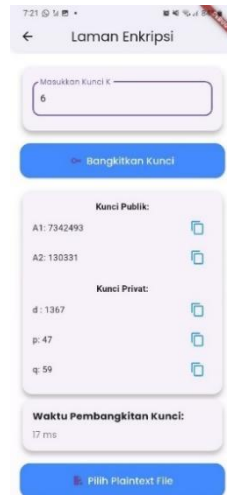
**Gambar 4.4** Halaman Kriptanalisis

## 4.2. Pengujian

Penelitian ini melakukan uji coba terhadap program yang dirancang dan dikembangkan menggunakan *framework* Flutter. Pengujian dilakukan dengan mengukur waktu eksekusi algoritma *Brute Force* dan *Kraitichik* pada kumpulan *ciphertext* dari algoritma kunci publik *Enhanced AA<sub>β</sub>*. Proses pengujian dilakukan pada perangkat Android fisik sebagai *platform* pengembangan. Pengukuran dilakukan dalam satuan milidetik (*ms*) untuk mengevaluasi efisiensi waktu masing-masing algoritma. Kompleksitas teoritis algoritma juga dianalisis untuk memberikan konteks performa lebih lanjut.

### 4.2.1. Pengujian Pembangkitan Kunci Publik dan Enkripsi

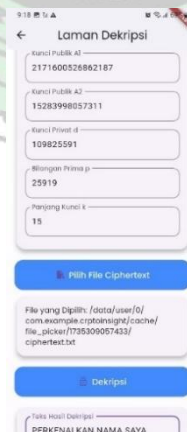
Pada fase ini akan dimulai dengan memasukkan panjang kunci  $k$ . Kemudian tombol “Bangkitkan Kunci” ditekan untuk membangkitkan kunci publik  $A_1$  serta  $A_2$  dan kunci privat  $d, p, q$ . Kunci publik dan privat yang diberi bisa diambil untuk digunakan pada laman dekripsi, sedangkan laman analisis hanya menggunakan kunci publik. Serta menampilkan waktu pembangkit kunci dalam satuan milidetik (*ms*).



**Gambar 4.5** Pengujian Pembangkitan Kunci Publik dan Enkripsi

#### 4.2.2. Pengujian Dekripsi Kunci Publik dan Privat

Pada fase ini akan dimulai dengan memasukkan kunci publik dan kunci privat serta panjang kunci  $k$  yang sama dengan pembangkitan kunci. Kemudian tombol “Pilih File *Ciphertext*” ditekan untuk mengambil file dalam format \*.txt atau \*.docx. kemudian tombol “Dekripsi” ditekan untuk mendekripsi *ciphertext* dengan kunci publik, kunci privat, dan panjang kunci  $k$ . Setelah berhasil, maka ditampilkan *plaintext*. Tombol “Simpan *plaintext* ke file” untuk menyimpan file *plaintext* ke dalam penyimpanan.



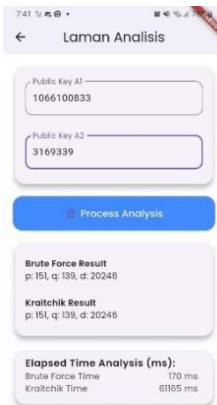
**Gambar 4.6** Pengujian Dekripsi Kunci Publik dan Privat

#### 4.2.3. Pengujian Pemfaktoran Kunci Publik

Pada fase ini akan dimulai dengan memasukkan kunci publik. Kemudian tombol “Proses Analisis” ditekan untuk memfaktorkan kunci publik



sampai menjadi kunci privat dari algoritma *Brute Force* dan *Kraitchik*. Setelah berhasil, maka ditampilkan kunci privat yaitu  $d$ ,  $p$ , dan  $q$ .



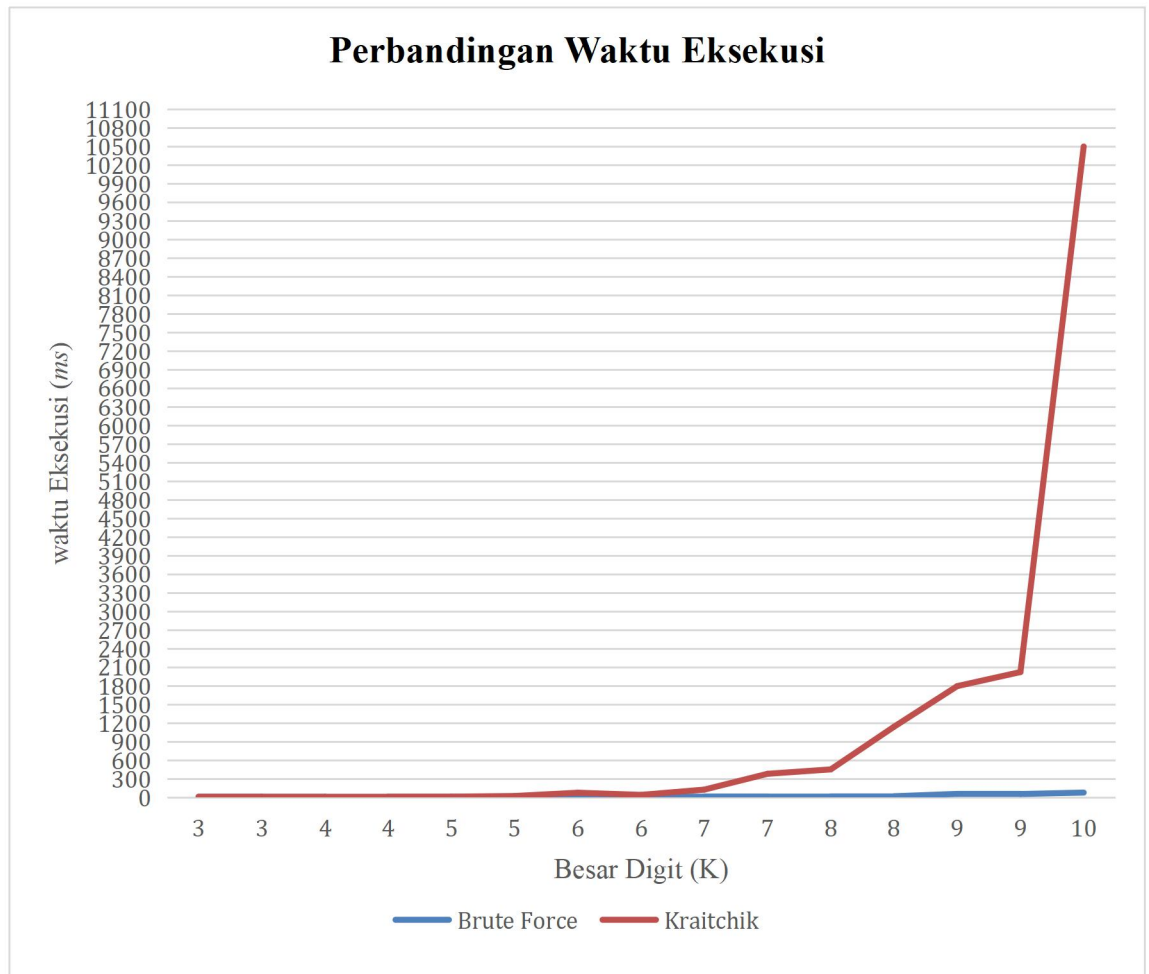
**Gambar 4.7** Pengujian Pemfaktoran Kunci Publik

#### 4.2.3.1. Pengujian Panjang Digit Kunci Publik

Pada tahap ini, pengujian dilakukan dengan panjang digit kunci publik yang berbeda. Pada Tabel 4.1 menunjukkan waktu eksekusi untuk proses pemfaktoran kedua metode dengan panjang digit kunci publik yang semakin besar.

**Tabel 4.1** Tabel Perbandingan Waktu Eksekusi

k	A <sub>1</sub>	A <sub>2</sub>	p	q	d	Waktu (ms)	
						<i>Brute Force</i>	<i>Kraitchik</i>
3	883	275	5	11	23	8	9
3	1101	637	7	13	32	7	8
4	6647	5687	11	47	15	5	7
4	9997	8959	17	31	120	7	10
5	1922342	11191	19	31	85	7	10
5	1257199	22103	31	23	501	8	21
6	16018373	163607	59	47	2550	8	74
6	8037476	86903	43	47	495	10	39
7	65089881	337747	71	67	601	12	124
7	131490615	1145159	127	71	4751	11	379
8	671427437	3690311	139	191	16870	14	450
8	963090656	9928183	211	233	44086	17	1138
9	4169142695	26491727	263	383	8357	55	1793
9	2532040355	28127903	271	383	23137	53	2020



**Gambar 4.8** Grafik Waktu Eksekusi Program

Berdasarkan tabel dan gambar di atas, terdapat beberapa informasi penting yang dapat diambil:

- Berdasarkan tabel, waktu eksekusi algoritma *Brute Force* cenderung lebih cepat dibandingkan dengan metode *Kraitchik* dalam memecahkan algoritma, terutama untuk nilai  $A_2$  yang lebih besar. Hal ini menunjukkan bahwa *Brute Force* lebih efisien dalam konteks waktu eksekusi karena tidak bergantung pada proses iteratif seperti *Kraitchik*.
- Pada metode *Kraitchik*, nilai  $k$ , yang merepresentasikan jumlah iterasi yang diperlukan untuk menemukan faktor, memiliki dampak signifikan terhadap waktu eksekusi. Nilai  $k$  cenderung meningkat seiring dengan pertumbuhan ukuran  $A_2$ , sehingga memperlambat proses dekripsi.

- c. Untuk nilai  $A_2$  kecil, metode *Brute Force* menunjukkan efisiensi yang sangat tinggi, karena jumlah percobaan untuk menemukan faktor jauh lebih sedikit dibandingkan metode *Kraitchik*.
- d. Meskipun lebih lambat, metode *Kraitchik* memiliki tingkat keakuratan yang sangat tinggi. Nilai  $p$ ,  $q$  dan  $d$  yang dihasilkan konsisten dan selalu sesuai dengan struktur  $A_2 = p^2q$ , mendukung validitas algoritma.
- e. Waktu eksekusi algoritma *Kraitchik* meningkat secara eksponensial seiring dengan pertumbuhan  $A_2$ . Hal ini disebabkan oleh kompleksitas iteratif yang meningkat untuk menemukan  $x$  dan  $y$ , terutama pada bilangan dengan faktor yang lebih besar.
- f. Faktor  $p$  dan  $q$  yang ditemukan oleh kedua algoritma konsisten dengan struktur  $A_2 = p^2q$ . Kedua algoritma menghasilkan dekomposisi yang valid, meskipun *Brute Force* lebih efisien dalam proses faktorisasi ini.
- g. Nilai  $d$ , yang dihasilkan dari invers modulo  $p$ , memainkan peran penting dalam keamanan kunci publik. Meningkatnya nilai  $d$  menunjukkan bahwa kunci menjadi lebih sulit untuk dipecahkan, sehingga mendukung desain algoritma yang lebih aman.

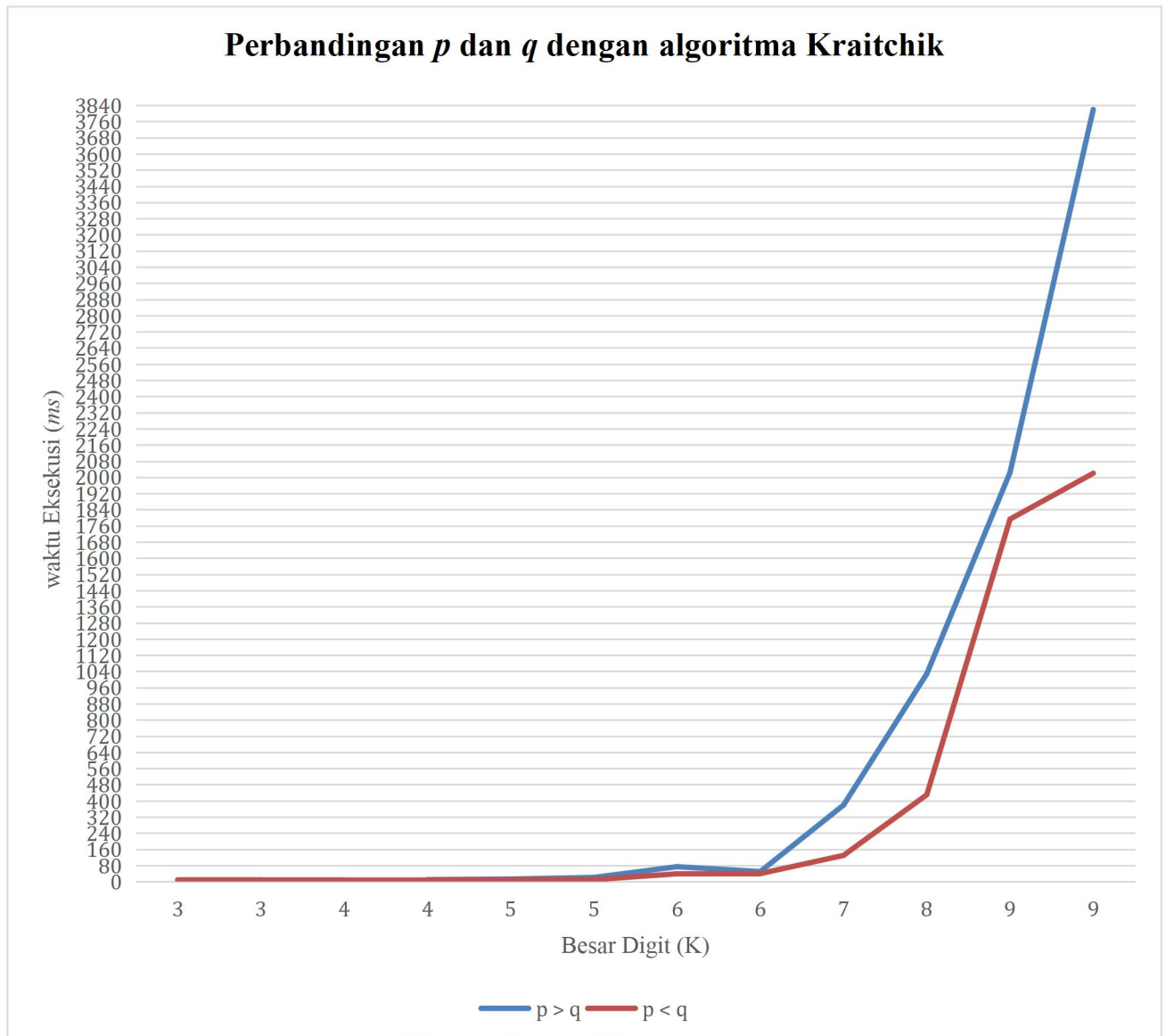
#### 4.2.3.2. Pengujian Perbandingan $p$ dan $q$ menggunakan Algoritma *Kraitchik*

Pada tahap ini, pengujian dilakukan dengan melihat perbandingan  $p$  dan  $q$ . Pada Tabel 4.2 menunjukkan waktu eksekusi untuk proses pemfaktoran algoritma *Kraitchik* dengan nilai  $p$  lebih besar dibanding  $q$ , dan ketika nilai  $p$  lebih kecil dibanding  $q$ .

**Tabel 4.2** Perbandingan  $p$  dan  $q$  pada Algoritma *Kraitichik*

k	A <sub>1</sub>	A <sub>2</sub>	$p > q$		d	Waktu (ms)	A <sub>1</sub>	A <sub>2</sub>	$p < q$		d	Waktu (ms)
			p	q					p	q		
3	1104	605	11	5	113	7	883	275	5	11	23	9
3	875	847	11	7	13	8	1101	637	7	13	32	8
4	5887	1859	13	11	6	8	6647	5687	11	47	15	7
4	88875	6137	19	17	293	10	9997	6647	17	23	107	8
5	1657289	10051	23	19	70	13	1925540	11191	19	31	120	10
5	1257199	22103	31	23	501	21	748838	8303	19	23	230	10
6	16018373	163607	59	47	2550	74	8037476	86903	43	47	2550	39
6	9839748	94987	47	43	1084	50	16066990	109091	43	59	1734	40
7	131490615	1145159	127	71	4751	379	86991064	642823	79	103	1762	130
8	728668962	6454963	199	163	31986	1028	671427437	3690311	139	191	16870	430
9	7472372925	20783803	271	283	18797	2024	4169142695	26491727	263	383	8357	1793
9	7549519991	41349523	367	307	110486	3819	2532040355	28127903	271	383	23137	2020





**Gambar 4.9** Grafik Perbandingan  $p$  dan  $q$  dengan *Kraitchik*

Berdasarkan tabel dan gambar di atas, terdapat beberapa informasi penting yang dapat diambil:

- Berdasarkan tabel, kondisi  $p < q$  cenderung menghasilkan nilai  $A_2$  yang lebih kecil dibandingkan dengan  $p > q$ . Hal ini menunjukkan bahwa proses perhitungan pada  $p < q$  lebih sederhana karena bilangan prima yang digunakan lebih kecil, sehingga mempermudah algoritma dalam menemukan faktor.
- Pada kondisi  $p > q$ , nilai  $k$  yang lebih tinggi merepresentasikan jumlah iterasi yang lebih besar untuk menemukan faktor  $p$  dan  $q$ . Sementara itu, pada kondisi  $p < q$ , meskipun nilai  $k$  juga



meningkat, iterasi yang diperlukan cenderung lebih stabil karena bilangan prima yang lebih kecil terlibat.

- c. Untuk nilai  $A_2$  kecil, kondisi  $p < q$  menunjukkan efisiensi yang lebih tinggi karena selisih antara  $p$  dan  $q$  lebih kecil. Hal ini membuat algoritma lebih cepat dalam menemukan solusi faktorisasi dibandingkan dengan kondisi  $p > q$ .
- d. Pada kondisi  $p > q$ , nilai  $d$  cenderung meningkat secara eksponensial seiring dengan pertumbuhan  $A_2$ . Hal ini menunjukkan bahwa selisih antara  $p$  dan  $q$  semakin besar pada kondisi ini, yang membuat proses dekomposisi lebih kompleks. Sebaliknya, pada  $p < q$ , pertumbuhan  $d$  relatif lebih terkontrol karena bilangan prima yang digunakan lebih kecil.
- e. Nilai  $d$ , yang dihasilkan dari selisih atau invers modulo  $p$ , memainkan peran penting dalam pengamanan kunci. Pada kondisi  $p > q$ , nilai  $d$  yang lebih besar menunjukkan peningkatan kesulitan dalam memecahkan kunci, sementara pada  $p < q$ , meskipun lebih kecil, tetap mendukung tingkat keamanan yang cukup baik.

#### 4.2.4. Kompleksitas Algoritma

##### 4.2.4.1. Kompleksitas Algoritma *Enhanced AA<sub>p</sub>*

##### Pembangkitan Kunci

Tabel ini menjelaskan langkah-langkah yang diambil dalam menghitung kompleksitas yang diterapkan pada proses pembangkitan kunci publik dan privat yang akan digunakan.

**Tabel 4.3** Kompleksitas Pembangkitan Kunci

No.	Kode Program	C	#	C . #
1	BigInt p = generatePrime(k);	$C_1$	n	$C_1 n$
2	BigInt q = generatePrime(k, exclude: p);	$C_2$	n	$C_2 n$
3	BigInt $A_2 = p * p * q$ ;	$C_3$	1	$C_3$
4	BigInt lowerBound = BigInt.one << (3 * k + 4);	$C_3$	1	$C_3$

5	<code>BigInt upperBound = BigInt.one &lt;&lt; (3 * k + 6);</code>	$C_3$	1	$C_3$
6	<code>Do {</code>	$C_4$	n	$C_4 n$
7	<code>A1 = randomBigInt(lowerBound, upperBound);</code>	$C_4$	n	$C_4 n$
8	<code>} while (gcd(A1, A2) != BigInt.one);</code>	$C_4$	n	$C_4 n$
9	<code>BigInt d = A1.modInverse(p * p);</code>	$C_5$	n	$C_5 n$
10	<code>return { 'A1': A1, 'A2': A2, 'd': d, 'p': p, 'q': q, 'k': k};</code>	$C_6$	1	$C_6$

Menurut tabel 4.3, perhitungan kompleksitas  $T(n)$  dapat dilakukan dengan cara sebagai berikut:

$$T(n) = C_1 n + C_2 n + 5C_3 + 3C_4 n + C_6$$

$$T(n) = (5C_3 + C_6)n^0 + (C_1 n + C_2 n + C_4 n + C_5)n^1$$

$$T(n) = \theta(n)$$

### Enkripsi

Tabel ini menjelaskan langkah-langkah yang diambil dalam menghitung kompleksitas yang diterapkan pada proses enkripsi yang akan digunakan.

**Tabel 4.4** Kompleksitas Enkripsi

No.	Kode Program	C	#	C . #
1	<code>BigInt lowerBound = BigInt.one &lt;&lt; (4 * k);</code>	$C_1$	1	$C_1$
2	<code>BigInt upperBound = BigInt.one &lt;&lt; (4 * k + 1);</code>	$C_1$	1	$C_1$
3	<code>BigInt range = upperBound - lowerBound;</code>	$C_1$	1	$C_1$
4	<code>BigInt t = lowerBound + randomBigInt(BigInt.zero, range);</code>	$C_1$	1	$C_1$
5	<code>BigInt c = A1 * BigInt.from(m).pow(2) + A2 * t;</code>	$C_1$	1	$C_1$
6	<code>return c;</code>	$C_1$	1	$C_1$
7	<code>throw Exception('Invalid plaintext character.');</code>	$C_1$	1	$C_1$

Menurut tabel 4.4, perhitungan kompleksitas  $T(n)$  dapat dilakukan dengan cara sebagai berikut:

$$T(n) = (7C_1)n^0$$

$$T(n) = \theta(1)$$

### Dekripsi

Tabel ini menjelaskan langkah-langkah yang diambil dalam menghitung kompleksitas yang diterapkan pada proses dekripsi yang akan digunakan.

**Tabel 4.5** Kompleksitas Dekripsi

No.	Kode Program	C	#	C . #
1	<code>BigInt w = (c * dPrime) % (p * p);</code>	$C_1$	1	$C_1$
2	<code>BigInt m_p = modularExponentiation(w, (p + BigInt.one) ~/ BigInt.from(4), p);</code>	$C_2$	$n$	$C_2 n$
3	<code>BigInt i = (w - (m_p * m_p)) ~/ p;</code>	$C_3$	1	$C_3$
4	<code>BigInt j = (i * modInverse(BigInt.from(2) * m_p, p)) % p;</code>	$C_4$	$n$	$C_4 n$
5	<code>BigInt m1 = m_p + j * p;</code>	$C_5$	1	$C_5$
6	<code>if (m1 &lt; (BigInt.one &lt;&lt; ((BigInt.from(2) * k - BigInt.one).toInt()))) {</code>	$C_6$	1	$C_6$
7	<code>  m = m1;}</code>	$C_6$	1	$C_6$
8	<code>else { m = (p * p) - m1; }</code>	$C_6$	1	$C_6$
9	<code>BigInt t = (c - A1 * (m * m)) ~/ A2;</code>	$C_7$	1	$C_7$
10	<code>String plaintext = String.fromCharCode(m.toInt());</code>	$C_7$	1	$C_7$
11	<code>return {'m': plaintext, 't': t};</code>	$C_7$	1	$C_7$

Menurut tabel 4.5, perhitungan kompleksitas  $T(n)$  dapat dilakukan dengan cara sebagai berikut:

$$T(n) = C_1 + C_2 n + C_3 + C_4 n + C_5 + 3C_6 + 3C_7$$

$$T(n) = (C_1 + C_3 + C_5 + 3C_6 + 3C_7)n^0 + (C_2 + C_4)n^1$$

$$T(n) = \theta(n)$$

#### 4.2.4.2. Kompleksitas Algoritma *Brute Force*

Tabel ini menjelaskan langkah-langkah yang diambil dalam menghitung kompleksitas yang diterapkan pada proses *Brute Force* yang akan digunakan.

**Tabel 4.6** Kompleksitas *Brute Force*

No.	Kode Program	C	#	C . #
1	for (BigInt p = BigInt.two; p <= sqrtBigInt(A2); p+= BigInt.one)	C <sub>1</sub>	n	C <sub>1</sub> n
2	if (A2 % p == BigInt.zero)	C <sub>2</sub>	n	C <sub>2</sub> n
3	BigInt q = A2 ~/ p * p;	C <sub>3</sub>	n	C <sub>3</sub> n
4	if ((p * p) != BigInt.one && q != BigInt.one && pSquared != A2 && q != A2){	C <sub>4</sub>	n	C <sub>4</sub> n
5	if ((p * p) * q == A2){	C <sub>5</sub>	n	C <sub>5</sub> n
6	if (isPrime(p)){	C <sub>6</sub>	n <sup>2</sup>	C <sub>6</sub> n <sup>2</sup>
7	BigInt d = modInverse(A1, pSquared);	C <sub>7</sub>	n <sup>2</sup>	C <sub>7</sub> n <sup>2</sup>
8	return {'p': p, 'q': q, 'd': d};}	C <sub>8</sub>	n	C <sub>8</sub> n

Menurut tabel 4.6, perhitungan kompleksitas  $T(n)$  dapat dilakukan dengan cara sebagai berikut:

$$T(n) = C_1n + C_2n + C_3n + C_4n + C_5n + C_6n^2 + C_7n^2 + C_8n$$

$$T(n) = (C_1 + C_2 + C_3 + C_4 + C_5 + C_8) n^1 + (C_6 + C_7)n^2$$

$$T(n) = \theta(n^2)$$

#### 4.2.4.3. Kompleksitas Algoritma *Kraitchik*

Tabel ini menjelaskan langkah-langkah yang diambil dalam menghitung kompleksitas yang diterapkan pada proses *Kraitchik* yang akan digunakan.

**Tabel 4.7** Kompleksitas *Kraitchik*

No.	Kode Program	C	#	C . #
1	BigInt x = sqrtBigInt(A2);	C <sub>1</sub>	1	C <sub>1</sub>
2	BigInt k = BigInt.one;	C <sub>1</sub>	1	C <sub>1</sub>
3	while (true) {	C <sub>2</sub>	n	C <sub>2</sub> n
4	BigInt kn = k * A2;	C <sub>3</sub>	n	C <sub>3</sub> n
5	BigInt z = (x * x) - kn;	C <sub>3</sub>	n	C <sub>3</sub> n

6	if (isPerfectSquare(z)) {	$C_4$	$n$	$C_4 n$
7	BigInt y = sqrtBigInt(z);	$C_5$	$n$	$C_5 n$
8	BigInt pSquared = x + y;	$C_6$	$n$	$C_6 n$
9	BigInt q = x - y;	$C_6$	$n$	$C_6 n$
10	if (pSquared * q == A2 && pSquared != BigInt.one && q != BigInt.one) {	$C_7$	$n$	$C_7 n$
11	BigInt p = sqrtBigInt(pSquared);	$C_8$	$n$	$C_8 n$
12	BigInt d = A1.modInverse(p);	$C_9$	$n^2$	$C_9 n^2$
13	return {'p': sqrtP, 'q': q, 'd': d};}	$C_{10}$	$n$	$C_{10} n$
14	k += BigInt.one;	$C_{11}$	$n$	$C_{11} n$
15	x += BigInt.one;	$C_{11}$	$n$	$C_{11} n$

Menurut tabel 4.7, perhitungan kompleksitas  $T(n)$  dapat dilakukan dengan cara sebagai berikut:

$$T(n) = 2C_1 + C_2n + 2C_3n + C_4n + C_5n + C_6n^2 + C_7n \\ + C_8n + C_9n^2 + C_{10}n + C_{11}n$$

$$T(n) = (2C_1)n^0 + (C_2 + 2C_3 + C_4 + C_5 + C_6 + C_7 + C_8 + C_{10}) \\ + C_{11}n^1 + (C_9)n^2$$

$$T(n) = \theta(n^2)$$

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1. Kesimpulan

Berdasarkan hasil analisis dan pengujian yang telah dilakukan pada penelitian perbandingan algoritma *Brute Force* dan *Kraitchik* dengan kriptanalisis pada algoritma kunci publik *Enhanced AA <sub>$\beta$</sub>*  yang diimplementasikan serta diuji pada perangkat berbasis Android, disimpulkan bahwa:

1. Algoritma *Brute Force* dan *Kraitchik* mampu memfaktorkan kunci publik  $(A_1, A_2)$  algoritma *Enhanced AA <sub>$\beta$</sub>*  dan menghasilkan faktor-faktor  $p$  dan  $q$  serta  $d$  yang sesuai, tetapi Algoritma *Kraitchik* kurang efisien jika panjang kunci  $k$  lebih besar dibandingkan dengan Algoritma *Brute Force*.
2. Algoritma *Kraitchik* menunjukkan performa yang lebih cepat dalam memfaktorkan  $A_2$  ketika nilai  $p$  lebih kecil dari  $q$  dibandingkan dengan ketika  $p$  lebih besar dari  $q$ . Selain itu, jika jarak antara  $p$  dan  $q$  berdekatan, algoritma ini dapat bekerja lebih efisien karena perbedaan kuadrat  $(x^2 - kn)$  yang dihasilkan menjadi lebih kecil, sehingga mempercepat proses identifikasi faktor. Hal ini disebabkan oleh pengaruh proses kuadrat  $p$  dalam perhitungan  $A_2$ , yang memberikan dampak signifikan terhadap waktu eksekusi.
3. Waktu eksekusi kedua algoritma meningkat secara signifikan dengan bertambahnya panjang kunci  $k$ . Namun, algoritma *Kraitchik* memperlihatkan lonjakan waktu yang jauh lebih tajam dibandingkan algoritma *Brute Force*, terutama pada panjang kunci  $k = 9$  hingga  $k = 10$ .
4. Untuk panjang kunci yang lebih kecil ( $k = 3$  hingga  $k = 5$ ), kedua algoritma relatif cepat dalam menemukan  $p$  dan  $q$ . Namun, ketika panjang kunci bertambah, algoritma *Kraitchik* menjadi kurang efisien dibandingkan *Brute Force*.
5. Nilai  $d$ , yang dihasilkan dari invers modulo  $p$ , memiliki peran penting dalam keamanan kunci publik. Pada kondisi  $p$  lebih besar dari  $q$ , nilai  $d$  cenderung lebih besar, menunjukkan tingkat kesulitan yang lebih tinggi untuk memecahkan kunci. Sebaliknya, pada  $p$  lebih kecil dari  $q$ , meskipun nilai  $d$  lebih kecil, kunci tetap memenuhi standar keamanan yang baik.

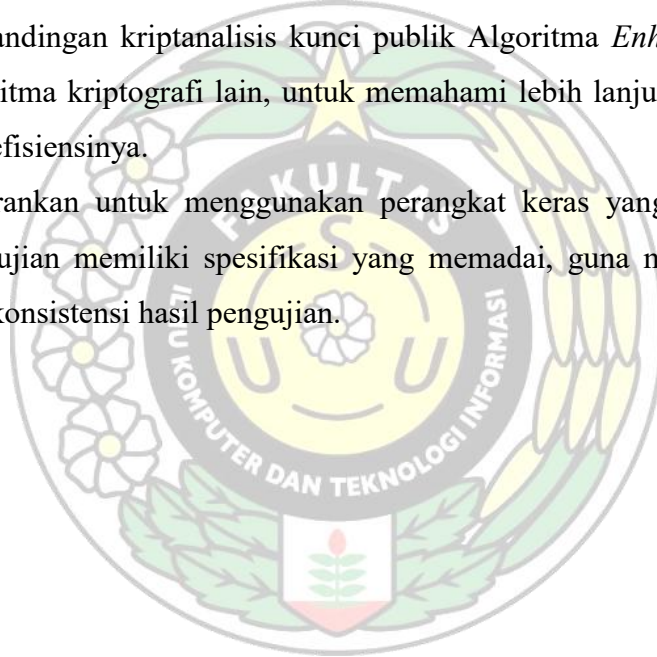


Dengan demikian, algoritma *Enhanced AA <sub>$\beta$</sub>*  mampu menghasilkan kunci yang aman dalam kedua kondisi.

## 5.2. Saran

Berdasarkan hasil temuan dari penelitian ini, terdapat beberapa rekomendasi yang dapat digunakan untuk penelitian di masa mendatang, yaitu sebagai berikut:

1. Disarankan untuk melakukan penelitian lebih lanjut untuk Algoritma pemfaktoran lain mengevaluasi kekuatan, kelemahan, efektivitas, dan efisiensi masing-masing algoritma.
2. Disarankan untuk melakukan penelitian lebih lanjut agar mengetahui perbandingan kriptanalisis kunci publik Algoritma *Enhanced AA <sub>$\beta$</sub>*  dengan algoritma kriptografi lain, untuk memahami lebih lanjut tingkat keamanan dan efisiensinya.
3. Disarankan untuk menggunakan perangkat keras yang digunakan untuk pengujian memiliki spesifikasi yang memadai, guna menjaga keakuratan dan konsistensi hasil pengujian.



## DAFTAR PUSTAKA

- Asbullah, M. A., & Ariffin, M. R. K. (2016). *Analysis on the  $AA_\beta$  Cryptosystem*. In 5th International Cryptology and Information Security Conference (pp. 41-48).
- Mahad, Z., Asbullah, M. A., & Ariffin, M. R. K. (2019). *Enhanced  $AA_\beta$  cryptosystem: A comparative analysis*. *Malaysian Journal of Mathematical Sciences*, 13(2), 187-200.
- Budiman, M. A., & Irvida, T. A. (2023). *Securing text using Rabin-p public key cryptosystem and Spritz algorithm in a hybrid cryptosystem: a tutorial*. In *Journal of Physics: Conference Series* (p. 012010).
- Asbullah, M. A., & Ariffin, M. R. K. (2016). *Design of Rabin-like cryptosystem without decryption failure*. *Malaysian Journal of Mathematical Sciences*, 10(1), 1-18.
- Muchlis, B. S., Budiman, M. A., & Rachmawati, D. (2017). Teknik Pemecahan Kunci Algoritma Rivest Shamir Adleman (RSA) dengan Metode *Kraitchik*. *Jurnal Sinkron*, 2(2), 49-56.
- Asbullah, M. A., Ariffin, M. R. K., & Mahad, Z. (2018). *A Fast and Efficient Design for  $AA_\beta$  Cryptosystem*. *International Journal of Cryptology Research*, 8(1), 1-16.
- Lydia, M. S., Budiman, M. A., & Rachmawati, D. (2021). *Factorization of small RPrime RSA modulus using Fermat's difference of squares and Kraitchik's algorithms in Python*. *Journal of Theoretical and Applied Information Technology*, 99(11), 1-18.
- Rechberger, C. (2013). *On Brute-force-Like Cryptanalysis: New Meet-in-the-Middle Attacks in Symmetric Cryptanalysis*. In Kwon, T., Lee, M. K., & Kwon, D. (Eds.). *Information Security and Cryptology – ICISC 2012*. Lecture Notes in Computer Science, vol. 7839. Springer, Berlin, Heidelberg.
- Ariffin, M. R. K., Asbullah, M. A., & Abu, N. A. (2021). *A New Efficient Asymmetric Cryptosystem for Large Data Sets*. Al-Kindi Cryptography Research Laboratory, Institute for Mathematical Research, Universiti Putra Malaysia (UPM).
- Ghaffar, A. H. A., & Ariffin, M. R. K. (2014). *Timing Attack Analysis on  $AA_\beta$  Cryptosystem*. *Journal of Computer and Communications*, 2, 1-9. Published Online March 2014 in SciRes.

- Adnan, S. F., Mat Isa, M. A., & Hashim, H. (2018). *RF Simulations for  $AA_\beta$  Cryptosystem, an Asymmetric Encryption Scheme*. Indonesian Journal of Electrical Engineering and Computer Science, 11(2), 542–548.
- Purba, D. S. (2021). Kriptanalisis Kunci Publik Algoritma Rabin Menggunakan Metode *Kraitichik*. Jurnal Sains Dan Teknologi Istp, 11(2), 205–213.
- Marouf, I., Asad, M. M., & Al-Haija, Q. A. (2017). *Reviewing and analyzing efficient GCD/LCM algorithms for cryptographic design*. International Journal of New Computer Architectures and their Applications, Society of Digital Information and Wireless Communication, 7(1), 1-7.
- Egecioglu, O., & Koç, C. K. (1990). *Fast modular exponentiation*. Communication, Control, and Signal Processing, 188-194.
- Mrabet, A., El-Mrabet, N., Bouallegue, B., Mesnager, S., & Machhout, M. (2017, May). *An efficient and scalable modular inversion/division for public key cryptosystems*. In 2017 International Conference on Engineering & MIS (ICEMIS) (pp. 1-6). IEEE.
- Somsuk, K. (2020). *The new integer factorization algorithm based on fermat's factorization algorithm and euler's theorem*. Int J Electr Comput Eng, 10(2), 1469–1476.
- Conrad, K. (2011). *The miller–rabin test*. Encyclopedia of Cryptography and Security.
- Azizah, U. N. (2013). *Perbandingan Detektor Tepi Prewitt dan Detektor Tepi Laplacian berdasarkan Kompleksitas Waktu dan Citra Hasil* (Doctoral dissertation, Universitas Pendidikan Indonesia).