TUGAS AKHIR - KM184801

KOMBINASI ALGORITMA MODIFIKASI CAESAR CIPHER DAN RIVEST SHAMIR ADLEMAN UNTUK ENKRIPSI DAN DEKRIPSI PESAN

RENJIRO OWEN
NRP. 06111640000067

Dosen Pembimbing:
Dr. Darmaji, S.Si., M.T.

DEPARTEMEN MATEMATIKA
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR - KM184801

KOMBINASI ALGORITMA MODIFIKASI CAESAR CIPHER DAN RIVEST SHAMIR ADLEMAN UNTUK ENKRIPSI DAN DEKRIPSI PESAN

RENJIRO OWEN
NRP. 06111640000067

Dosen Pembimbing:
Dr. Darmaji, S.Si., M.T.

DEPARTEMEN MATEMATIKA
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT - KM184801

COMBINATION OF MODIFIED CAESAR CIPHER AND RIVEST SHAMIR ADLEMAN ALGORITHMS FOR MESSAGE ENCRYPTION AND DECRYPTION

RENJIRO OWEN
NRP. 06111640000067

Supervisor:
Dr. Darmaji, S.Si., M.T.

DEPARTMENT OF MATHEMATICS
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember
Surabaya 2020

LEMBAR PENGESAHAN

KOMBINASI ALGORITMA MODIFIKASI CAESAR CIPHER DAN RIVEST SHAMIR ADLEMAN UNTUK ENKRIPSI DAN DEKRIPSI PESAN

COMBINATION OF MODIFIED CAESAR CIPHER AND RIVEST SHAMIR ADLEMAN ALGORITHMS FOR MESSAGE ENCRYPTION AND DECRYPTION

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Matematika
Pada bidang studi Ilmu Komputer
Program Studi S1 Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
RENJIRO OWEN
NRP. 06111640000067

Menyetujui,
Dosen Pembimbing

Dr. Darmaji, S.Si., M.T.
NIP. 19691015 199412 1 001

Mengetahui,
Kepala Departemen Matematika
FSAD ITS

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS SAINS DAN ANALITIKA DATA DEPARTEMEN MATEMATIKA

Subchan, Ph.D.
NIP. 19710513 199702 1 001

Surabaya, 19 Agustus 2020

KOMBINASI ALGORITMA MODIFIKASI CAESAR CIPHER DAN RIVEST SHAMIR ADLEMAN UNTUK ENKRIPSI DAN DEKRIPSI PESAN

Nama	:	Renjiro Owen
NRP	:	06111640000067
Departemen	:	Matematika FSAD ITS
Dosen Pembimbing	:	Dr. Darmaji, S.Si., M.T.

Teknologi telepon pintar berkembang sangat pesat pada jaman sekarang. Telepon pintar merupakan salah satu alat untuk mengirim pesan dengan efisien dan cepat. Ada banyak sistem operasi yang menjadi tulang punggung dari telepon pintar, salah satunya adalah sistem operasi android. Pesatnya laju perkembangan ini menyebabkan komunikasi yang lebih baik.

Komunikasi rahasia tidak boleh dibaca oleh orang ketiga yang tidak berhak.Akan tetapi banyak komunikasi bersifat rahasia yang terbaca oleh pihak yang tidak berhak untuk melihat. Selain dari sifat kerahasiaan data, integritas data juga harus dijaga agar tidak dapat diubah dengan mudah. Oleh karena itu diperlukan algoritma komputer untuk menjaga kerahasiaan dan integritas data. Banyak algoritma yang sudah dikembangkan sejak dahulu, seperti Alfine Cipher, Hill Cipher, Cipher Blok Chaining(CBC), dan sebagainya. Salah satu cara untuk menambah keamanan data adalah dengan menggabungkan dua atau lebih algoritma yang sudah lazim dikenal. Dua algoritma yang digabungkan dalam Tugas Akhir ini adalah algoritma Modifikasi Caesar Cipher dan algoritma RSA. Algoritma ini berhasil diimplementasikan di Android. Penggabungan dua algoritma ini meningkatkan keamanan data dan waktu dari enkripsi dan dekripsi. Algoritma MCCRSA baik digunakan ketika teks yang dikirim merupakan teks yang panjang.

***Kata Kunci* : Kriptografi, Caesar Cipher, RSA, Android**

COMBINATION OF MODIFIED CAESAR CIPHER AND RIVEST SHAMIR ADLEMAN ALGORITHMS FOR MESSAGE ENCRYPTION AND DECRYPTION

Name	:	Renjiro Owen
NRP	:	06111640000067
Department	:	Mathematics FSDA ITS
Supervisor	:	Dr. Darmaji, S.Si., M.T.

The technology of smartphones is growing at a very rapid rate nowdays. A smartphone itself, is a device with numerous functions, one of which is it can be used to text other people in a fast and convenient fashion. There are a lot of Operating Systems (OS) that are the main pillar of smartphones, one of which is the Android OS. The rapid pace of smartphone's development offers improved, faster, and more secure communicating experience between people.

Regarding the security aspect, private conversations between users are not supposed to be misused, let alone be read at all by third parties who do not have the right to do so.However, a large number of private communications are being used by irresponsible parties who use it for their own interest. Apart from the hidden nature of data, the data's integrity must also be maintained so that it cannot be modified easily without the owner's consent. Therefore, an algorithm is needed to maintain the hidden nature and integrity of data. Lots of algorithms have been developed long ago, such as Alfine Cipher, Hill Cipher, Cipher Block Chaining (CBC), and many others. One way to increase data security is to combine two or more algorithms in which its secureness have already been acknowledged. This final project aims to combined two algorithms that are the Modification Caesar Cipher algorithm and the RSA algorithm. This algorithm is

succesfully implemented on Android. Combined algorithm increases data security and encryption decryotion running time. MCCRSA works great when the text sent is a long text.

Keywords* : *Cryptography, Caesar Cipher, RSA, Android

KATA PENGANTAR

Syukur dan puji saya haturkan ke hadirat Tuhan Yang Maha Esa karena atas berkat dan perlindungan-Nya, Tugas Akhir ini bisa terselesaikan dengan baik dan tepat waktu.

Tugas Akhir merupakan salah satu dari mata kuliah wajib pada Departemen Matematika FSAD ITS dan menjadi syarat kelulusan. Mata kuliah ini ditujukan kepada mahasiswa semester 7 atau 8 yang akan menyelesaikan studi S1.

Tugas Akhir ini disusun dalam rangka untuk memenuhi syarat memperoleh gelar Sarjana Sains (S.Mat.) pada bidang minat Ilmu Komputer. Dalam Tugas Akhir ini, penulis memberikan hasil penelitian tentang performa algoritma MCCRSA (Modifikasi Caesar Cipher Rivest Shamir Adleman) dalam melakukan enkripsi dan dekripsi pesan pada telepon pintar berbasis android. Pada Tugas Akhir ini, algoritma MCCRSA akan dibandingkan juga dengan algoritma RSA dan MCC. Tujuannya untuk mengetahui seberapa cepat algoritma MCCRSA. Judul dari Tugas Akhir ini adalah

KOMBINASI ALGORITMA MODIFIKASI CAESAR CIPHER DAN RIVEST SHAMIR ADLEMAN UNTUK ENKRIPSI DAN DEKRIPSI PESAN

Penyusunan Tugas Akhir ini tidak terlepas dari dukungan berbagai pihak. Oleh sebab itu, tak lupa penulis menyampaikan ucapan terima kasih kepada:

1. Orang tua penulis, bapak Hendro dan ibu Rosmawati, serta adik - adik penulis, Reifira Jasmine dan Allegra Pheony, atas inspirasi, dukungan, dan bantuannya selama perjalanan hidup penulis hingga Tugas Akhir ini selesai.

2. Bapak kepala Departemen Matematika FSAD ITS yang telah memberikan motivasi dan dukungan kepada penulis.

3. Bapak dan ibu Sekretaris Departemen Matematika FSAD ITS, yang telah membantu penulis dalam hal menyelesaikan kebutuhan administrasi selama periode pengerjaan Tugas Akhir.

4. Bapak Dr. Darmaji, S.Si., M.T., selaku dosen pembimbing atas bimbingan, saran, kritik, motivasi, dan pengalaman yang tak terlupakan selama proses pengerjaan Tugas Akhir.

5. Bapak Dr. Mahmud Yunus, M.Si., Bapak Moh. Iqbal, S.Si., M.Si., dan Bapak Dr. Imam Mukhlash, S.Si., M.T., selaku dosen penguji yang sudah memberikan masukan yang membangun.

6. Bapak Prof. Dr. Basuki Widodo, M.Sc. dan Dr. Budi Setiyono, S.Si., M.T. sebagai dosen wali yang sudah memberikan arahan selama kegiatan perkuliahan di ITS.

7. Seluruh pihak yang tidak bisa disebutkan satu - persatu, yang telah memberikan saran, dukungan dan motivasi dalam menyelesaikan Tugas Akhir ini. Penulis mengucapkan terima kasih yang sangat dalam, atas doa dan semangat yang diberikan kepada penulis.

Penulis juga mengharapkan kritik dan saran yang membangun dari berbagai pihak untuk perbaikan isi Tugas Akhir ini. Segala kritik dan saran akan penulis terima dengan senang hati.

Surabaya, 13 Mei 2020

Renjiro Owen

Daftar Isi

HALAMAN JUDUL **i**

LEMBAR PENGESAHAN **v**

ABSTRAK **vii**

ABSTRACT **ix**

KATA PENGANTAR **xi**

Daftar Isi **xiii**

Daftar Gambar **xv**

Daftar Tabel **xvii**

1 PENDAHULUAN **1**

1.1 Latar Belakang Masalah 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Sistematika Penulisan 4

2 TINJAUAN PUSTAKA **5**

2.1 Penelitian Terdahulu 5
2.1.1 Kombinasi Algoritma Caesar Cipher Algoritma dan Algoritma Rivest Shamir Adleman untuk mengamankan data dokumen dan pesan teks 5

2.1.2 Penyandian Pesan Teks Berbasis Android Menggunakan Metode Rivest, Shamir, Adleman (RSA) 5
2.2 RSA 6
2.2.1 Pembuatan Kunci 6
2.2.2 Enkripsi dan Dekripsi Pesan 7
2.3 Caesar Cipher 7
2.4 MCC 8
2.5 MCCRSA 9
2.6 Android 9

3 METODE PENELITIAN 11
3.1 Spesifikasi Perangkat 11
3.2 Tahapan Penelitian Tugas Akhir 12

4 HASIL DAN PEMBAHASAN 15
4.1 Pengembangan Aplikasi 15
4.2 Pengkodean Pesan 17
4.3 Implementasi Algoritma MCC 18
4.4 Implementasi Algoritma RSA 23
4.5 Implementasi Algoritma MCCRSA 26
4.6 Hasil Implementasi 30
4.6.1 Uji MCCRSA 35
4.6.2 Waktu Enkripsi dan Waktu Dekripsi 35
4.6.3 Keamanan Data 38
4.6.4 Volume Bit 41

5 KESIMPULAN DAN SARAN 43

DAFTAR PUSTAKA 45

LAMPIRAN 47

Daftar Gambar

3.1 Diagram alir penyelesaian masalah Tugas Akhir. . . 13
4.1 Desain aplikasi 16
4.2 Potongan kode untuk pemetaan karakter ke byte array 17
4.3 Potongan kode untuk pengambilan pesan 18
4.4 Potongan kode untuk enkripsi pesan 19
4.5 Potongan kode untuk pengiriman pesan 20
4.6 Potongan kode untuk penerimaan pesan 20
4.7 Potongan program untuk membaca pesan dan kunci 21
4.8 Potongan program untuk dekripsi pesan 22
4.9 Potongan program untuk pemetaan balik 22
4.10 Diagram alir desain aplikasi algoritma MCC. 23
4.11 Potongan program untuk pengambilan pesan dan kunci 24
4.12 Potongan program untuk enkripsi RSA 24
4.13 Potongan program untuk pengambilan kunci privat. . 25
4.14 Potongan program untuk dkripsi RSA 26
4.15 Diagram alir desain aplikasi algoritma RSA yang diusulkan. 27
4.16 Diagram alir desain aplikasi algoritma MCCRSA. . 29
4.17 Pesan dan Kunci untuk MCC. 30
4.18 Tombol ENKRIP ditekan 31
4.19 Pengiriman pesan cipher. 32
4.20 Tombol DEKRIP ditekan 33
4.21 Graph waktu enkripsi dan dekripsi MCC 37
4.22 Graph waktu enkripsi dan dekripsi RSA 38
4.23 Graph waktu enkripsi dan dekripsi MCCRSA 39

Daftar Tabel

BAB I
PENDAHULUAN

Pada bab pendahuluan ini, diuraikan latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan dari Tugas Akhir.

1.1 Latar Belakang Masalah

Teknologi telepon pintar berkembang sangat cepat. Perkembangan ini dimulai dengan ponsel yang hanya bisa digunakan sebagai perantara berbicara dan mengirim pesan lalu berkembang menjadi telepon pintar yang memiliki berbagai fungsi seperti multimedia, merekam video, bermain game, dan lainnya. Terdapat berbagai *Operating System*(OS) yang digunakan oleh telepon pintar, salah satunya adalah android. Android merupakan OS yang mempunyai pasar terbesar. Pada Desember 2019, pangsa pasar dari android mencapai 93,22%[1].

Seiring berkembangnya teknologi android, pengguna internet juga berkembang sangat pesat. Kejadian ini disebabkan karena kemudahan yang disediakan oleh android untuk mengakses internet. Internet di Indonesia juga mengalami kenaikan dimana pada tahun 2017 pengguna internet menyentuh angka 143,26 juta jiwa dan pada tahun 2018 menjadi 171,17 juta penduduk dari penduduk indonesia 264,16 juta orang[2]. Perkembangan ini mengakibatkan potensi *cybercrime* semakin besar. Pada tahun 2007, University of Maryland mengungkapkan bahwa peretas melakukan serangan setiap 39 detik[3]. Para peretas menggunakan *scripts* otomatis untuk mencari dan menyerang sistem yang terbuka[3]. Pada tahun 2018, cybercrime menghasilkan uang sekitar 1,5 triliun dollar[4]. Masalah ini menjadi aspek penting dalam menjaga penyimpanan data, terutama data yang disimpan dalam bentuk digital. Telah diciptakan berbagai macam algoritma komputer untuk menjaga data. Algoritma - algortima ini dipelajari di ilmu kriptografi.

Kriptografi merupakan salah satu bidang sains yang mempelajari cara mengamankan pesan. Ilmu ini mulai diteliti karena banyaknya peretas yang semakin meningkatkan kuantitas dan kualitasnya. Kriptografi dapat dibagi menjadi 2, yaitu kriptografi klasik dam modern. Caesar Cipher adalah suatu algoritma klasik yang menyusun kembali text awal (*plain text*) dengan cara menggeser semua karakter pada *plain text* sebanyak 3 kali sesuai urutan alfabet. Caesar Cipher kemudian dikembangkan dengan berbagai macam ide yang disebut Modifikasi Caesar Cipher (MCC). Kelemahan metode ini dapat dengan mudah dan cepat dipecahkan oleh komputer terkini menggunakan metode *brute force*. Selain dari algoritma klasik, ada algoritma modern yang menggunakan 2 kunci berbeda, yaitu kunci privat dan kunci publik. Banyak algoritma kunci publik yang pernah dibuat, salah satu algoritma yang terkenal adalah algoritma RSA. Letak kesulitan algoritma ini adalah pemfaktoran hasil kali 2 bilangan prima yang relatif besar[5]. Peningkatan keamanan data juga dapat dilakukan dengan cara penggabungan dua atau lebih algoritma. Perkembangan teknolgi dan meningkatnya jumlah peretas inilah yang menjadi salah satu alasan algoritma - algoritma harus diteliti ulang agar semakin tinggi keamanannya. Maka dari itu, dalam tugas akhir ini akan dikerjakan implementasi algorima Modifikasi Caesar Cipher RSA (MCCRSA) untuk layanan pesan berbasis android.

1.2 Rumusan Masalah

Berjalan dari latar belakang yang sudah dipaparkan sebelumnya, bisa diformulasikan masalah tugas akhir adalah sebagai berikut.

1. Berapa perbandingan *running time* dari MCCRSA, RSA, dan MCC?

2. Bagaimana keamanan dari MCCRSA?

3. Berapa volume bit pesan cipher dan pesan awal dari MCCRSA?

4. Bagaimana desain dan implementasi enkripsi dekripsi MCCR-SA di aplikasi Android?

1.3 Batasan Masalah

Untuk membatasi masalah pada tugas akhir ini, digunakan beberapa batasan masalah sebagai berikut.

1. Aplikasi ini hanya akan mengirim pesan karakter, huruf, dan tanda baca.
2. Tool yang digunakan adalah Android Studio.
3. Aplikasi ini hanya bisa dijalankan sistem operasi Android.

1.4 Tujuan

Tujuan pada Tugas Akhir adalah sebagai berikut.

1. Mengetahui perbandingan *running time* dari MCCRSA, RSA, dan Caesar.
2. Mengembangkan aplikasi pengirim pesan dengan enkripsi dekrip-si MCCRSA.
3. Mengetahui kekuatan dari MCCRSA.
4. Mengetahui besar volume bit dari MCCRSA.

1.5 Manfaat

Manfaat dari tugas akhir ini adalah memberikan aplikasi ber-basis android dengan menggunakan algoritma enkripsi dekripsi MCCRSA.

1.6 Sistematika Penulisan

Sistematika penulisan Tugas Akhir ini adalah sebagai berikut.

1. BAB I PENDAHULUAN
   Pada bab ini dipaparkan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan dari Tugas Akhir ini.

2. BAB II TINJAUAN PUSTAKA
   Pada bab ini diberikan dasar teori yang digunakan dalam pengerjaan Tugas Akhir.

3. BAB III METODE PENELITIAN
   Pada bab ini dipaparkan langkah - langkah yang dilakukan dalam pengerjaan Tugas Akhir.

4. BAB IV HASIL DAN PEMBAHASAN
   Pada bab ini dipaparkan hasil yang diperoleh selama pengerjaan Tugas Akhir beserta dengan pembahasan mengenai hasil tersebut.

5. BAB V KESIMPULAN DAN SARAN
   Pada bab ini diberikan kesimpulan berdasarkan hasil yang diperoleh selama pengerjaan Tugas Akhir dan saran untuk penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

2.1.1 Kombinasi Algoritma Caesar Cipher Algoritma dan Algoritma Rivest Shamir Adleman untuk mengamankan data dokumen dan pesan teks

Indra dkk melakukan penelitian untuk menambahkan tingkat keamanan dari pengirimanan sebuah pesan. Penelitian ini menggunakan 2 metode yang berbeda yang kemudian yaitu metode Caesar Cipher dan dilanjutkan dengan metode Rivest-Shamir-Adleman(RSA). Metode Caesar Cipher merupakan salah satu metode lama yang digunakan untuk mengubah suatu text ke kode rahasia (pesan cipher)[6]. Sedangkan RSA merupakan metode kriptografi modern yang menggunakan kunci publik untuk enkripsi dan kunci privat untuk dekripsi.

Kombinasi dari 2 metode ini menambah keamanan dari pesan cipher dibandingkan dengan penggunaan 1 metode. Kalkulasi yang digunakan dengan algoritma Caesar Cipher oleh algoritma RSA membuat tingkat keamanan data semakin tinggi.

2.1.2 Penyandian Pesan Teks Berbasis Android Menggunakan Metode Rivest, Shamir, Adleman (RSA)

Iwan dkk melakukan penelitian untuk mengimplementasikan metode RSA pada aplikasi berbasis android, dimana RSA merupakan salah satu metode enkripsi modern yang sangat popular. Android merupakan salah satu OS yang digunakan di telepon pintar digunakan dalam penelitian ini. Metode RSA dapat diaplikasikan di Android[7].

2.2 RSA

Algoritma RSA adalah salah satu algoritma kriptografi modern yang mempunyai tingkat kerumitan yang tinggi. Algortima ini merupakan algoritma terpopuler dari sekian banyak algoritma kriptografi kunci publik yang pernah dibuat. Keamanan dari algorit-ma ini terletak pada sulitnya memfaktorkan bilangan besar menjadi faktor - faktor prima [5]. Jadi dapat disimpulkan bahwa jika tidak ada algoritma yang dengan cepat memfaktorkan bilangan bulat besar menjadi faktor - faktor primanya, maka selama itu pula keamanan algoritma RSA masih terjamin[5]. Runtun dari algoritma ini adalah pembuatan kunci, enkripsi, dan dekripsi.

Algoritma ini memiliki beberapa notasi :

(a)	p dan q bilangan prima	(rahasia)
(b)	$n = p.q$	(tidak rahasia)
(c)	$\phi(n) = (p-1)(q-1)$	(rahasia)
(d)	e (kunci enkripsi)	(tidak rahasia)
(e)	d (kunci dekripsi)	(rahasia)
(f)	m (pesan awal)	(rahasia)
(g)	c (pesan cipher)	(tidak rahasia)

2.2.1 Pembuatan Kunci

Pembuatan kunci dari RSA mempunyai beberapa langkah :

(a) Pilih dua bilangan prima sembarang p dan q, disarankan agar $p \neq q$

(b) Menghitung nilai $n = p.q$

(c) Menghitung nilai $\phi(n) = (p-1)(q-1)$

(d) Pilih kunci publik e yang relatif prima dengan $\phi(n)$

(e) Menghitung nilai d dengan rumus $d.e \equiv 1(\mod \phi(n))$

Hasil dari algoritma diatas adalah kunci publik pasangan (e,n) dan kunci privat d. Kunci publik dapat diumumkan kepada siapapun yang akan mengirim pesan. Sedangkan kunci privat harus dijaga agar pesan rahasia tidak dapat dibaca pihak ketiga.

2.2.2 Enkripsi dan Dekripsi Pesan

Enkripsi pesan diawali dengan penerima pesan membangkitkan kunci publik (e,n) dan kunci privat d. Penerima pesan kemudian mengirimkan kunci publik ke pengirim pesan. Pengirim kemudian mengubah pesan yang ingin dikirim ke angka m, dimana $m <$ kunci publik n. Selanjutnya pengirim menghitung $c = m^e \mod n$ dan mengirim c ke tujuan.

Pesan yang telah terkirim kemudian didekripsi oleh penerima pesan. Penerima menerima pesan dalam bentuk kode c dari pengirim. Kode ini kemudian didekripsi menggunakan rumus $m = c^d \mod n$. Lalu bilangan m ini diubah menjadi pesan awal.

2.3 Caesar Cipher

Dalam Caesar Cipher, tiap huruf disubtitusi dengan huruf berjarak 3 di sebelah kanannya dalam susunan alphabet. Contoh kata “halo”, dalam hal ini “h” digeser sebanyak 3 menjadi “k”, “a” digeser sebanyak 3 menjadi "d" dan seterusnya menjadi “kdor”. Penggeseran ini bersifat siklik semisal setiap huruf digeser sebanyak 2, maka huruf y menjadi b, huruf z menjadi c[5].

Dekripsi dan enkripsinya dalam model matematika sangat mudah. Pertama pesan awal dipetakan menjadi integer. Kemudian untuk setiap huruf di pesan awal (P) ditambahkan dengan 3 sebagai kunci menjadi pesan cipher (C). Jumlah dari huruf yang dapat digunakan adalah 26 sehingga modulus yang dipakai adalah modulus 26. Maka didapat rumus enkripsi

$$C = (P+3) \mod 26$$

C kemudian dikirim ke penerima. Pesan cipher yang diterima oleh penerima kemudian melakukan operasi dekripsi, yaitu dengan mengu-rangi C dengan 3 dalam modulus 26, didapat rumus dekripsi

$$P = (C - 3) \mod 26$$

Selanjutnya P dipetakan balik menjadi pesan yang diharapkan.

2.4 MCC

MCC merupakan metode untuk menangani kelemahan dari metode Caesar Cipher. MCC yang digunakan dalam tugas akhir ini adalah dengan mensubstitusikan dengan huruf berjarak n disebelah kanannya dari susunan alfabet, dimana n adalah hasil kali sebarang bilangan bulat dan posisi dari setiap huruf dari huruf pertama.

Enkripsi dan dekripsi dari metode ini tidak rumit. Pesan yang diinginkan dienkripsi menjadi integer dan key R. Kemudian untuk setiap huruf di pesan awal (P) ditambahkan dengan R dikalikan dengan posisi huruf pada pesan awal (T). Berbeda dengan Caesar Cipher, teks yang digunakan adalah numerik, huruf, dan tanda baca. Berdasarkan tabel ASCII, jumlah karakter yang dapat digunakan adalah 128[8]. Maka didapat rumus enkripsi

$$C = (P + R * T) \mod 128$$

C kemudian dikirim ke penerima. Pesan cipher yang diterima oleh penerima kemudian melakukan operasi dekripsi, yaitu dengan mengurangi C dengan $R * T$ dalam modulus 128, didapat rumus dekripsi

$$P = (C - R * T) \mod 128$$

Selanjutnya P diubah menjadi pesan yang diharapkan. Contoh kata adalah "ABC" dimana konversi bitnya adalah 65 66 67. Menggunakan rumus MCC kunci 1, maka pesan cipher dari "A" adalah $(65 + 1 * 1) \mod 128 = 66$ dan konversi bitnya "B". Menggunakan cara yang sama didapatkan "B" menjadi "D" dan "C" menjadi "F". Maka pesan cipher dari "ABC" adalah "BDF".

2.5 MCCRSA

Kombinasi dari MCC dan RSA bertujuan untuk menutup kelemahan MCC yang berbasis substitusi urutan alfabet. Kombinasi ini dapat berkerja dengan menggunakan algoritma MCC terlebih dahulu diikuti dengan algoritma RSA. Untuk dekripsinya akan dilakukan berkebalikan, yaitu dengan metode dekripsi RSA dilanjutkan dengan metode dekripsi MCC.

2.6 Android

Android merupakan sistem operasi berbasis Linux yang dirancang untuk perangkat lunak layar sentuh telepon pintar maupun komputer tablet[9]. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005[9]. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan - perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler[10].

Antarmuka dari Android berupa manipulasi langsung, yaitu dengan menggunakan sentuhan seperti menggeser, mengetuk, dan mencubit untuk memanipulasi objek di layar. Selain perangkat layar sentuh, Google juga telah mengembangkan Android TV untuk televisi, Android Auto untuk mobil, dan Android Wear untuk jam tangan, masing-masingnya memiliki antarmuka pengguna yang berbeda[10]. Varian Android juga digunakan pada komputer jinjing, konsol permainan, kamera digital, dan peralatan elektronik lainnya[11].

BAB III
METODE PENELITIAN

Pada bab ini dijelaskan mengenai perangkat yang digunakan dan langkah - langkah yang dilakukan dalam penyelesaian masalah Tugas Akhir ini.

3.1 Spesifikasi Perangkat

Untuk melakukan eksperimen pada penelitian tugas akhir digunakan Android Studio 3.5.3. Spesifikasi perangkat keras yang digunakan pada Android Studio adalah sebagai berikut.

- Prosesor: Intel(R) Core(TM) I7-7500U CPU @2.70 GHz, 2.90GHz.
- RAM: 8 GB.
- GPU: NVIDIA GeForce 940MX.

Bahasa pemrograman yang digunakan adalah Java. Implementasi dari RSA dan MCCRSA akan digunakan *library* BigInteger[12] dan Random [13]. BigInteger berfungsi untuk menyimpan Integer yang lebih dari 2147483648. Simulasi telepon pintar di laptop digunakan spesifikasi Android 9 dengan API 28. Aplikasi ini juga menggunakan Firebase sebagai backend. Eksperimen hasil implementasi dilakukan di perangkat keras dengan spesifikasi sebagai berikut.

- Prosesor: Samsung Exynos 7885 Octa-Core Processor (2.2GHz Dual + 1.6GHz Hexa)
- RAM: 6 GB.
- OS: Android 9
- GPU: Mali-G71

3.2 Tahapan Penelitian Tugas Akhir

Berikut adalah penjelasan dari tiap langkah yang dilakukan dalam penelitian Tugas Akhir ini:

1. **Studi literatur**
   Tahap pertama adalah melakukan studi referensi mengenai algoritma Caesar Cipher dan RSA. Studi literatur yang digunakan berupa buku Kriptografi untuk mengetahui cara kerja Caesar Cipher dan RSA.

2. **Implementasi algoritma RSA**
   Setelah melakukan studi literatur, akan dibuat implementasi dari algoritma RSA dengan menggunakan android studio[14]. Implementasi yang dimaksud merupakan pembuatan aplikasi android untuk enkripsi dekripsi menggunakan algoritma RSA.

3. **Implementasi algoritma MCC**
   Pada tahap ini, akan dibuat implementasi dari algoritma MCC dengan menggunakan android studio. Implementasi yang dimaksud merupakan pembuatan aplikasi android untuk enkripsi dekripsi menggunakan algoritma MCC.

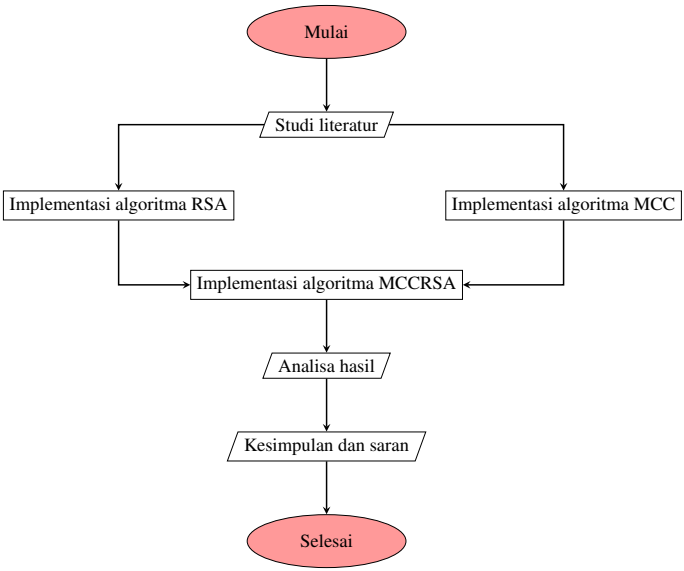
4. **Implementasi algoritma MCCRSA**
   Pada tahap ini, akan dibuat implementasi algoritma MCCRSA dengan menggunakan tool android studio. Implementasi yang dimaksud merupakan pembuatan aplikasi android untuk enkripsi dekripsi menggunakan algoritma MCCRSA.

5. **Analisis hasil implementasi**
   Setelah mendapatkan hasil enkripsi dan dekripsi MCCRSA, akan dibandingkan dengan RSA dan MCC. Waktu dari enkripsi dan dekripsi akan dibandingkan agar mendapatkan hasil performa dari MCCRSA.

6. **Kesimpulan dan saran**
   Untuk tahap terakhir, dipaparkan kesimpulan dari hasil si-

mulasi serta beberapa saran rencana penelitian untuk periode selanjutnya.

Berikut adalah diagram alir dari langkah - langkah yang sudah diuraikan di atas:



Gambar 3.1. Diagram alir penyelesaian masalah Tugas Akhir.

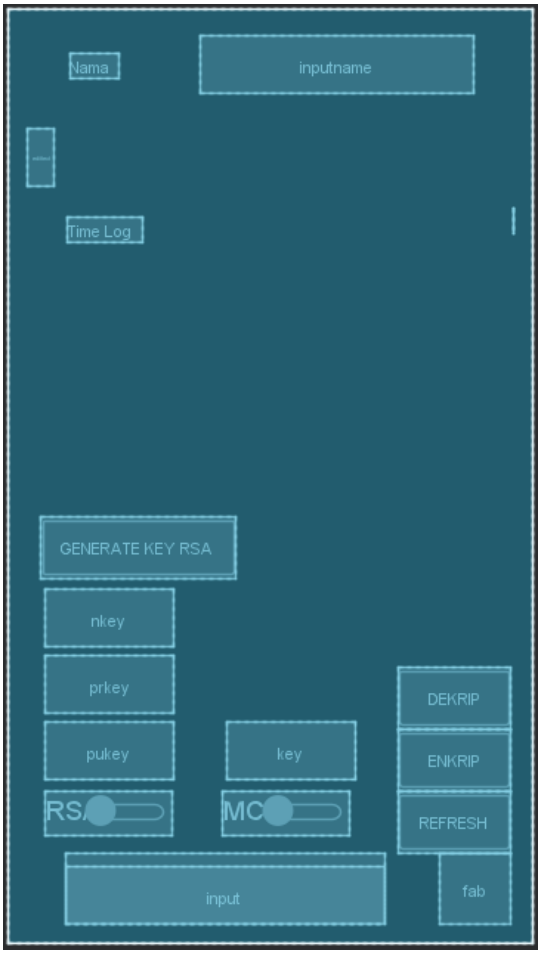
BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan bagaiman implementasi dari algoritma RSA, implementasi algoritma MCC, serta analisa hasil eksperimen.

4.1 Pengembangan Aplikasi

Aplikasi untuk mengirim pesan tentu dibutuhkan nama pengirim dan tempat pesan untuk mengisi pesan awal. Maka dari itu akan dibuat 2 EditText untuk tempat menampung nama dan pesan awal. Penerimaan pesan juga harus membutuhkan tempat untuk menerima pesan dan nama pengirimnya sehingga akan dibentuk 1 EditText untuk menampung pesan dan 1 tempat untuk menampung nama dari pengirim. Pengiriman pesan juga membutuhkan sebuah server sehingga pesan dapat dikirim ke pengguna aplikasi lainnya. Server yang akan digunakan adalah server dari Firebase. Data yang dikirim ke server merupakan pesan awal dan nama pengirim. Penerimaan pesan dari server akan berjalan secara otomatis.

Pengguna dapat memilih algoritma untuk enkripsi dan dekripsi sehingga akan dibentuk sebuah tombol untuk memilih masing - masing algortima. Algortima MCC membutuhkan 1 tempat untuk menapung kunci sehingga akan harus disediakan 1 tempat. Algoritma RSA diutuhkan 3 tempat kunci, yaitu kunci n, kunci e, dan kunci d sehingga akan dibuat 3 EditText. Kunci dari RSA harus dibangkitkan sehingga akan disediakan tombol untuk membangkitkan kunci. Eksekusi enkripsi dan dekripsi harus dimulai dengan sebuah tanda sehingga akan disediakan 1 tombol untuk masing - masing fungsi. Hasil enkripsi akan dilitakan di tempat pesan awal sehingga dapat dikirimkan dan hasil dari dekripsi akan diletakan di tempat penerimaan pesan. Pesan yang berada di tempat penerimaan pesan dapat dimuat ulang dengan menyediakan 1 tombol. Waktu untuk ekripsi dan dekripsi akan disediakan untuk mengetahui kecepatan algortima. Hasil dari desain aplikasi sebagai berikut.



Gambar 4.1. Desain aplikasi

Nama dari aplikasi akan diletakan diatas dari desain Gambar 4.1. Kemudian EditText dengan id inputname merupakan tempat untuk menyimpan nama dari pengirim. Selanjutnya terletak dibawah tempat nama merupakan tempat untuk menerima pesan. Tempat dengan id TimeLog merupakan tempat untuk menampilkan waktu enkripsi atau waktu dekripsi. Sebelah kanan dari TimeLog merupakan tempat untuk nama pengirim. EditText dengan id input merupakan tempat untuk pesan yang ingin dikirimkan. Tombol disebelah kanan dari input berguna untuk mengirim pesan. Tombol REFRESH untuk memuat ulang pesan yang diterima dari server. Tombol DEKRIP dan tombol ENKRIP mempunyai fungsi enkripsi dan dekripsi. Tombol *swicth* dengan id RSA dan MCC berguna untuk memilih algoritma yang digunakan. Tiga uah EditText diatas tombol RSA berguna untuk melakukan masukan kunci. Tombol GENERATE KEY RSA merupakan tomol untuk memangkitkan kunci RSA. EditText yang terletak diatas tombol MCC erguna untuk masukan kunci MCC.

4.2 Pengkodean Pesan

Implementasi dari kedua algoritma dibutuhkan pemetaan pesan dalam bentuk huruf ke numerik. String adalah class di Java mempunyai metode getBytes() berguna untuk memetakan setiap karakter pada pesan ke numerik. Metode ini mengeluarkan array byte sebagai output. Tabel pemetaan merupakan tabel ASCII yang terlampirkan. Sebagai contoh berdasarkan Tabel ASCII, a adalah 01100001 dikarenakan 1 byte adalah 8 bit .Jika diubah ke numerik maka a = $2^0 + 2^5 + 2^6$ = 97. Potongan program untuk pemetaan sebagai berikut. Potongan program ini berfungsi untuk mengubah semua

```
    byte[] bytes = text.getBytes();
```

Gambar 4.2. Potongan kode untuk pemetaan karakter ke byte array

karakter dari String text menjadi bit. Semua karakter yang sudah dikonversikan kemudian disimpan ke array byte dengan nama bytes.

4.3 Implementasi Algoritma MCC

Entitas untuk implementasi algoritma ini ada 2, yaitu pengirim dan penerima. Desain dari aplikasi algoritma MCC yang diusulkan untuk pengirim adalah sebagai berikut.

1. **Input Pesan dan Kunci**
   Data input yang digunakan adalah pesan yang sudah di kodekan dalam bentuk array. Kunci yang digunakan diubah ke bentuk integer. Potongan program untuk pengambilan pesan dan kunci pada Gambar 4.3. Potongan program ini dimu-

```
EditText input = findViewById(R.id.input);
EditText key = findViewById(R.id.key);
temp = new BigInteger(Libraries.encryptmcc(
    input.getText().toString(),
        Integer.parseInt(key.getText().toString)
    ));
input.setText(temp.toString())
```

Gambar 4.3. Potongan kode untuk pengambilan pesan

lai dengan mencari EditText yang mempunyai id input. Jika EditText ditemukan, maka EditText tersebut diberi nama input. Baris selanjutnya mempunyai fungsi yang sama dengan sebelumnya. Program mencari EditText dengan id key dan memberikan nama key. Kemudian program memanggil fungsi encryptmcc yang berada di kelas Libraries dengan 2 parameter. Parameter pertama berfungsi untuk mengambil teks yang berada di input. Parameter kedua berfungsi untuk mengambil teks yang berada di key dan diubah menjadi Integer.

2. **Enkripsi MCC**
   Kemudian, setiap elemen array ditambahkan dengan posisinya dikalikan dengan kunci lalu dimodulus 128. Potongan program untuk enkripsi MCC pada Gambar 4.4 Potongan

```
public static byte[] encryptmcc(String text, int
    key){
    byte[] bytes = text.getBytes();
    int temp;
    for(int i=0; i<bytes.length();i++){
        temp = (bytes[i]+key*(i+1))%128;
        bytes[i] = (byte)temp;
    }
    return bytes;
}
```

Gambar 4.4. Potongan kode untuk enkripsi pesan

program ini merupakan fungsi encryptmcc di class Libraries dengan parameter text dan key. Program diawali dengan mengubah String text menjadi array Byte yang diberikan nama bytes. Baris berikutnya program inisialisasi Integer temp. Pengulangan for dilakukan sebanyak elemen bytes. Setiap pengulangan mengubah 1 elemen pada bytes menggunakan algoritma enkripsi. Setelah itu, fungsi diakhiri dengan memberikan bytes sebagai output.

3. **Pengiriman Pesan Cipher**
   Pada tahap ini, array akan diubah menjadi BigInteger yang kemudian dikirimkan ke server Firebase. Potongan program untuk pengiriman pesan cipher pada Gambar 4.5. Pada Gambar 4.3, dapat dilihat output dari fungsi encrptmcc diubah menjadi BigInteger dan disimpan dalam temp. Temp ini kemudian diubah menjadi String dan diletakan di EditText input. Potongan program pada Gambar 4.5 mencari EditText dengan

```
EditTex input = findViewById(R.id.input);
message.setValue(input.getText().toString);
```

Gambar 4.5. Potongan kode untuk pengiriman pesan

id input dan memberikan nama input. Setelah itu, program mengambil teks diinput dan memasukannya ke message, dimana message berasal dari database Firebase.

Desain dari aplikasi algoritma MCC yang diusulkan untuk penerima adalah sebagai berikut.

1. **Penerimaan Pesan Cipher**
   Setelah diperoleh pesan dari server, BigInteger kemudian dibentuk menjadi array byte. Gambar 4.6 merupakan potongan program untuk penerimaan pesan cipher. Program ini dapat

```
message.addValueEventListener(new
    ValueEventListener){
    public void onDataChange(DataSnapshot data
        Snapshot){
        String value = dataSnapshot.getValue(
            String.class);
        EditText output = findViewId(R.id.edittext
            );
        output.setText(value);
    }

}
```

Gambar 4.6. Potongan kode untuk penerimaan pesan

mendeteksi perubahan pada database. Jika data berubah, maka program akan mengambil pesan dan meletakan pesan tersebut pada EditText dengan id edittext.

2. **Input**
   Input kunci untuk dekripsi dalam bentuk BigInteger. BigInteger kemudian diubah menjadi Byte array. Potongan program untuk pengambilan kunci dan pesan cipher pada Gambar 4.7. Potongan program mengubah pesan cipher pada output men-

```
BigInteger BIctext = new BigInteger(output.getText
    ().toString());
temp = new BigInteger(Libraries.decryptmcc(
    BIctext.getByteArray(),
    Integer.parseInt(key.getText().toString)
));
```

Gambar 4.7. Potongan program untuk membaca pesan dan kunci

   jadi BigInteger dan disimpan di BIctext. Kemudian program memanggil fungsi decryptmcc pada class Libraries dengan 2 parameter. Parameter pertama adalah pesan cipher yang diubah ke dalam array Byte dan parameter kedua mengambil kunci dan diubah ke bentuk integer.

3. **Dekripsi MCC**
   Kemudian, setiap elemen dari array byte dikurangi dengan posisinya dikalikan dengan kunci kemudian dimoduluskan dengan 128. Gambar 4.8 merupakan potongan program dekripsi MCC. Potongan program ini merupakan fungsi decryptmcc di class Libraries dengan parameter text dan key. Program diawali dengan mengubah String text menjadi array Byte yang diberikan nama bytes. Baris berikutnya program inisialisasi Integer temp. Pengulangan for dilakukan sebanyak elemen bytes. Setiap pengulangan mengubah 1 elemen pada bytes menggunakan algoritma dekripsi. Setelah itu, fungsi diakhiri dengan memberikan bytes sebagai output.

4. **Pemetaan Balik Pesan**

```
public static byte[] decryptmcc(String text, int
    key){
    byte[] bytes = text.getBytes();
    int temp;
    for(int i=0; i<bytes.length();i++){
        temp = (bytes[i]-key*(i+1))%128;
        bytes[i] = (byte)temp;
    }
    return bytes;
}
```

Gambar 4.8. Potongan program untuk dekripsi pesan

Setiap elemen di array kemudian disubtitusikan menjadi pesan karakter. Potongan program pada Gambar 4.9 mengubah

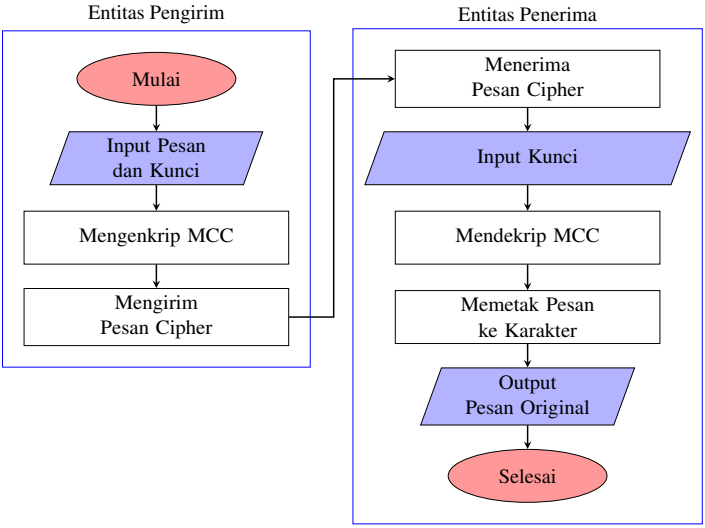
```
output.setText(new String(temp.toByteArray()))
```

Gambar 4.9. Potongan program untuk pemetaan balik

pesan yang sudah melalui dekripsi di temp ke Byte array yang kemudian di ubah lagi menjadi String.

5. **Hasil Enkripsi Dekripsi**
   Setelah tahap pemetaan balik, diperoleh pesan awal yang ingin dikirim. Hasil dekripsi ditampilkan di EditText dengan nama output. Potongan program untuk menampilkan hasil enkripsi dekripsi pada Gambar 4.9

Berikut adalah diagram alir dari desain algoritma MCC yang diusulkan.



Gambar 4.10. Diagram alir desain aplikasi algoritma MCC.

4.4 Implementasi Algoritma RSA

Entitas untuk implementasi algoritma ini ada 2, yaitu pengirim dan penerima. Desain dari aplikasi algoritma RSA yang diusulkan untuk entitas pengirim adalah sebagai berikut.

1. **Input Data dan Kunci**
   Data input yang digunakan adalah pesan yang sudah di kodekan dalam bentuk array. Kunci yang diinput berbentuk BigInteger yang dimiliki oleh penerima. Potongan program untuk pengambilan pesan dapat dilihat di Gambar 4.3 dan untuk pengambilan kunci RSA dapat dilihat di Gambar 4.11. Potongan program diawali dengan mencari EditText dengan id pukey dan nkey yang akan diberi nama pub dan n. Setelah itu program memanggil fungsi encryptrsa di class Libraries dengan 3 parameter. Parameter pertama merupakan pesan yang ingin di enkripsi diambil dari input. Parameter kedua

```
EditText pub = findViewById(R.id.pukey);
EditText n = findViewById(R.id.nkey);
BigInteger cipher = Libraries.encryptrsa(
    input.getText.toString(),
    new BigInteger(pub.getText().toString()),
    new BigInteger(n.getText().toString())
);
```

Gambar 4.11. Potongan program untuk pengambilan pesan dan kunci

dan ketiga merupakan kunci enkripsi yang diambil dari pub dan n.

2. **Enkripsi RSA**
   Kemudian, array pesan diubah menjadi BigInteger lalu dienkripsi dengan kunci publik. BigInteger berguna untuk menampung Integer yang leih dari 2147483648. Fungsi yang dipanggil untuk melakukan enkripsi adalah encryptrsa di class Libraries, pada Gambar 4.11 dan isi dari fungsi encryptrsa pada Gambar 4.12 Potongan program merupakan fungsi encr-

```
public static BigInteger encryptrsa(byte[] message
    , BigInteger e, BigInteger N){
    return (new BigInteger(message)).modPow(e,N);
}
```

Gambar 4.12. Potongan program untuk enkripsi RSA

yptrsa yang berada di class Libraries. Fungsi ini mengubah array byte pesan ke BigInteger pesan cipher menggunakan algoritma enkripsi RSA.

3. **Pengiriman Pesan Cipher**
   Pada tahap ini, BigInteger yang kemudian dikirimkan ke ser-

ver Firebase. Potongan program ini dapat dilihat di Gambar 4.5.

Desain dari aplikasi algoritma RSA yang diusulkan untuk entitas penerima adalah sebagai berikut.

1. **Penerimaan Pesan Cipher**
   Aplikasi menerima pesan Cipher dari server Firebase berupa BigInteger. Potongan program dapat dilihat di Gambar 4.6.

2. **Input Kunci Privat**
   Kunci Privat dalam bentuk BigInteger. Potongan program dapat dilihat di Gambar 4.13. Potongan program ini diawali

```
EditText pri = findViewById(R.id.prkey);
EditText n = findViewById(R.id.nkey);
BigInteger cipher = Libraries.encryptrsa(
    input.getText.toString(),
    new BigInteger(pri.getText().toString()),
    new BigInteger(n.getText().toString())
);
```

Gambar 4.13. Potongan program untuk pengambilan kunci privat.

   dengan mencari dan menamai EditText id prkey dan nkey dengan nama pri dan n. Setelah itu program memanggil fungsi decryptrsa di class Libraries dengan 3 parameter. Parameter pertama merupakan pesan cipher yang ingin di dekripsi diambil dari ouput. Parameter kedua dan ketiga merupakan kunci dekripsi yang diambil dari pri dan n.

3. **Dekripsi RSA**
   Kemudian, BigInteger didekripsi dengan menggunakan kunci privat.Fungsi yang dipanggil untuk melakukan dekripsi adalah decryptrsa di class Libraries, pada Gambar 4.11 dan isi dari fungsi decryptrsa pada Gambar 4.14. Potongan program

```
public static BigInteger encryptrsa(BigInteger
    message, BigInteger d, BigInteger N){
    return (new BigInteger(message)).modPow(d,N);
            }
```

Gambar 4.14. Potongan program untuk dkripsi RSA

merupakan fungsi decryptrsa yang berada di class Libraries. Fungsi ini mengubah BigInteger pesan cipher ke BigInteger hasil dekripsi menggunakan algoritma dekripsi RSA.

4. **Pemetaan Balik Pesan**
   Setiap elemen di array kemudian disubtitusikan menjadi pesan karakter. Potongan program untuk pemetaan balik dapat dilihat di Gambar 4.9.

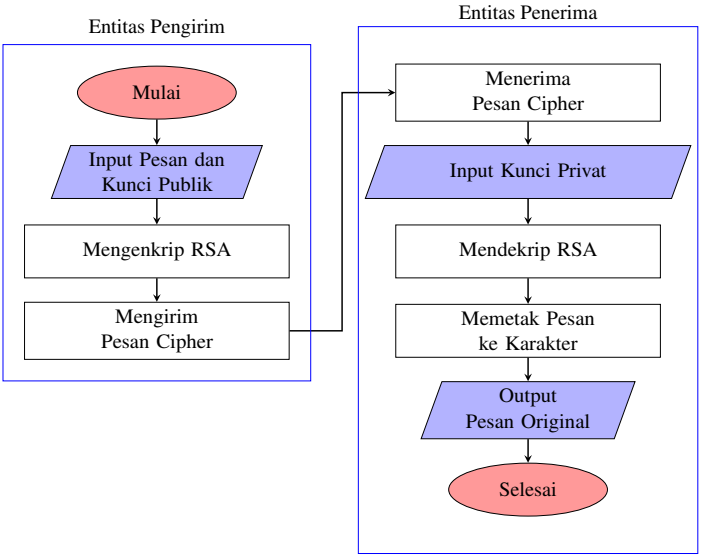
5. **Hasil Enkripsi Dekripsi**
   Setelah tahap pemetaan balik, diperoleh pesan awal yang ingin dikirim. Hasil dekripsi ditampilkan di EditText dengan nama output. Potongan program untuk menampilkan hasil enkripsi dekripsi pada Gambar 4.9

Diagram alir untuk desain algoritma RSA pada Gambar 4.15

4.5 Implementasi Algoritma MCCRSA

Entitas untuk implementasi algoritma ini ada 2, yaitu pengirim dan penerima. Selain entitasnya, MMCRSA merupakan gabungan dari MCC dan RSA. Desain dari aplikasi algoritma MCCRSA yang diusulkan untuk entitas pengirim adalah sebagai berikut.

1. **Input Data dan Kunci**
   Data input yang digunakan adalah pesan yang sudah di kodekan dalam bentuk array. Algortima ini memerlukan 2 kunci, yaitu kunci publik RSA dan kunci MCC. Kunci publik RSA



Gambar 4.15. Diagram alir desain aplikasi algoritma RSA yang diusulkan.

berbentuk BigInteger yang dimiliki oleh penerima. Kunci MCC berbentuk Integer. Potongan program untuk membangkitkan kunci RSA terdapat pada Gambar 4.11. Potongan program untuk mengambil kunci MCC terdapat pada Gambar 4.3.

2. **Enkripsi MCC**
   Kemudian, setiap elemen array ditambahkan dengan posisinya dikalikan dengan kunci lalu dimodulus 128. Potongan program untuk enkripsi MCC pada Gambar 4.4.

3. **Enkripsi RSA**
   Kemudian, array pesan diubah menjadi BigInteger lalu dienkripsi dengan kunci publik. Fungsi yang dipanggil untuk melakukan enkripsi adalah encryptrsa di class Libraries, pada

Gambar 4.11 dan isi dari fungsi encryptrsa pada Gambar 4.12.

4. **Pengiriman Kunci publik**
   Pada tahap ini, kunci publik berupa BigInteger yang dikirimkan ke server Firebase sebagai pesan biasa. Potongan program untuk pengiriman kunci dalam bentuk pesan biasa dapat dilihat di Gambar 4.5.

Desain dari aplikasi algoritma MCCRSA yang diusulkan adalah sebagai berikut.

1. **Penerimaan Pesan Cipher**
   Aplikasi menerima pesan Cipher dari server Firebase berupa BigInteger. Potongan program dapat dilihat di Gambar 4.6.

2. **Input Kunci Privat**
   MCCRSA merupakan gabungan MCC dan RSA, maka kunci yang diperlukan ada 2, yaitu kunci privat RSA dan kunci MCC. Kunci Privat RSA dalam bentuk BigInteger. Potongan program untuk kunci privat RSA dapat dilihat di Gambar 4.13. Potongan program untuk kunci MCC 4.7.

3. **Dekripsi RSA**
   Kemudian, BigInteger didekripsi dengan menggunakan kunci privat.Fungsi yang dipanggil untuk melakukan dekripsi adalah decryptrsa di class Libraries, pada Gambar 4.11 dan isi dari fungsi decryptrsa pada Gambar 4.14.

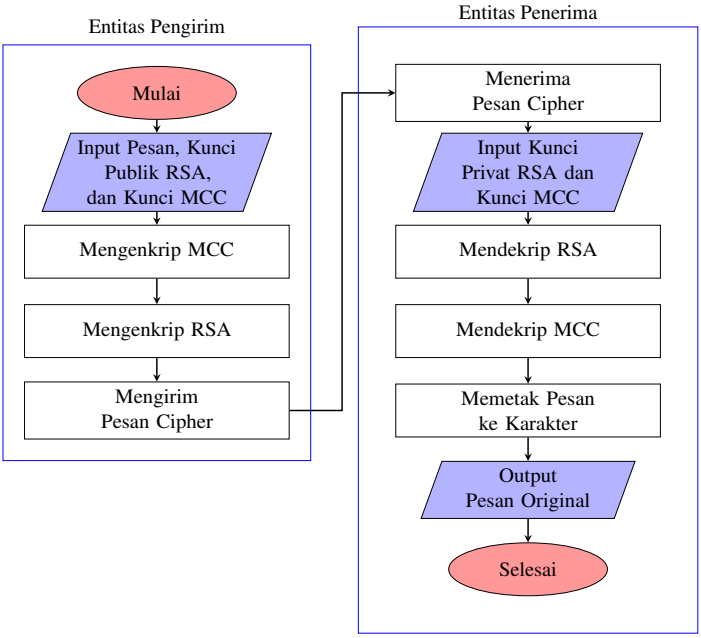
4. **Dekripsi MCC**
   Kemudian, setiap elemen dari array byte dikurangi dengan posisinya dikalikan dengan kunci kemudian dimoduluskan dengan 128. Gambar 4.8 merupakan potongan program dekripsi MCC.

5. **Pemetaan Balik Pesan**
   Setiap elemen di array kemudian disubtitusikan menjadi pe-

san karakter. Potongan program untuk pemetaan balik dapat dilihat di Gambar 4.9.

6. **Hasil Enkripsi Dekripsi**
   Setelah tahap pemetaan balik, diperoleh pesan awal yang ingin dikirim. Hasil dekripsi ditampilkan di EditText dengan nama output. Potongan program untuk menampilkan hasil enkripsi dekripsi pada Gambar 4.9.

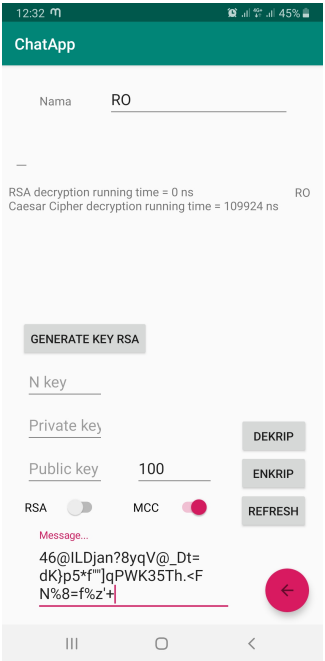
Diagram alir untuk desain algoritma pada Gambar 4.16.



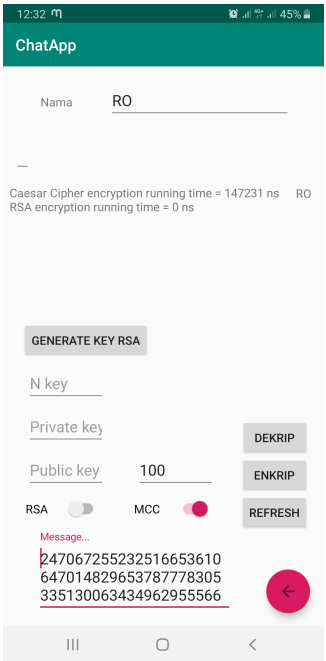
Gambar 4.16. Diagram alir desain aplikasi algoritma MCCRSA.

4.6 Hasil Implementasi

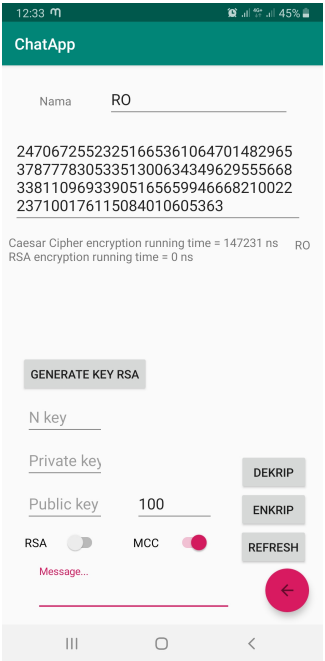
Implementasi algoritma dimulai dengan memilih algoritma enkripsi dan dekripsi. Pembangkitan untuk kunci RSA dapat dengan tombol GENERATE KEY RSA. Setelah itu, pengguna melakukan input kunci dan pesan seperti pada Gambar 4.17. Selanjutnya untuk enkripsi menggunakan tombol berlabel ENKRIP menjadi Gambar 4.18



Gambar 4.17. Pesan dan Kunci untuk MCC.

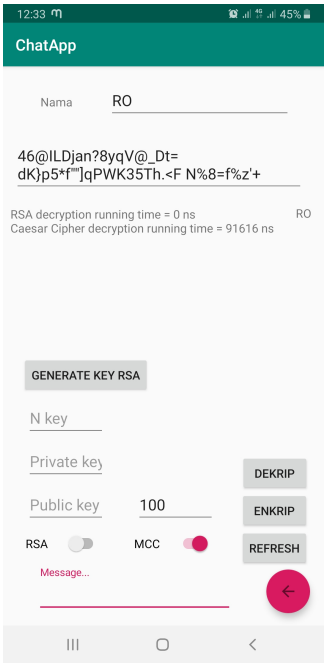


Gambar 4.18. Tombol ENKRIP ditekan



Gambar 4.19. Pengiriman pesan cipher.

Pengiriman dilakukan dengan menekan tombol panah. Pesan Cipher ini akan terkirim ke semua telepon pintar yang memiliki aplikasi ini. Jika pesan telah diterima seperti pada Gambar 4.19, maka dapat didekripsi dengan tombol DEKRIP menjadi Gambar 4.20.



Gambar 4.20. Tombol DEKRIP ditekan

Hasil dari tiga implementasi bisa diukur dengan empat parameter sebagai berikut.

- Waktu enkripsi
  Jumlah waktu yang diperlukan untuk mengenkripsi pesan.
- Waktu dekripsi
  Jumlah waktu yang diperlukan untuk mendekripsi pesan cipher.
- Keamanan
  Efisiensi algoritma untuk meretas pesan cipher.
- Volume bit pesan cipher
  Jumlah bit yang digunakan untuk menyimpan pesan cipher.

4.6.1 Uji MCCRSA

MCC merupakan algoritma modifikasi dari Caesar Cipher. Algoritma ini mempunyai rumus enkripsi $C = (P + R * T) \mod 128$ dan rumus dekripsi $P = (C - R * T) \mod 128$. Perhatikan bahwa jika C disubstitusikan dengan rumus enkripsi didapat

$$P = ((P + R * T) - R * T) \mod 128$$

$$P = (P + R * T - R * T) \mod 128$$

$$P = P \mod 128$$

Karena setiap karakter P lebih kecil dari 128, maka berdasarkan rumus enkripsi dan dekripsi $P = P$, dimana P adalah setiap karakter pada pesan awal. Jadi algoritma MCC terbukti akan mempunyai hasil dekripsi yang sama dengan pesan awal.

MCCRSA merupakan algoritma yang menggabungkan algoritma MCC dan RSA, dimana RSA sudah dibuktikan kebenarannya[5]. Algoritma ini dimulai dengan mengenkripsi pesan awal ke pesan cipher 1 dengan algoritma MCC. Kemudian pesan cipher 1 dienkripsi untuk kedua kalinya menjadi pesan cipher 2. Hasil dari enkripsi kemudian dikirimkan ke penerima pesan sehingga tidak ada perubahan pesan. Pesan cipher 2 kemudian didekripsi dengan algoritma RSA yang menghasilkan pesan cipher 1. Hal ini dikarenakan hasil dekripsi RSA sama dengan pesan sebelum enkripsi RSA. Kemudian dekripsi dilanjutkan dengan algoritma MCC. Karena pesan cipher 1 merupakan hasil enkripsi MCC, makahasil dekripsi pesan cipher 1 merupakan pesan awal. Jadi terbukti bahwa hasil dekripsi algoritma MCCRSA saam dengan pesan awal.

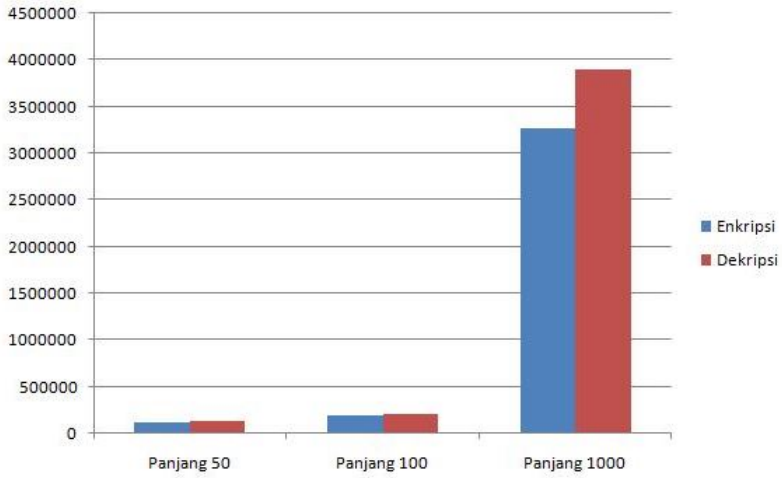
4.6.2 Waktu Enkripsi dan Waktu Dekripsi

Penelitian dilakukan dengan mengirim pesan enkripsi dan dekripsi yang kemudian diukur besar waktu enkripsi dan waktu dekripsi. Pesan yang dikirim adalah pesan dengan panjang 50, 100, dan,

1000 yang dibangkitkan secara acak. Kemudian semua pesan akan dienkripsi dan didekripsi untuk menghasilkan waktu enkripsi dan dekripsi. Setiap dari kategori pesan akan dilakukan uji coba selama 50 kali dan waktu enkripsi dan dekripsi akan dirata - rata.

Pesan yang sudah dibangkitkan secara acak dikodekan menjadi array byte. Kemudian array ini di enkripsi dengan algoritma MCC dengan kunci '100'. Hasil dari enkripsi kemudian didekripsi kembali menjadi pesan awal oleh algoritma dekripsi MCC. Waktu untuk enkripsi dan dekripsi kemudian dicatat dan menghasilkan rata - rata waktu enkripsi 120818 nanodetik dan waktu dekripsi 141598 nanodetik untuk panjang pesan 50. Panjang pesan 100 mempunyai rata - rata waktu enkripsi 195550 nanodetik dan waktu dekripsi 209152 nanodetik. Panjang pesan 1000 mempunyai rata - rata waktu enkripsi 3268290 nanodetik dan waktu dekripsi 3903828 nanodetik. Waktu enkripsi dan dekripsi terlihat semakin besar jika panjang pesan semakin panjang. Hal ini disebabkan karena algoritma melakukan kalkulasi setiap karakter yang terdapat pada pesan.

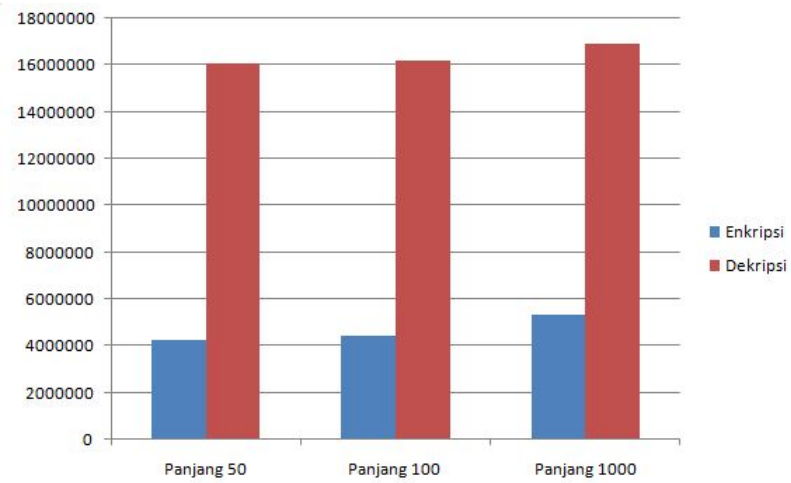
Setelah uji coba algoritma MCC, dilakukan uji coba algoritma RSA. Pesan yang akan dikirim merupakan pesan yang diangkitkan secara acak. Kemudian pesan ini diubah menjadi BigInteger. BigInteger ini dienkripsi dengan algoritma RSA dengan kunci yang dilampirkan. Hasil dari enkripsi didekripsi kemali menjadi pesan awal dengan algoritma RSA. Waktu enkripsi dan dekripsi dicatat dan menghasilkan rata - rata waktu enkripsi 4283564 nanodetik dan waktu dekripsi 16104904 nanodetik untuk panjang pesan 50. Panjang pesan 100 mempunyai rata - rata waktu 4463538 enkripsi nanodetik dan waktu 16199480 dekripsi nanodetik. Panjang pesan 1000 mempunyai rata - rata waktu enkripsi 5314318 nanodetik dan waktu dekripsi 16898676 nanodetik. Waktu enkripsi terlihat semakin besar jika panjang pesan semakin panjang. Hal ini disebakan karena semakin panjang pesan, semakin besar pula BigInteger untuk menyimpan pesan sehingga kalkulasi yang dilakukan oleh komputer semakin besar. Berbeda dengan waktu enkripsi, waktu dekripsi



Gambar 4.21. Graph waktu rata - rata enkripsi dan dekripsi MCC

tidak jauh berbeda. Hal ini dikarenakan hasil enkripsi dari pesan bergantung pada kunci publik *n* dimana pesan cipher mempunyai panjang maksimum kunci publik *n*. Maka dari itu, waktu dekripsi tidak bergantung pada pesan awal melainkan bergantung pada hasil enkripsi.

Uji coba MCCRSA dimulai dengan pembangkitan pesan secara acak dikodekan menjadi array byte. Kemudian array ini di enkripsi dengan algoritma MCC dengan kunci '100'. Hasil dari enkripsi kemudian diubah menjadi BigInteger dan dienkripsi dengan algoritma RSA. Hasil dari enkripsi RSA didekripsi menggunakan algortima RSA. Hasil dari dekripsi kemudian diubah menjadi array dan didekripsi lagi dengan algoritma MCC. Waktu untuk enkripsi dan dekripsi kemudian dicatat dan menghasilkan rata - rata waktu enkripsi 4777312 nanodetik dan waktu dekripsi 17004736 nanodetik untuk panjang pesan 50. Panjang pesan 100 mempunyai rata - rata waktu enkripsi 5001014 nanodetik dan waktu dekripsi 18460090 na-

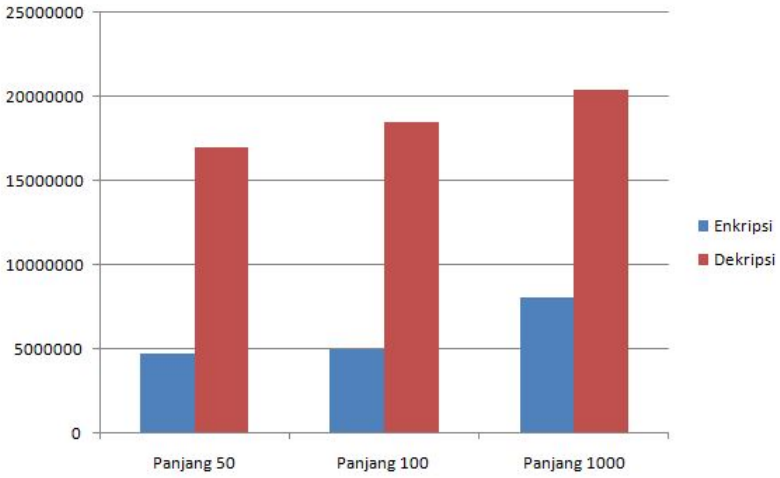


Gambar 4.22. Graph waktu rata - rata enkripsi dan dekripsi RSA

nodetik. Panjang pesan 1000 mempunyai rata - rata waktu enkripsi 8117970 nanodetik dan waktu dekripsi 20390526 nanodetik. Waktu enkripsi dan dekripsi terlihat semakin besar jika panjang pesan semakin panjang. Hal ini disebabkan karena algoritma MCC dalam MCCRSA melakukan kalkulasi setiap karakter yang terdapat pada pesan dan enkripsi algoritma RSA melakukan kalkulasi terhadap pesan awal. Perubahan pada waktu dekripsi dari MCCRSA tidak sebesar waktu enkripsi karena dekripsi dari RSA tidak dipengaruhi oleh panjang pesan. Algoritma ini merupakan gabungan dari kedua algortima sehingga waktu enkripsi dan dekripsi yang dibutuhkan merupakan jumlahan dari waktu enkripsi dan dekripsi kedua algoritma.

4.6.3 Keamanan Data

Keamanan data dapat dihitung dari masing-masing algoritma untuk meretas pesan ciphernya. Perhitungan kompleksitas dari al-



Gambar 4.23. Graph waktu rata - rata enkripsi dan dekripsi MCCR-SA

gortima peretas pesan cipher sebagai berikut.

- MCC
  MCC dapat dipecahkan dengan *brute force*. Teknik ini adalah mencoba semua kemungkinan kunci. Hal ini dikarenakan banyaknya variasi kunci hanya ada 128. Berdasarkan potongan program pada Gambar 4.8, diketahui bahwa diperlukan loop sebanyak jumlah elemen karakter dari pesan cipher. Maka kompleksitas program untuk meretas pesan cipher MCC adalah $O(j)$, dimana j adalah panjangnya pesan yang akan dikirim.

- RSA
  RSA berdasarkan pada sulitnya pemfaktorkan bilangan bulat besar menjadi faktor - faktor prima[5]. Jika ingin meretas pesan cipher, diperlukan kunci privat. Cara memperoleh kunci

privat adalah dengan memfaktor-kan kunci publik n menjadi p dan q. Setelah itu, menghitung nilai $\phi(n)$. kunci privat d dapat dihitung dengan $e.d \equiv 1(\mod \phi(n))$ karena kunci publik e diketahui. Algoritma terbaik untuk pemfaktoran saat ini memiliki kompleksitas $O(\exp(\sqrt[3]{\frac{64}{9}b(\log(b)^2)}))$ untuk bilangan bulat besar yang panjangnya b-bit[5]. Sehingga dapat dikatakan bahwa untuk meretas algoritma ini, dibutuhkan algoritma yang mempunyai kompleksitas $O(\exp(\sqrt[3]{\frac{64}{9}b(\log(b)^2)}))$ dimana b merupakan besarnya volume bit yang menyimpan kunci publik n. Menurut Rivest dan kawan - kawan, usaha untuk mencari faktor prima dari bilangan 200 angka desimal membutuhkan waktu komputasi selama 4 milyar tahun (Meyer, 1982) (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algortima yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik)[5].

- MCCRSA
  Sebuah pesan yang sudah dienkripsi kemudian dienkripsi lagi mempunyai kemungkinan tingkat keamanan yang lebih besar dari pesan yang hanya dienkripsi sekali. Algoritma MCCRSA merupakan gabungan dari algoritma RSA dan MCC. Algoritma peretas untuk RSA merupakan algortima untuk memfaktorkan kunci publik n menjadi 2 bilangan prima. Algoritma peretas MCC menggunakan algoritma *Brute Force* sehingga dapat disimpulakan bahwa algoritma peretas MCC tidak sama dengan algoritma peretas RSA. Maka dari itu algoritma untuk meretas MCCRSA harus menggunakan kedua algoritma peretas. Dekripsi dimulai dengan dekripsi RSA dengan $O(\exp(\sqrt[3]{\frac{64}{9}b(\log(b)^2)}))$ dengan b adalah panjang bit bilangan bulat besar dari kunci n dan dilanjutkan dengan dekripsi MCC dengan $O(j)$ dengan j adalah panjang pesan. Maka

algoritma ini mempunyai tingkat kompleksitas

$$O(\exp(\sqrt[3]{\frac{64}{9}b(\log(b)^2)})) + O(j) =$$

$$O(\max(\exp(\sqrt[3]{\frac{64}{9}b(\log(b)^2)}), j))$$

dimana j adalah panjang pesan dan b adalah besar volume bit dari kunci publik n. Dapat dilihat bahwa kompleksitas algoritma peretas dari MCC jauh lebih sederhana jika dibandingkan dengan RSA. Perhatikan bahwa nilai j dan b tidak bergantung, sehingga ada kemungkinan bahwa nilai dari j jauh lebih besar dari nilai b. Nilai dari j merupakan panjang dari pesan, maka semakin panjang teks yang dikirimkan semakin baik algoritma ini digunakan.

4.6.4 Volume Bit

Perhitungan Volume bit dapat dihitung berdasarkan dari algoritma masing-masing. Perhitungan untuk setiap algoritma sebagai berikut.

- MCC
  MCC menambahkan setiap elemen pada array pesan dengan sejumlah angka yang kemudian dimoduluskan dengan 128 dan disimpan di sebuah array byte. Tidak ada penambahan maupun pengurangan jumlah elemen array dari pesan. Selain itu setiap array dari byte membutuhkan volume 8 bit. Maka dapat disimpulkan bahwa volume bit yang dipakai untuk menyimpan pesan cipher adalah sejumlah elemen array itu sendiri dikalikan dengan 8. Jika elemen array dari sebuah pesan adalah j, maka volume bit yang terpakai untuk pesan cipher berbanding lurus dengan j.

- RSA
  Berbeda dengan MCC, algoritma RSA mengubah array byte pesan menjadi sebuah BigInteger. BigInteger kemudian dipangkatkan dan dimoduluskan dengan kunci n. Penambahan volume bit dari BigInteger berbanding lurus dengan besarnya BigInteger itu sendiri, sehingga semakin besar BigInteger semakin besar juga volume bit. Berdasarkan algoritma RSA, besarnya BigInteger pesan cipher tidak lebih besar dari kunci publik n. Jadi volume bit maksimum dari pesan cipher adalah volume bit kunci publik n.

- MCCRSA
  Algoritma enkripsi MCCRSA diawali dengan enkripsi MCC dan diakhiri oleh enkripsi RSA. Algoritma enkripsi MCC tidak mengubah jumlah dari volume bit pesan. Jadi volume it pesan awal sama dengan volume bit pesan setalah enkripsi MCC. Setelah enkripsi MCC dilakukan enkripsi RSA. Enkripsi RSA dimulai dengan mengubah pesan ke BigInteger. BigInteger mempunyai volume bit yang berbanding lurus dengan besarnya BigInteger itu sendiri. Berdasarkan algoritma RSA, besarnya BigInteger pesan cipher tidak lebih besar dari kunci publik n. Jadi volume bit maksimum dari pesan cipher adalah volume bit kunci publik n.

BAB V
KESIMPULAN DAN SARAN

Pesan dibangkitkan secara acak dengan panjang yang bereda dan dilakukan percobaan. Algoritma MCCRSA dapat melakukan enkripsi dan dekripsi dari semua pesan yang sudah dibangkitkan. Hasil dari enkripsi MCCRSA bereda dengan pesan awal dan hasil dari dekripsi sama dengan pesan awal. Jadi dapat disimpulkan ahwa algoritma MCCRSA dapat diimplementasikan untuk enkripsi dan dekripsi pesan pada Android.

Waktu enkripsi dan dekripsi terkecil adalah algoritma MCC. Sebaliknya, waktu enkripsi dekripsi teresar merupakan MCCRSA. Hal ini dikarenakan enkripsi dan dekripsi MCCRSA merupakan kombinasi dari kedua algoritma. Namun penambahan waktu ini tidak signifikan jika dilihat dari kompleksitas program yang linear sehingga MCCRSA dapat digunakan di. aplikasi *real time*.

Keamanan dari algoritma MCCRSA bisa dari kompleksitas dari algoritma peretasnya. Kompleksitas dari algoritma peretas pesan cipher MCCRSA merupakan gabungan dari algoritma peretas RSA dan MCC. Kompleksitasnya adalah

$$O(\max(\exp(\sqrt[3]{\frac{64}{9}b(\log(b)^2)}), j))$$

, dimana b adalah besar volume bit kunci pulik n dan j adalah panjang pesan. Kompleksitas ini bergantung pada pesan dan kuncinya. Jadi algoritma ini semakin baik jika pesan yang dikirimkan semakin panjang dan kunci publik n semakin besar.

Besarnya volume bit dari algoritma MCCRSA dan RSA bergantung pada besarnya kunci privat. Besar maksimum dari volume bit adalah b dimana b adalah panjang bit kunci publik n. Besar volume bit dari pesan cipher tidak bergantung pada besar volume bit dari pesan awal. Bereda dengan kedua algoritma sebelumnya, besarnya volume bit algoritma MCC bergantung pada panjangnya pesan. Volume bit yang diperlukan untuk menyimpan pesan cipher

MCC bergantung pada j dimana j merupakan panjangnya pesan. Dapat disimpulkan bahwa algoritma MCCRSA dapat digunakan karena besarnya volume bit sama dengan RSA.

Aplikasi dari pengirim dan penerima pesan membutuhkan 2 tempat, yaitu tempat untuk mengirim pesan dan menerima pesan. Pesan yang sudah dikirim harus diletakan di sebuah server sebelum dikirimkan ke aplikasi lain. Aplikasi ini juga membutuhkan 1 tempat untuk setiap kunci dari algoritma yang digunakan. Setiap dari fungsi ini membutuhkan sebuah tombol sebagai kontrol . Maka dari itu, desain pada Gambar 4.1 merupakan desain untuk algoritma MCCRSA.

Algoritma MCCRSA mempunyai keamanan data yang lebih dari algortima MCC dan RSA. Waktu enkripsi dan dekripsi dari MCCRSA tidak jauh berbeda dengan RSA dan volume bit pesan cipher setara dengan RSA. Oleh karena itu, algoritma MCCRSA bisa digunakan dan layak dalam aplikasi *real time*.

Saran untuk penelitian selanjutnya sebagai berikut

1. Menggabungkan 3 algoritma enkripsi dan dekripsi untuk menambah keamanan dari pesan.

2. Melakukan enkripsi pada kunci MCC.

DAFTAR PUSTAKA

[1] Anonim. **Mobile Operating System Market Share Indonesia**, <URL: https://gs.statcounter.com/os-market-share/mobile/indonesia> (dikunjungi pada: 22 Nov. 2019).

[2] Indonesia, A. P. J. I., 2018. "Penetrasi dan Profil Perilaku Pengguna Internet Indonesia". Versi S 20190518.

[3] Cukier, M. 2007. **Study: Hackers Attack Every 39 Seconds**, <URL: https://eng.umd.edu/news/story/study-hackers-attack-every-39-seconds> (dikunjungi pada: 22 Nov. 2019).

[4] McGuire, M. Apr. 2018. **Into the Web of Profit: Tracking the Proceeds of Cybercrime**.

[5] Munir, R. 2019. **Kriptografi**. Bandung, Indonesia: Informatika Bandung, vol. 2.

[6] Gunawan, I., S., Heru Satria T., Eka I., Hendry Q., dan Dedy H., 2019. "Combination of Caesar Cipher Algorithm and Rivest Shamir Adleman Algorithm for Securing Document Files and Text Messages". **Journal of Physics: Conference Series Series 1255 012077**. DOI: 10.1088/1742-6596/1255/1/012077.

[7] Purnama, I., Sudi S., Ronal W., Deci I., dan Marnis N., Mei 2019. "Android-Based Text Message Security Application With Rivest Method, Shamir, Adleman (RSA)". **International Journal of Scientific and Technology Research**, vol. 05, ISSN: 2277-8616. DOI: 10.31227/osf.io/f42xr.

[8] Gorn, S., R. W. B., dan J. G., 1963. "American standard code for information interchange". **Communications of the ACM**, vol. 6, no. 8, hal. 422–426. DOI: 10.1145/366707.367524.

[9] Anonim., <URL: http://www.openhandsetalliance.com/android%5C_overview.html> (dikunjungi pada: 11 Des. 2019).

[10] ——, <URL: http://www.openhandsetalliance.com/press%5C_110507.html> (dikunjungi pada: 11 Des. 2019).

[11] Manjoo, F., 2015. "A Murky Road Ahead for Android, Despite Market Dominance"., ISSN: 0362-4331.

[12] Anonim. Agust. 2018., <URL: https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html> (dikunjungi pada: 2 Juni 2019).

[13] ——, Mar. 2020., <URL: https://docs.oracle.com/javase/8/docs/api/java/util/Random.html> (dikunjungi pada: 2 Juni 2019).

[14] ——, **Android Developers**, <URL: https://developer.android.com/> (dikunjungi pada: 2 Juni 2019).

LAMPIRAN

Tabel ASCII

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	p

Kunci RSA

n:

2991700773293837644296628904047130859726411240256041383038402822047571401662406136123473269902211905303861406470805614745550396497594246808522588409400364561466555117353274658795174783017420887027348064887172155797896510525605024979875858629351279027361739221976037377703402839945370471090118641776042590719964239715330078705715023001665220990479443029200225250879404404 1

2334279195104726359693174772342568056027735983915323 8
3806919329641590059621175461647702380923809823086589 2
4626474307928035146488204891287119835710774261568312 3
6896398665674025836918296776494347217212140569456462 3
1786207380040505081992265946584989

d:

1546754724329298867178143375654067650683972137189759 7
8932311586659007671127986941016977875416540756577223 3
2223945602243053503378028687934245435815451617924691 9
0972963894792959744755310308974586691276107477375851 1
5559951523230168474481393672767479962047141204355902 6
6826925261412606578244423381440806119505069835376724 6
4942188970666990093946971782213523074872416787258171 7
0249049759179880949007787162306870047401804634991727 1
4630237667971812889181694201925073695485130221147149 0
2718942877134043096865654648359259082048544869990098 2
6999078449754081912108151041216820537100487090323154 9
3395145527495192419448838543751577

e:

1105824572284211601549132383991567361015083329981710 3
0828580514989867391574531158152512883388755777099656 1
2370087456719118998987493430249137035261389337673

Kunci MCCRSA

n:

2991700773293837644296628904047130859726411240256041 3
8303840282204757140166240613612347326990221190530386 1
4064708056147455503964975942468085225884094003645614 6
6555117353274658795174783017420887027348064887172155 7
9789651052560502497987585862935127902736173922197603 7
3777034028399453704710901186417760425907199642397153 3
0078705715023001665220990479443029200225250879404404 1
2334279195104726359693174772342568056027735983915323 8

3806919329641590059621175461647702380923809823086589246264743079280351464882048912871198357107742615683123689639866567402583691829677649434721721214056945646231786207380040505081992265946584989

d:

1546754724329298867178143375654067650683972137189759789323115866590076711279869410169778754165407565772233222394560224305350337802868793424543581545161792469190972963894792959744755310308974586691276107477375851155599515232301684744813936727674799620471412043559026682692526141260657824442338144080611950506983537672464942188970666990093946971782213523074872416787258171702490497591798809490077871623068700474018046349917271463023766797181288918169420192507369548513022114714902718942877134043096865654648359259082048544869990098269990784497540819121081510412168205371004870903231549339514552749519241944883854375157 7

e:

1105824572284211601549132383991567361015083329981710308285805149898673915745311581525128833887557770996561237008745671911899898749343024913703526138933767 3

MCC:

100