TUGAS AKHIR - KI141502

IMPLEMENTASI ENKRIPSI TEXT MENGGUNAKAN ALGORITMA ADVANCED ENCRYPTION STANDARD DAN ELLIPTIC CURVE CRYPTOGRAPHY

ARMIRARA REFA AZIZ
NRP 5113100174

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D.

Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

IMPLEMENTASI ENKRIPSI TEXT MENGGUNAKAN ALGORITMA ADVANCED ENCRYPTION STANDARD DAN ELLIPTIC CURVE CRYPTOGRAPHY

ARMIRARA REFA AZIZ
NRP 5113100174

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D.

Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES - KI141502

IMPLEMENTATION TEXT ENCRYPTION USING ADVANCED ENCRYPTION STANDARD AND ELLIPTIC CURVE CRYPTOGRAPHY ALGORITHM

ARMIRARA REFA AZIZ
NRP 5112100174

First Advisor
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Second Advisor
Bagus Jati Santoso, S.Kom., Ph.D.

Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI ENKRIPSI TEXT MENGGUNAKAN ALGORITMA ADVANCED ENCRYPTION STANDARD DAN ELLIPTIC CURVE CRYPTOGRAPHY

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur Jaringan Komputer
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
ARMIRARA REFA AZIZ
NRP: 5113100174

Disetujui oleh Pembimbing Tugas Akhir:

1. Henning Titi Ciptaningtyas, S.Kom., M.Kom. .....
   (NIP. 198407082010122004) (Pembimbing 1)

2. Bagus Jati Santoso, S.Kom., Ph.D. .....
   (NIP. 051100116) (Pembimbing 2)

SURABAYA
JULI, 2017

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI ENKRIPSI TEXT MENGGUNAKAN ALGORITMA ADVANCED ENCRYPTION STANDARD DAN ELLIPTIC CURVE CRYPTOGRAPHY

Nama Mahasiswa	**: ARMIRARA REFA AZIZ**
NRP	**: 5113100174**
Departemen	**: Teknik Informatika FTIf-ITS**
Dosen Pembimbing 1	**: Henning Titi Ciptaningtyas, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Bagus Jati Santoso, S.Kom., Ph.D.**

Abstrak

Keamanan data selalu menjadi perhatian khusus dalam era digital. Pengiriman data melalui jaringan internet memungkinkan seseorang melakukan pencurian data ditengah pengiriman. Enkripsi adalah metode untuk mengamankan sebuah data. Enkripsi mengubah data yang dapat dibaca menjadi data yang tidak dapat dibaca (ciphertext) menggunakan algoritma tertentu.

Advanced Encryption Standard (AES) adalah algoritma enkripsi simetrik yang memerlukan kunci yang sama untuk proses enkripsi dan dekripsi. Algoritma AES sering digunakan untuk mengamankan data yang besar karena cepat dan efisien tetapi memiliki kekurangan manajemen key yang komplek dan tidak aman. Elliptic Curve Cryptography (ECC) adalah algoritma enkripsi asimetrik yang memerlukan kunci yang berbeda untuk proses enkripsi dan dekripsi. ECC memiliki manajemen key dan keamanan yang baik namun hanya cocok untuk data berukuran kecil.

Penelitian ini menggabungkan metode AES dan ECC pada aplikasi berbasis website dengan cara data text dienkripsi menggunakan algoritma AES lalu AES key dienkripsi menggunakan algoritma ECC. Data dalam bentuk ciphertext adakan dikirimkan bersama dengan AES cipherkey melalui

jaringan umum. Proses enkripsi dan dekripsi akan dilakukan pada sisi client yaitu pada browser.

Kecepatan proses enkripsi dan dekripsi yang dilakukan di browser memiliki trendline naik seiring pertambahan ukuran file dan dipengaruhi oleh service lain yang sedang berjalan pada browser. Ukuran cipher mengalami penambahan dengan rata-rata 36,297% untuk text dan 76,835% untuk file.

Kata kunci: AES, ECC, ECIES, Dekripsi, Enkripsi, Keamanan

IMPLEMENTATION TEXT ENCRYPTION USING ADVANCED ENCRYPTION STANDARD AND ELLIPTIC CURVE CRYPTOGRAPHY ALGORITHM

Student's Name	**: ARMIRARA REFA AZIZ**
Student's ID	**: 5113100174**
Department	**: Informatics FTIF-ITS**
First Advisor	**: Henning Titi Ciptaningtyas, S.Kom., M.Kom.**
Second Advisor	**: Bagus Jati Santoso, S.Kom., Ph.D.**

Abstract

Data security is always be a special concern in digital era. Data transmitionover the internet network allows people to capture the data during submissions. Encryption is a method for secure data. Encryption convert readable data into non-readable data (ciphertext) by using particular algorithm.

Advanced Encryption Standard (AES) is a symmetric encryption algorithm that requires same key for encryption dan decryption. AES is often used to secure large data due to fast and efficient but has a lack of complex and unsafe key management. Elliptic Curve Cryptography (ECC) is an asymmetric encryption algorithm that require different key for encryption and decryption. ECC has good security and key management but it only suitable for small data.

This study combine AES and ECC algorithm in web based aplicataion by encrypting data text with AES algorithm and then encrypting AES key using ECC algorithm. The cipher data is sent along with AES cipher key over the public network. The encryption and decryption process will be done on the client side ie browser.

The running time of the encryption and decryption processes has a rising trendline as the file size increase and is influenced by other services running on the browser. The size of cipher has an

increment with an average up to 36.297% for text and 76.835% for files.

Keywords : AES, ECC, ECIES, Decryption, Encryption, Security

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "**Implementasi Enkripsi Text Menggunakan Algoritma Advanced Encrryption Standard dan Elliptic Curve Cryptography** ".

Tugas Akhir ini merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulisa selama menempuh perkuliahaan di Teknik Informatika ITS. Selain itu penulis berharap dapat memberikan kontribusi positif bagi bagi Teknik Informatika ITS.

Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT yang telah melimpahkan rahmat, hidayah dan inayah-Nya sehingga penulis mampu menyelesaikan tugas akhir.
2. Junjungan kita Nabi Muhammad SAW yang telah teladan yang baik bagi penulis sehingga termotivasi untuk menyelesaikan Tugas Akhir ini.
3. Keluarga penulis yang selalu memberikan dukungan doa, moral dan meterial sehingga penulis dapat menyelesaikan Tugas Akhir ini.
4. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. dan Bapak Bagus Jati Santoso, S.Kom., Ph.D. selaku dosen pembimbing penulis yang telah membimbing serta

memberikan nasehat dan arahan sehingga penulis dapat menyelesaikan Tugas Akhir ini.

5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
6. Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator Tugas Akhir.
7. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
8. Rekan-rekan laboratorim NCC yang selalu membantu ketika penulis menemukan kususahan dalam pengerjaan tugas akhir.
9. Rekan-rekan yang sama-sama mengambil Tugas Akhir yang selalu saling menyemangati, mengingatkan, dan menghibur dalam pengerjaan Tugas Akhir
10. Seluruh teman seperjuangan angkatan 2013, yang sama-sama menempuh perkuliahan di Teknik Informatika yang telah memtivasi dan memberikan pelajaran hidup yang berharga bagi penulis.
11. Sahabat-sahabaat penulis yang selalu mendukung, menghibur, dan tempat berbagi penulis.
12. Semua pihak yang belum sempat disebutkan satu-satu yang telah membantu terselesaikannya Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juni 2017

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengiriman data melalui internet memungkinkan seseorang melakukan pencurian data ditengah pengiriman. Keamanan data-data pada internet selalu menjadi perhatian khusus, teruta data-data yang bersifat rahasia. Bagaimana cara untuh mencegah pencurian dan sabotase data telah menjadi fokus perhatian [1]. Enkripsi data adalah metode yang penting untuk meningkatkan keamanan data. Enkripsi mengubah data yang dapat dibaca menjadi data yang tidak dapat dibaca menggunakan algoritma atau metode tertentu. Algoritma enkripsi dibagi menjadi *symmetrical encryption* dan *asymmetrical encryption*.

Untuk waktu yang lama, Data Encryption Standard (DES) menjadi standard untuk *symmetric key encryption*. DES memiliki ukuran *key* sepanjang 56 bit. Panjang ukuran *key* ini terlalu kecil dan mudah untuk diserang. Pada tahuun 2001, National Institute of Standards and Technology (NIST) mengumumkan *block cipher* Rijndael sebagai AES. Advanced Encryption Standard (AES) yang merupakan pengembangan dari algoritma Data Encryption Standard (DES) dapat memiliki *key* sepanjang 128, 192, dan 256 bit [2]. Masing-masing ukuran kunci menggunakan jumlah *round* yang berbeda. 10 *round* untuk 128 bit, 12 *round* untuk 192 bit, dan 14 *round* untuk 256 bit [3, 4]. Algoritma AES yang cepat cocok untuk enkripsi text yang panjang, tetapi memiliki kekurangan yaitu manajemen *key* sangat komplek dan tidak aman [1].

Manajemen *key* pada algoritma *asymmetric encryption* lebih sederhana, terutama untuk pengiriman dara melalui internet. Namun hanya dapat digunakan untuk data berukuran kecil. Salah satu algoritma *asymmetric encryption* adalah *Elliptic Curve Cryptography* (ECC). ECC menyediakan keamanan dengan tinggi dengan ukuran kunci yang lebih kecil dibandingkan dengan metode kriptografi yang lain [5]. *Symmetric* dan *asymmetric*

encryption memiliki keunggulan dan kekurangan masing-masing [6].

Mengambil keunggulan masing-masing algoritma dan menggabungkannya diharapkan akan menghasilkan metode enkripsi yang lebih aman dan efisien. *ECC* yang mudah dalam manajemen *key* cocok unuk enkripsi *key*. Menggabungkan AES yang memiliki efisiensi tinggi dan cocok untuk enkripsi *plaintext* yang panjang untuk enkripsi plaintext dengan ECC yang mudah dalam manajemen *key* cocok unuk enkripsi AES *keyblock* diharapkan dapat mengamankan pertukaran *key* dan meningkatkan keamanan *cipertext*.

1.2 Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana penggabungan algoritma *Advance Encryption Standart* (AES) dan *Elliptic Curve Encryption* (ECC)?
2. Bagaimana hasil dari penggunaan gabungan metode *Advance Encryption Standart* (AES) dan *Elliptic Curve Encryption* (ECC)?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Data yang dienkripsi adalah *text.*
2. Bahasa pemrograman yang digunakan untuk implementasi adalah *javascript*.
3. Implementasi dilakukan pada sisi client pada aplikasi berbasis website.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Menggabungkan metode enkripsi *Advanced Encryption Standard* (AES) dan *Elliptic Curve Cryptography* (ECC).
2. Membuat suatu perangkat lunak berbasis web dengan mengimplementasikan penggabungan metode *Advanced Encryption Standard* (AES) dan *Elliptic Curve Cryptography* (ECC).

1.5 Manfaat

Pengerjaan tugas akhir ini memiliki manfaat unutk menghasilkan sebuah perangkat lunak berbasis *website* yang dapat mengamankan data pesan yang dikirmkan pengguna kepada pengguna lainnya.

Pengerjaan tugas akhir ini juga memiliki manfaat bagi penulis. Manffat bagi penulis yaitu sebagai sarana unuk mengimplementasikan yang telah dipelajari selama kuliah agar berguna bagi masyarakat.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagi berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal, berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Prosposal juga berisi tentang garis besa tugas akhir yang akan dikerjakan sehingga memberikan gambaran untuk dapat mengerjakan tugas akhir sesuai dengan *timeline* yang dibuat.

1.6.2 Studi Literatur

Pada tahap ini dilakukan untuk mencari informasi dan studi literatur apa saja yang dapat dijadikan sebagai referensi untuk membantu pengerjaan tugas akhir ini. Tahap ini merupakan tahap untuk memahami semua metode yang akan dikerjakan, sehingga memberi gambaran selama pengerjaan tugas akhir. Informasi didapatkan dari buku dan literatur yang berhubungan dengan metode yang digunakan.

1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk mengimplementasikan metode-metode yang sudah diajukan pada proposal Tugas Akhir. Untuk membangun algoritma yang telah dirancang sebelumnya, implementasi dilakukan dengan menggunakan bahasa pemrograman javascript.

1.6.4 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian fungsionalitas perangkat lunak sesuai dengan perancangan kasus penggunaan dan pengujian performa penggabungan algoritma enkripsi AES dan ECC.

1.6.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan

   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir

2. Bab II. Tinjauan Pustaka

Bab ini berisi kajian teori dari metode dan teknologi yang digunakan dalam penyusunan tugas akhir ini. Secaca garis besar, bab ini berisi tentang metode AES, AES CTR, ECC, ECIES, javascript, NodeJS, MySQL, dan Intellij IDEA.

3. Bab III. Perancangan Perangkat Lunak

   Bab ini berisi pembahasan mengenai perancangan dari perangkat lunak berbasis web dan implementasi penggabungan metode *Advanced Encryption Standard* dan *Elliptic Curve Cryptography*.

4. Bab IV. Implementasi

   Bab ini menjelaskan implementasi dari pengembangan perangkat lunak dan metode enkripsi yang digunakan.

5. Bab V. Pengujian dan Evaluasi

   Bab ini berisikan hasil uji coba fungsionalitas perangkat lunak sesuai dengan kasus penggunaan dan hasil uji performa dari penggabungan metode *Advanced Encryption Standard* dan *Elliptic Curve Cryptography.*

6. Bab VI. Kesimpulan dan Saran

   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan tugas akhir, dan saran untuk pengembangan solusi ke depannya.

7. Daftar Pustaka

   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

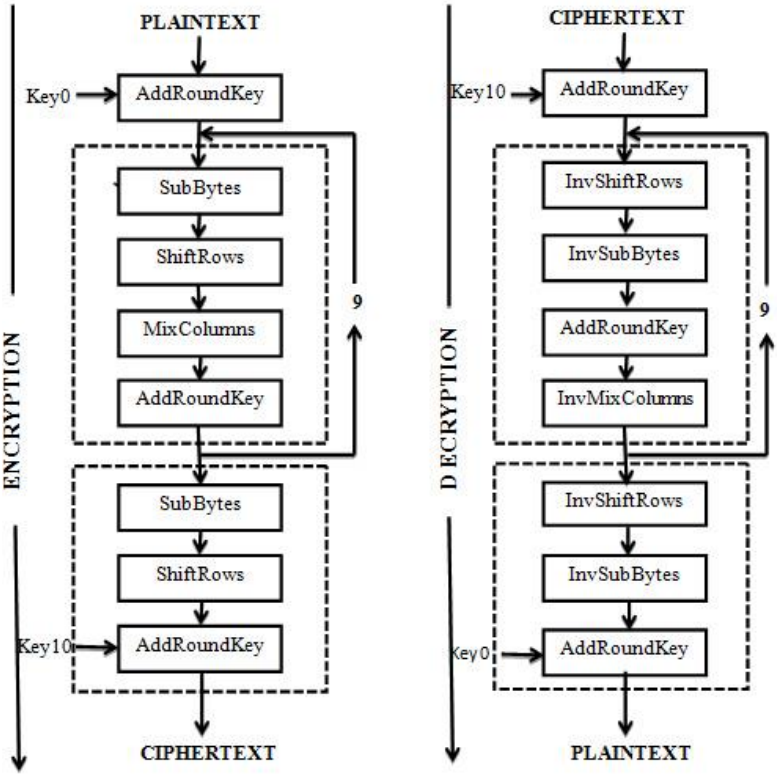
Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam tugas akhir. Penjelesan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 *Advanced Encryption Standard*

Advanced Encryption Standard (AES) adalah Federal Information Processing Standard (FIPS) yang dipublikasikan oleh National Institute of Standards and Technology (NIST) pada tahun 2001. AES adalah salah satu teknik enkripsi yang paling banyak dipakai kerana tingkat efisiensiyang tinggi dan sederhana. AES juga merupakan algoritma yang aman.

AES, Advanced Encryption Standart, adalah *symmetric encryption* [2]. *Symetric* berarti AES mengunakan *key* yang sama untuk proses enkripsi dan dekripsi. AES adalah *block-cipher* yang mengenkripsi 128-bit blok (*plaintext*) menjadi 128-bit blok (*ciphertext*), atau mendekripsi 128-bit blok (*ciphertext*) menjadi 128-bit blok (*plaintext*). AES menggunakan kunci (*cipher key*) berukuran 128, 192, atau 256 bit [3]. Masing-masing ukuran kunci menggunakan jumlah *round* yang berbeda. 10 *round* untuk 128 bit, 12 *round* untuk 192 bit, dan 14 *round* untuk 256 bit [7].

Setiap iterasi pada AES terdiri dari empat langkah seperti ditunjukkan pada Gambar 2.1.



Gambar 2.1 Proses Enkripsi dan Dekripsi pada AES

Setiap langkah tersebut dapat dijabarkan sebagai berikut:

1. *SubBytes* adalah permutasi non-linier pada S-Box. Setiap *byte* input digantikan oleh *byte* lain berdasarkan table *lookup*.
2. *ShiftRow* adalah proses yang melakukan *shift* atau penggeseran pada setiap elemen blok atau table yang dilakukan per baris.

3. *MixColumns* adalah mengalikan setiap elemen dari blok *cipher* dengan matriks biasa yaitu menggunakan *dot product*. Setiap byte akan digantikan oleh hasil operasinya.
4. *AddRoundKey* adalah menggabungkan *cipher text* dengan *cipher key* mengunakan operasi XOR.

Pada system 32-bit atau lebih, memungkinkan untuk mempercepat eksekusi dengan menggabungkan *SubByte* dan *ShiftRow* dengan *MixColloum*, dan mengubahnya menjadi *sequence table lookup* [8].

Performa algoritma AES yang cepat cocok untuk enkripsi *text* yang panjang. Namun memiliki kekurangan yaitu manajemen *key* sangat komplek dan tidak aman. Pada tugas akhir ini, algoritma AES akan digunakan untuk enkripsi data text. Mode operasi yang digunakan adalah *Counter Mode*.

2.2 *Counter Mode*

Mode operasi AES yang digunakan dalam tugas akhir ini adalah *Counter Mode*. AES *Counter Mode* (CTR) distadardisasi pada 2001 oleh NIST pada SP 800-38A [9]. CTR menggunakan *counter* daripada *IV* pada mode AES yang lain. Counter memiliki atribut tambahan berupa *nonce* dan *counter block*. Mode ini tidak memerlukan padding untuk plain text ke ukuran block cipher. Plaintext akan dibagi menjadi *block* dan

Proses enkripsi setiap blok dapat dijelaskan sebagai berikut

$$O_j = CIPH_k(T_j) \qquad \text{for j = 1, 2 … n;}$$

$$C_j = P_j \oplus O_j \qquad \text{for j = 1, 2 … n-1;}$$

$$C_n = P_n \oplus MSB_u(O_n)$$

Pada enkripsi CTR, fungsi cipher dipanggil pada setiap blok *counter* dengan masukan berupa *counter block* dan *key*. dan hasil keluaran tiap blok di XOR kan dengan blok plaintext yang sesuai untuk menghasilkan blok ciphertext. Untuk blok terakhir, yang kemungkinan hanya berukuran sebagian dari ukuran blok. Bits

paling signifikan dari hasil blok terakhir digunakan untuk operasi XOR, sisanya akan diabaikan.

Untuk proses dekripsi counter mode hanya perlu membalik proses pada proses enkripsi. Ditulis dalam persamaan sebagai berikut

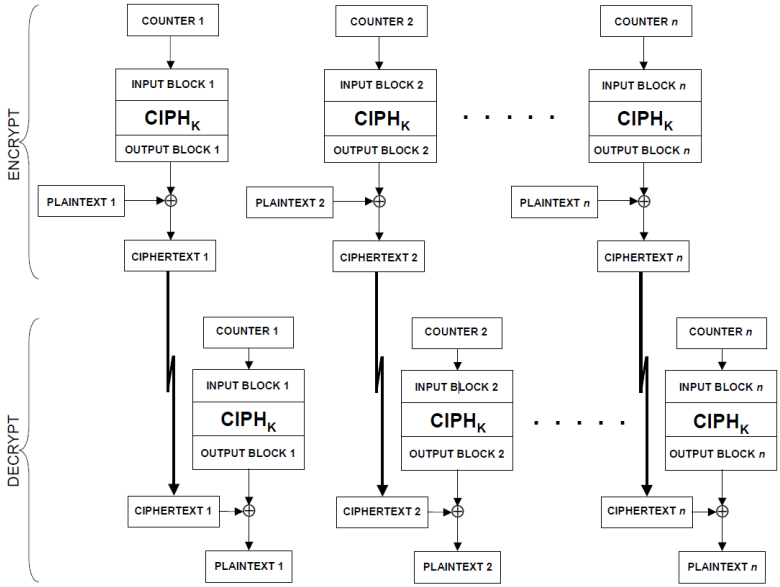
$$O_j = CIPH_k(T_j) \qquad \text{for j = 1, 2 … n;}$$

$$P_j = C_j \oplus O_j \qquad \text{for j = 1, 2 … n-1;}$$

$$P_n = C_n \oplus MSB_u(O_n)$$

Pada proses dekripsi CTR, fungsi cipher dipanggil pada setiap *counter block* dengan masukan berupa *cipher block* dan *key* dan hasil *output block* dilakukan operasi XOR dengan ciphertext yang sesuai untuk mengembalikan ke *plaintext block*. Untuk blok terakhir, yang kemungkinan hanya berukuran sebagian dari ukuran blok. Bits paling signifikan dari hasil blok terakhir digunakan untuk operasi XOR, sisnya akan diabaikan.

Pada Kedua proses enkripsi dan dekripsi, block cipher selanjutnya dapat dilakukan secara parallel. [9] Diagram Counter Mode ditunjukkan pada Gambar 2.2.



Gambar 2.2. Counter Mode [9]

2.3 *Elliptic Curve Encryption*

Elliptic Curve Cryptography (ECC) adalah *public key cryptography* yang diciptakan oleh Victor Miller dan Neal Kobliz pada tahun 1985. ECC menyediakan keamanan dengan tinggi dengan ukuran kunci yang lebih kecil dibandingkan dengan metode kriptografi yang lain [5]. ECC memiliki bentuk persamaan

$$y^2 = x^3 + ax + b \qquad (1)$$

yang dikenal dengan persamaan *Weirerstrass*, dimana a dan b adalah konstan dengan

$$4a^3 + 27b^2 \neq 0 \qquad (2)$$

Salah satu keuntungan utama dari ECC adalah ukuran *key* yang kecil. 160-bit *key* pada ECC dianggap memiliki tingakat keamanan yang setara dengan 1024-bit pada RSA [10]. Karena ECC merupakan *public key cryptography*, diperlukan *public key* dan *private key*. Anggap Alice dan Bob berkomunikasi menggunakan ECC dan *generator G*. *Private key* Alice dan Bob berurutan adalah *n A* dan *nB*. *Public key* yang diberikan adalah

$$P_a = nAG \quad (3)$$

dan

$$P_b = nBG \quad (4)$$

Jika Alice ingin mengirimkan pesan '*Pm*' kepada Bob, Alice menggunakan *public key* Bob untuk enkripsi pesan. *Ciphertext* yang dihasilkan

$$P_c = \{kG, P_m + kP_b\} \quad (5)$$

k adalah *integer* acak. *k* berbeda setiap pengiriman pesan. Bob medekripsi pesan dengan mengurangkan koordinat '*kG*' dikalikan dengan *nB* from '*Pm* + *kPb*'.

$$P_m = \{P_m + kP_b - nBkG\} \quad (6)$$

ECC memiliki kelebihan

1. Masalah keamanan pada distribusi kunci sangat baik.
2. Manajemen kunci lebih baik karena ukuran kunci yang lebih kecil

ECC tidak memiliki fungsi enkripsi sendiri seperti pada RSA dan *asymmetric encryption* lainnya. ECC menangani *key agreements*, *digital signature*, *pseudo-random generator* dan fungsi lainnya. Secara tidak lagsung ECC dapat digunakan untuk proses enkripsi dengan menggabungkan *key agreement* dengan

symmetric encryption. Beberapa variasi dari metode ECC adalah E*lliptic Curve Diffie–Hellman* (ECDH) dan *Elliptic Curve Integrated Encryption Scheme* (ECIES). Tugas akhir ini kan memanfaatkan kedua varasi tersebut untuk proses enkripsi. ECDH adalah skema *key agreement* dan ECIES adalah skema untuk enkripsi.

2.4 Elliptic Cureve Integrated Encryption Scheme

Pada ahun 1997, Mihir Bellare dan Philip Rogaway memperkenalkan *Discrete Logarithm Augmented Encryption Scheme* (DLAES), yang kemudian dikembangkan oleh orang yang sama dan Michel Abdalla menjadi *Diffie-Hellman Augmented Encryption Scheme* (DHAES) pada tahun 1998 dan kemudian *Diffie-Hellman Integrated Encryption Scheme* (DHIES) pada tahun 2001 unutk menghindari kebingungan dengan *Advanced Encryption Standard* (AES). DHIES merupakan versi yang dikembangkan dari *ElGarnal encryption scheme* menggunakan *elliptic curve* yang melibatkan operasi *public key*, algorima enkripsi dan *Message Authentication Code* (MAC) dan *hash*. Karena menggunakan integrasi dari fungsi-fungsi yang berbeda, DHIES aman terhadap *chosen ciphertext attack* tanpa harus meningkatan jumlah operasi atau panjang kunci.

ECIES merupakan skema *hybrid encryption* berdasarkan *Diffie-Hellman Problem*. ECIES adalah pola enkripsi yang terintegrasi yang menggunakan beberapa fungsi berikut:

1. *Key Agreement* (KA)
   Key Agreement adalah fungsi untuk pembuatan *shared secret* oleh dua pihak.
2. *Key Derivation* (KDF)
   Key Derivation adalah mekanisme untuk menghasilkan *public key* dari *private key* dan beberapa parameter pilihan.
3. *Hash* (HASH)
   Digest fungtion.

4. *Symmetric Encryption*
   Algoritma enkripsi simetrik dibutukan untuk enkripsi pesan dengan menggunakan *key* yang dibuat dari hasi proses *Key Agreement.*
5. *Message Authentication Code* (MAC)
   Message Authentication Code adalah informasi yang digunakan untuk autentikasi sebuah pesan.

ECIES memanfaatkan E*lliptic Curve Diffie–Hellman* (ECDH) sebagai *key agreement* dan menggunakan *symmentric encryption* untuk proses enkripsi. Pada tugas akhir ini ECIES akan digunakan untuk enkripsi AES *cipher key* yang telah digunakan untuk enkripsi data *text*.

2.5 Javascript

Javascript adalah bahasa pemrograman level tinggi, dinamis, Javascript telah distandardisasi pada spesifikasi ECMAScript language. Javascript merupakan salah satu dari 3 teknologi utama pada produksi konten *World Wide Web*. Javascript memiliki API untuk menangani text, arrays, dates, dan expression, tetapi tidak termasuk I/Oseperti, networking, storage, dan grafik. [11]

Javascript adalah bahasa pemrograman yang bisa melakukan komputasi pada browser. Berdasarkan kemampuan ini, bahasa pemrograman javascript akan digunakan untuk implementasi algoritma AES dan ECC yang dijalankan browser.

2.6 Node JS

Node.js adalah sebuah platform yang dibuat di atas *Chrome JavaScript Runtime*. NodeJS menggunakan event-driven, non-blocking I/O model yang membuat ringan dan efisien. NodeJS memiliki *library* web server sehingga dapat digunakan sebagai web server tanpa menggunakan web server pada umumnya sehingga menjadi lebih ringan. [12]

Sama seperti bahasa pemrograman lain seperti PHP, NodeJS memliki *package manager* yaitu *Node Package Manager* (NPM).

NPM digunakan untuk mengistal package tambahan yang tidak terdapat pada modul NodeJS standard. NPM ini merupakan *open source library* yang membuat semua orang bisa membuat *package* sendiri.

Karena menggunakan bahasa pemrograman javascript, NodeJS memiliki kemampuan untuk menangani banyak *request* sekaligus. Hal ini dikarenankan *javascript* bersifat *asynchronous.* Dengan kemampuan ini NodeJS seringkali digunakan untuk pengembangan perangkat lunak yang memiliki menangani banyak request. NodeJS juga tidak memerlukan pengaturan *resource* pada setiap *thread* yang cocok digunakan sebagai *web server*. NodeJS juga akan digunakan pada perangkat lunak ini sebagai *web server*.

2.7 MySQL

MySQL adalah perangkat lunak yang digunakan untuk manajemen basis data. MySQL AB menyediakan MySQL dalam versi gratis dabawah lisensi GNU *General Public License* (GPL). MySQL tersedia pada platform Windows, Linux, dan Mac OS. [13]

MySQL memiliki kelebihan antara lain *open source*, skalabilitas, *multi-user*, dan keamanan yang cukup baik. MySQL akan digunakan sebagai manajemen basis data pada tugas akhir ini.

2.8 Intellij IDEA

Intellij IDEA adalah sebuah IDE yang sebenarnya dikhususkan untuk bahasa pemrograman *java*. Namun Intellij IDEA memiliki banyak *plug-in* yang bisa menyesuaikan dengan bahasa pemrograman lain termasuk *javascript*. [14] Dengan kemampuan ini Intellij IDEA menjadi IDE yang fleksibel. Intellij IDEA juga merupakan perangkat lunak yang cerdasyang dapat memberikan *suggestion* sesuai dengan bahasa pemrograman yang sedang dikerjakan. Pada tugas akhir ini akan menggunakan Intellij IDEA sebagai perangkat pengembang.

[Halaman ini sengaja dikosongkan]

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan dan pembuatan perangkat lunak yang akan dibangun pada tugas akhir. Perancangan perangkat lunak yang akan dibahas adalah mengenai dekripsi umum system, proses perancangan, dan antarmuka.

3.1 Deskripsi Umum Sistem

Pada Tugas akhir ini akan dibangun suatu sitem pengiriman pesan berbasis web. Perangkat lunak akan mengimplementasikan metode enkripsi gabungan *Advanced Encryption Standard* dan *Elliptic Curve Cryptography*. AES merupakan algoritma *Symmetric-key* yang memiliki performa yang cepat dan dapat digunakan pada data berukuran besar. Metode enkripsi AES akan digunakan untuk enkripsi pesan. ECC merupakan algoritma *asymmetric-key* yang akan digunakan untuk enkripsi *cipher key* yang digunakan pada metode AES.

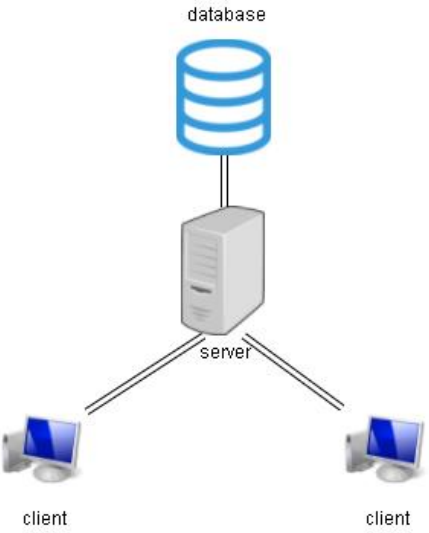
Proses enkripsi dilakukan dengan cara membangkitkan kunci random yang akan digunakan pada enkripsi AES. Pesan yang akan dikrimkan akan dienkripsi menggunakan algoritma AES dengan kunci tersebut. Kemudian menggunakan algoritma ECC untuk men-enkripsi kunci random yang digunakan pada algoritma AES. Algoritma ECC akan dilakukan sebanyak dua kali untuk masing-maing penerima dan pengirim pesan. Untuk proses dekripsi membutuhkan dua proses: pertama dekripsi kunci random yang digunakan pada algoritma AES menggunakan algoritma ECC, kemudian dekripsi pesan menggunakan algoritma AES.

Proses enkripsi dan dekripsi akan dilakukan pada sisi *client*. Data text akan diproses terlebih dahulu sebelum dikrimkan ke server. Dengan proses enkripsi dan dekripsi yang dilakukan pada sisi *client*, sistem ini dapat diimplementasikan pada jaringan *public*. Data yang disimpan dalam database juga berupa cipher text. Diharapkan dengan adanya system ini, keamanan data dapat

ditingkatkan mengingat pihak server juga tidak dapat membaca data tersebut.

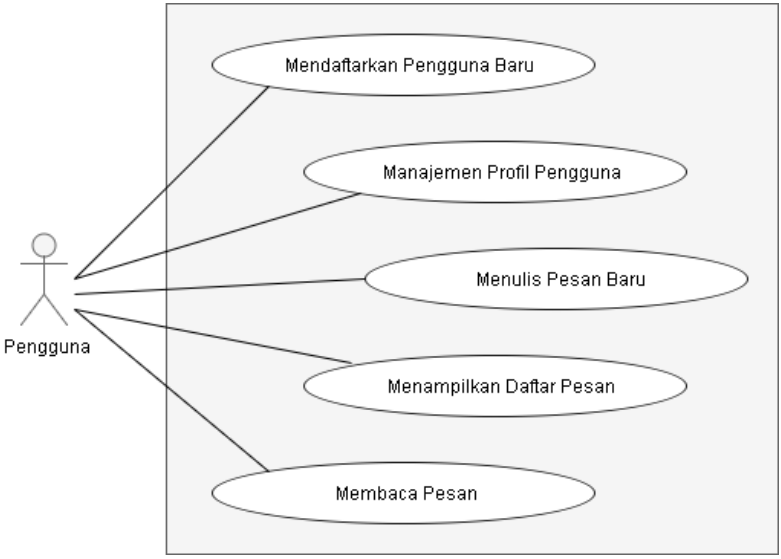
3.2 Desain Umum Sistem

Perangkat lunak ini menggunakan arsitektur client-server. Rancangan arsitektur umum dari sistem dapat dilihat pada Gambar 3.1. Pada perangkat lunak ini akan menggunakan Node JS sebagai web server. Basis data yang akan digunakan adah MySQL.



Gambar 3.1. Arsitektur Sistem

3.3 Perancangan Fungsionalitas Sistem



Gambar 3.2 Diagram Kasus Penggunaan

Pada bagian ini akan dijelaskan rincian kasus penggunaan yang akan dibangun dalam perangkat lunak. Secara umum kasus penggunaan yang akan dibuat pada perangkat lunak terdiri dari beberapa hal yaitu mendafttarkan pengguna baru, manajemen profile pengguna, menulis pesan baru, menampilkan daftar pesan masuk, dan membaca pesan. Diagram Kasus Penggunaan ditunjukkan pada Gambar 3.2. Berikut rincian dari setiap kasus penggunaan yang ada:

3.3.1 Mendaftarkan Pengguna Baru

Pada kasus penggunaan ini, calon pengguna baru dapat mendaftarkan diri dengan mengisi identitas diri pada form yang tersedia. Data yang harus dimasukkan adalah *username*, *password*, nama lengkap, *email*, dan *secret key*. *Secret key* tidak akan disimpan pada database dan hanya digunakan untuk membangkitkan *public key.*

3.3.2 Manajemen Profil Pengguna

Pada kasus penggunaan ini, pengguna yang telah terdaftar dapat mengubah data diri dan menambahkan foto profil dengan memilih menu *edit* dan mengisi data baru pada masing-masing form yang tersedia.

3.3.3 Menulis Pesan Baru

Pada kasus penggunaan ini, pengguna dapat mengirimkan pesan baru kepada pengguna lain. Masukan yang tersedia adalah penerima, subyek pesan, teks pesan, dan lampiran. Untuk mengirimkan pesan, pengguna harus memasukkan username yang valid dari penerima. Sistem kemudian akan mengecek apakah username tersebut valid atau tidak. Jika valid pesan akan dikrimkan kepada penerima. Pesan dan lampiran akan dilakukan enkripsi terlebih dahulu sebelum dikirimkan.

3.3.4 Menampilkan Daftar Pesan

Pengguna dapat melihat daftar pesan yang ditujukan untuknya dengan memilih menu pesan masuk dan daftar pesang yang telah terkirim dengan memilih menu pesan terkirim. Pada dafttar pesan ini akan ditampilkan tanggal pengiriman pesan, penerima/pengirim pesan, dan subyek pesan. Penguna dapat melakukan pencarian pada daftar tersebut.

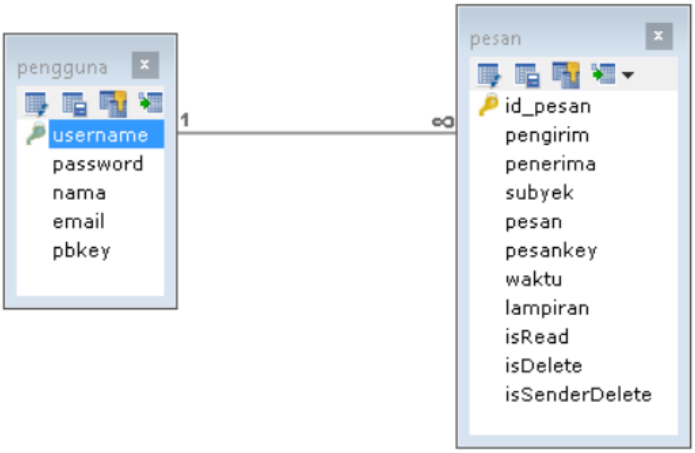
3.3.5 Membaca Pesan

Pengguna dapat membaca baik pesan dan lampiran yang ditujukan kepadanya atau yang dikrim olehnya. Pada halaman ini, akan ditampilkan detail pesan yaitu penerima, pengirim, subyek pesan, isi pesan, lampiran, dan tanggal pengiriman pesan. Pesan dan lampiran akan didekripsi terlebih dahulu sebelum ditampilkan.

3.4 Perancangan Basis Data Sistem

Perancangan basis data adalah merancang basis data yang akan dugunakan untuk menyimpan data yang digunakan pada perangkat lunak. Sistem Manajemen basis data yang digunakan

pada perankat lunak ini adalah MySQL. Skema dari basis data yang digunakan ditunjukkan pada Gambar 3.3.



Gambar 3.3 Skema Basis Data

Berikut penjelasan dari detail setiap entitas dan atribut dalah basis data.

3.4.1 Tabel Pengguna

Tabel pengguna digunakan untuk menyimpan data pengguna. Atribut yang terdapat pada tabel pengguna yaitu username, password, nama, email, dan pbkey. Penjelasan lebih detail setiap atribut dapat dilihat pada Tabel 3.1.

Tabel 3.1 Detail Tabel pengguna

No	Nama Atribut	Tipe Data	Keterangan
1.	*usernmame*	*varchar(50)*	Merupakan *primary key* dari tabel prngguna. *username* dari pengguna yang digunakan sebagai pengenal untuk masuk (login) ke dalam website dan juga sebagai

No	Nama Atribut	Tipe Data	Keterangan
			identitas saat pengiriman dan penerimaan pesan.
2.	*password*	*varchar(32)*	*password* dari Pengguna yang digunakan sebagai kode pengaman untuk masuk (login) ke dalam website.
3.	nama	*varchar(255)*	Nama lengkap dari pengguna yang didaftarkan saat pendaftaran.
4.	*email*	*varchar(100)*	Merupakan *email* didaftarkan saat proses regsitrasi
5.	pbkey	*varchar(255)*	Merupakan *publickey* pengguna yang akan digunakan untuk proses enkripsi pada ssaat pengiriman pesan. Proses pembangkitan *publickey* dilakukan pada saat pendaftaran menggunakan *secretkey* yang dimasukkan oleh pengguna.

3.4.2 Tabel pesan

Tabel pesan digunakan untuk menyimpan data pesan yang dikirimkan oleh pengguna. Atribut yang terdapat pada tabel pesan yaitu id_pesan, pengirim, penerima, subyek, pesan, recieverkey, senderkey, waktu, lampiran, *isRead*, *isDelete*, *isSenderDelete*. Penjelasan lebih detail setiap atribut dapat dilihat pada Tabel 3.2

Tabel 3.2 Detail Tabel Pesan

No	Nama Atribut	Tipe Data	Keterangan
1.	id_pesan	*varchar(10)*	*Primary key* dari tabel pesan. Id_pesan merupan attribut unik yang dibangkitkan dengan membuat 10 karakter acak.

No	Nama Atribut	Tipe Data	Keterangan
2.	pengirim	*varchar(50)*	Merupakan *foreign key* yang terhubung dengan tabel pengguna. Satu user dapat mengirimkan banyak pesan.
3.	penerima	*varchar(50)*	Merupakan *foreign key* yang terhubung dengan tabel pengguna. Satu user dapat menerima banyak pesan.
4.	subyek	*Text*	Merupakan subyek dari pesan yang dikirimkan.
5.	pesan	*text*	Merukan isi dari pesan yang dikirimkan.
6.	recieverkey	*text*	Merupakan kunci yang akan digunakan penerima untuk dekrisi pesan. Attribut ini adalah kunci pada algoritma AES yang telah dienkripsi menggunakan algoritma ECC.
7.	senderkey	*text*	Merupakan kunci yang akan digunakan pengirim untuk dekrisi pesan. Attribut ini adalah kunci pada algoritma AES yang telah dienkripsi menggunakan algoritma ECC.
8.	Waktu	*timestamp*	Merupakan waktu pengiriman pesan.
9.	Lampiran	*text*	Merupakan *path* dari dokumen lampiran pesan.
10.	*isRead*	*tinyint(1)*	Merupakan penanda untuk menandakan

No	Nama Atribut	Tipe Data	Keterangan
			apakah pesan telah dibaca oleh penerima.
11.	*isDelete*	*tinyint(1)*	Merupakan penanda untuk menandakan apakah pesan telah dihapus oleh penerima.
12.	*isSenderDelete*	*tinyint(1)*	Merupakan penanda untuk menandakan apakah pesan telah dihapus oleh pengirim.

3.5 Perancangan Proses

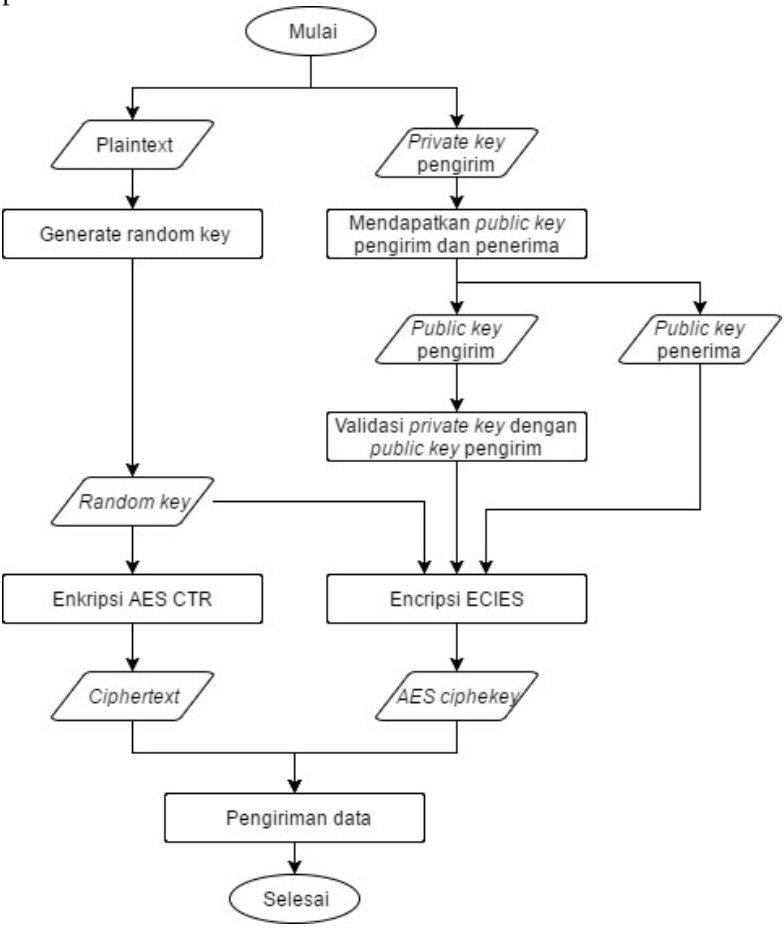
Pada subbab ini akan dijelaskan mengenai proses enkripsi dan dekripsi yang dilakukan pada perangkat lunak serta prosedur pengiriman pesan dan membaca pesan. Proses ini akan dilakukan di *browser* pada sisi *client*. Bahasa pemrograman ang digunakan adalah *javascript*.

3.5.1 Proses Enkripsi Keseluruhan

Proses enkripsi dimulai setelah pengguna menuliskan pesan dan memasukkan *private key* miliknya. Untuk melakukan enkripsi sistem memerlukan *public key* penerima pesan. Sistem mendapatkan *public key* pengirim dan *public key* penerima menggunakan API yang tersedia. *Public key* pengirim akan digunakan untuk validasi *private key*. Sistem akan membuat kunci acak sepanjang 32 *byte* yang akan digunakan pada AES counter mode. Hasil dari proses enkripsi perama adalah pesan dalam bentuk *ciphertext*.

Setelah kunci tersebut digunakan untuk melakukan proses enkripsi, kunci tersebut akan di enkripsi sebanyak dua kali. *Cipher key* pertama akan dienkripsi menggunakan *private key* pengirim dan *public key* pengirim. *Cipher key* pertama ini akan digunakan sebagai salinan untuk pengirim pesan untuk membaca pesan. *Cipher key* kedua akan dienkripsi menggunakan *private key* penerima dan *public key* pengirim. *Cipher key* kedua ini akan

digunakan sebagai salinan untuk penerima pesan untuk membaca pesan. Diagram Alir proses enkripsi secara umum ditunjukkan pada Gambar 3.4.



Gambar 3.4 Diagram Alir Proses Enkripsi

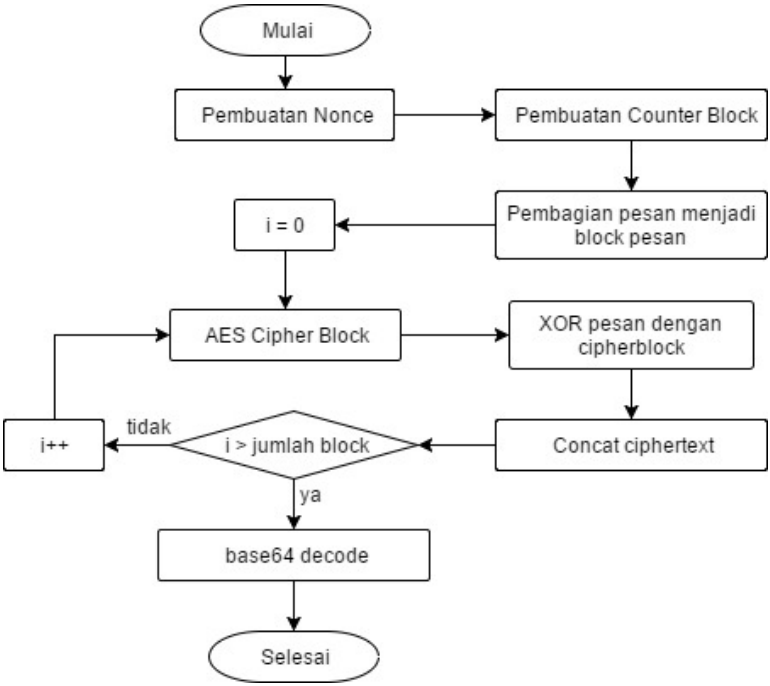
Cipher text beserta kedua *cipher key* tersebut akan dikirimkan bersamaan dengan data pada *form* menulis pesan.

Penjelasan lebih detail mengenai proses enkripsi AES dan ECC akan dibahas pada subbab selanjutnya.

3.5.1.1 Enkripsi AES *Counter Mode*

Pada perangkat lunak akan menggunakan AES-256. Proses enkripsi menggunakan metode AES *Counter* Mode (CTR) digunakan pada plaintext pesan dan file lampiran.

Plaintext akan dibagi menjadi menjadi block. Setiap block terdiri dari 16 byte dan akan dikonversikan menjadi array 2 dimensi berukuran 4 x 4. Kemudian dilakukan proses iterasi sesuai dengan panjang kunci. Untuk AES-256 menggunakan kunci sepanjang 32 byte dan jumlah round sebanyak 14. Diagram alir fungsi AES Cipher Block ditunjukkan pada Gambar 3.5



Gambar 3.5 Diagram Alir Enkripsi AES CTR

Pada mode *counter* tidak memerlukan *IV* seperti pada mode lainnya melainkan *counter block* yang terdiri dari 8 *byte nonce* dan 8 *byte counter*. *Nonce* akan dibuat berbeda setiap kali nekripsi. Pembuatan *nonce* berdasarkan waktu saaat proses enkripsi dimulai. Nonce akan terdiri dari gabungan 2 byte milidetik, 2 byte angka random, dan 4 byte detik. *Counter* adalah iterasi jumlah *counter block* yang dimulai dari 0.

Key yang dibangkitkan secara acak akan dilakukan proses *key expansion* terlebih dahulu. Hasi dari *key expansion* adalah *key schedule* yang akan digunakan pada fungsi AES *cipher block*. Hasil dari proses iterasi *cipher block* akan di *decode* menggunakan *decoder base64* untuk menghasilkan karakter yang bisa ditampilkan oleh computer.

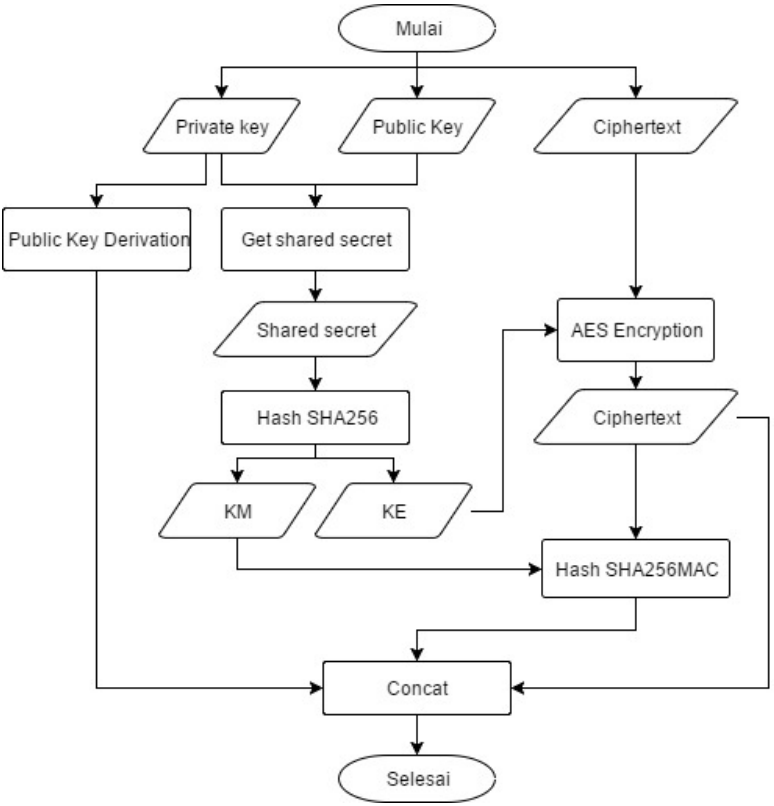
3.5.1.2 Enkripsi ECIES

Algoritma enkripsis ECIES akan digunakan untuk men-enkripsi key yang digunakan pada algoritma AES CTR.

Proses enkripsi ECIES membutuhkan *private key* pengirim dan *public key* penerima pesan. Pasangan kunci tersebut akan digunakan untuk menentukan *shared secret*. Pembentukan *shared* secret akan menggunakan aloritma ECDH menggunnakan kurva secp256k1. *Shared secret* didapatkan dari perkalian *private key* dengan titik dari *public key* pada kurva. Dari *shared secret* tersebut akan dilakukan fungsi hash SHA512. Hasil dari hash ini akan dibagi menjadi dua masing-masing sepangjang 32 byte buffer. Bagian perttama akan digunakan sebagai *key* untuk enkripsi menggunakan *symmetric encryption* yang pada tugas akhir ini menggunakan AES CTR. Bagian kedua dari fungsi hash tadi akan digunakan untuk membuat *Message Authentication Code* (MAC). Bagian ini akan digabungkan dengan pesan yang telah dienkripsi menggunkan *symmetric encryption* dan dilakukan fungsi hash SHA256 untuk mendapatkan 32 byte buffer.

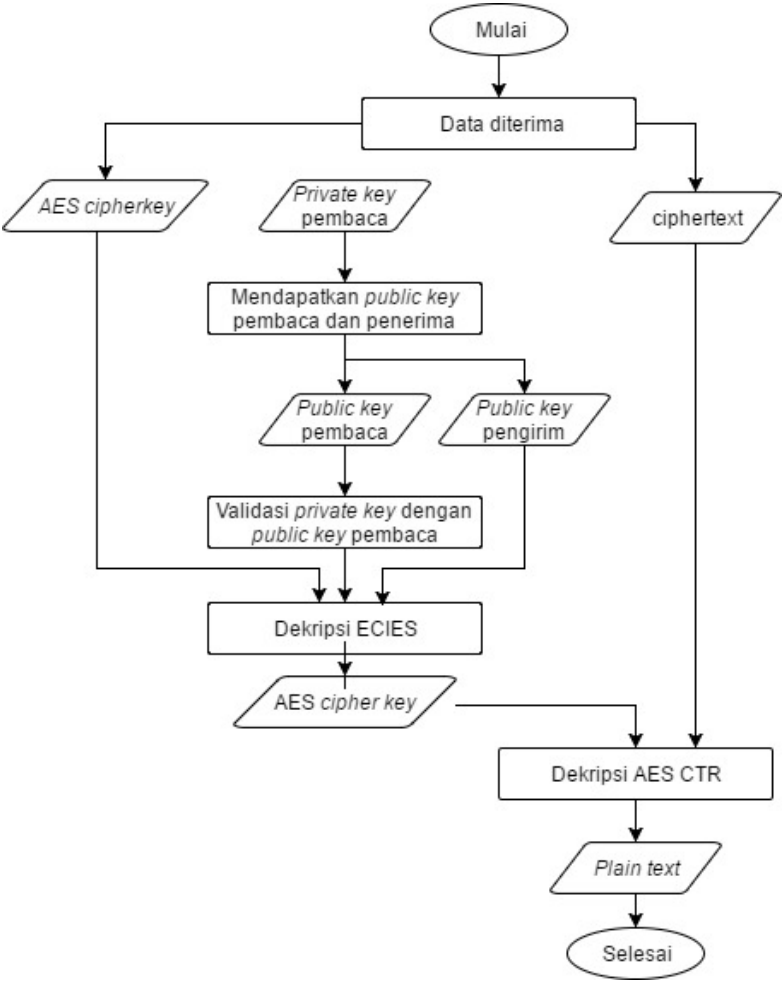
Hasil keluaran dari proses enkripsi algoritma AES adalah gabungan dari:

1. Perkalian *private key* dengan *Generator point* (G)
2. Pesan yang telah dienkripsi menggunakan *symmetric encryption*
3. *Message Authentication Code* (MAC)



Gambar 3.6 Diagram Alir Proses Enkripsi ECIES

3.5.2 Proses Dekripsi Keseluruhan



Gambar 3.7 Diagram Alir Proses Dekripsi

Diagram alir proses dekripsi seccara umum ditunjukkan pada Gambar 3.7. Proses dekripsi dilakukan setelah semua

informasi pada pesan termuat pada halaman website. Diagram Alir Proses Dekripsi secara umum ditunjukkan pada Gambar 3.7. Data yang diterima dari server adalah AES *cipher key* dan pesan dalam bentuk *ciphertext*. Untuk melakukan enkripsi sistem memerlukan *public key* penerima pesan. Sistem mendapatkan *public key* pembaca dan *public key* pengirim menggunakan API yang tersedia. *Public key* pengirim akan digunakan untuk validasi *private key* yang dimasukkan oleh pembaca pesan. Pertama sistem akan melakukan dekripsi pada AES *cipher key* dengan algoritma ECC menggunakan *private key* pembaca dan *public key* pengirim pesan. Hasil dari proses dekripsi pertama ini adalah kunci yang akan digunakan pada proses dekripsi AES.

Proses dekripsi kedua adalah dengan algoritma AES counter mode menggunakan kunci yang telah didekripsi sebelumnnya. Hasil dari proses dekripsi kedua ini adalah pesan asli dalam bentuk *plaintext*.

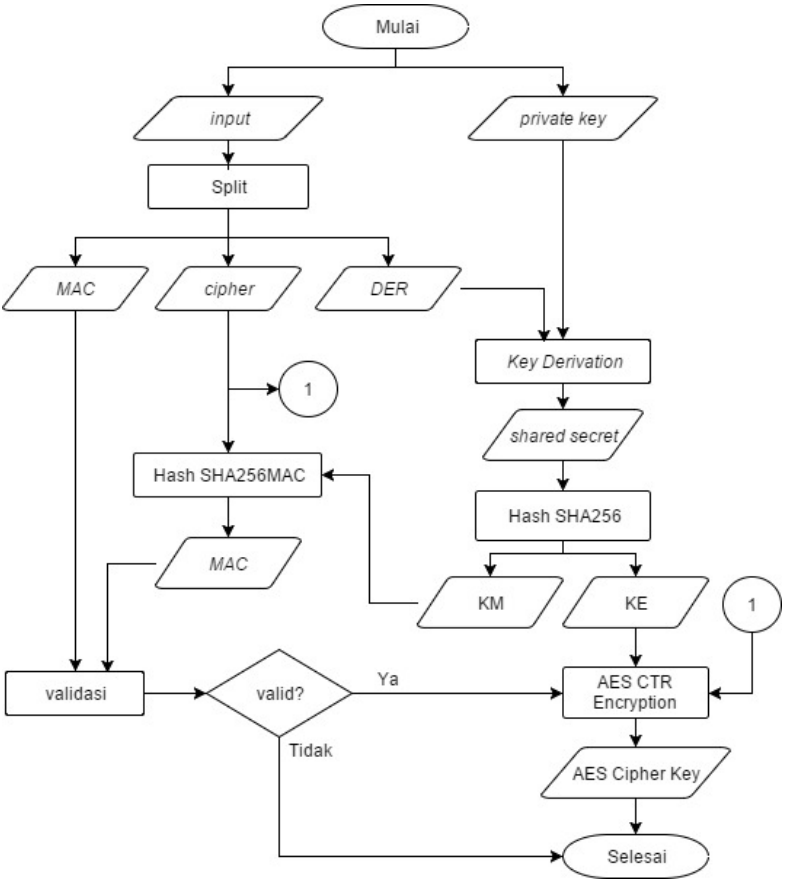
3.5.2.1 Dekripsi ECIES

Karena *keluaran* dari algoritma ECIES adalah gabungan dari *public key* pengirim, pesan dalam buntuk *ciphertext*, dan MAC, untuk proses dekripsi ECIES diperlukan pemisahan atribut-atribut tersebut.

1. Public key pengirim adalah 130 byte pertama.
2. Ciphertext
3. MAC adalah 64 byte terakhir.

Public key pengirim akan digunkan untuk membuat *share secret* menggunakan *private key* pembaca pesan. *Shared secret* kemudian kan dilakukan fungsi hash SHA512 dan dibagi menjadi 2 bagian seperti pada proses enkripsi. Bagian kedua akan digunakan untuk membuat validasi MAC dengan cara digabungkan dengan cipertext dan dilakukan fungsi hash SHA256. Hasil dari proses ini kemudian akan dicocokkan dengan MAC yang terdapat pada pesan. Jika sesuai ciphertext akan dienpkripsis menggunkan *symmetric encryption* dengan bagian pertama dari

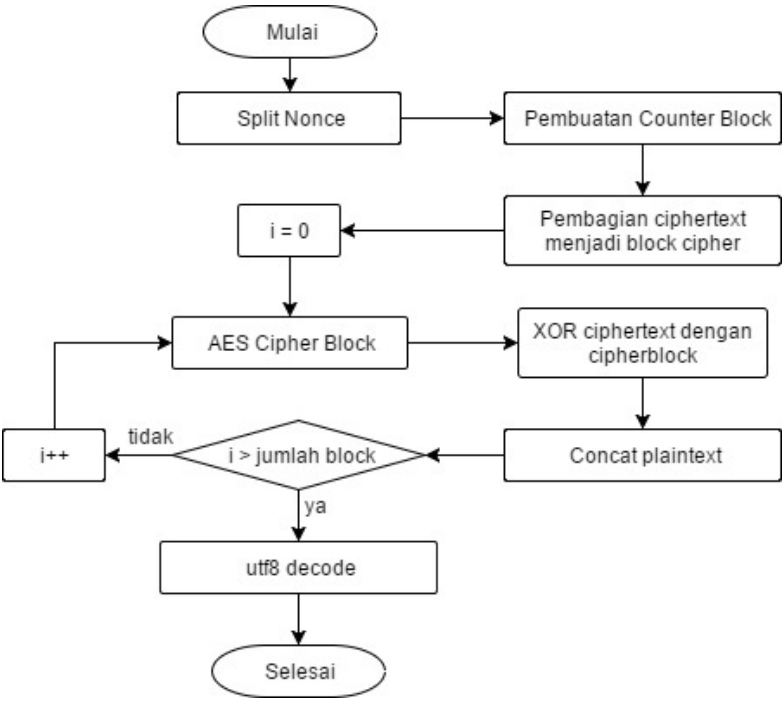
hasil fungsi hash SHA512 *shared secret*. Hasil dari proses dekripsi ECIES adalah AES *key* yang akan digunakan untuk proses enkripsi pesan yang sebernarnya. Diagram alir proses dekripsi ECIES ditunjukkan pada Gambar 3.8.



Gambar 3.8 Diagram Alir Proses Dekripsi ECIES

3.5.2.2 Dekripsi AES Counter Mode

Dekripsi AES CTR memerlukan masukan ciphertext dan key yang didapatkan dari proses dekripsi ECIES. *Key* akan dilakukan proses *key expansion* terlebih dahulu untuk menghasilkan *key schedule*. *Ciphertext* akan dibagi menjadi block berukuran 16 byte. Setiap blok akan didekripsi dengan fungsi AES *cipher block* dengan *key schedule*. Setiap karakter dari hasil AES *cipher block* akan di-XOR-kan dengan ciphertext yang bersesuaian. Hasil dari operasi XOR tersebut akan dikonversikan menjadi karakter ASCII. Diagram alir proses dekripsi AES CTR dapat dilihat pada Gambar 3.9.



Gambar 3.9 Diagram Alir Proses Dekripsi AES CTR

3.6 Perancangan Antarmuka

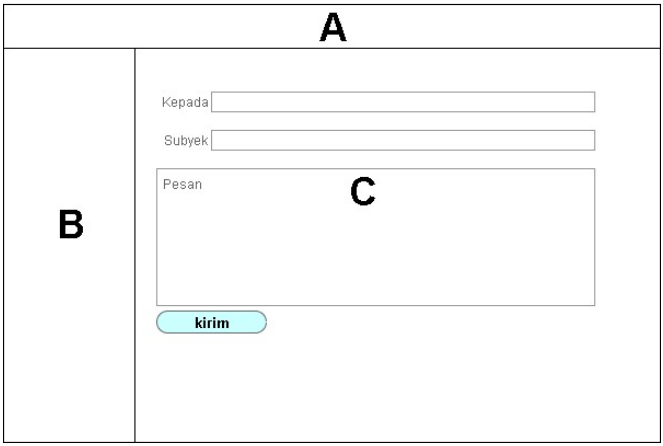
Perancangan antarmuka merupakan tahap yang dilakukan untuk persiapan dalam pembuatan antarmuka website. Antarmuka akan memnggunakan framework bootstrap. Website yang akan dibangun memiliki halaman login, halaman pesan masuk, halaman pesan terkirim, halaman tulis pesan, halaman membaca pesan dan laman profile.

3.6.1 Halaman Menulis Pesan

Halaman ini merupakan halaman saat pengguna akan menuliskan sebuah pesan baru. Pada halaman ini akan ditampilkan *form* pesan yang terdiri dari penerima pesan, subyek pesan, isi pesan dan lampiran. Rancangan antarmuka halaman tulis pesan ditunjukkan pada Gambar 3.10.

Adapun komponen yang terdapat dalam Gambar 3.10 antara lain:

A. Bagian A merupakan *navigation bar*. Pada *Navigation bar* terdapat menu kembali ke halaman utama, *toggle* untuk *side menu* dan perintah *logout.*
B. Bagian B merupakan *side menu*. Pada *side menu* terdapat foto profile dan username pengguna yang sedang aktif, menu untuk mengedit profile, menulis pesan, melihat pesan masuk, dan melihat pesan keluar.
C. Bagian C merupakan *form* untuk menulis pesan baru. *Form* ini berisikan penerima pesan, subyek pesan, isi pesan, lampiran, dan tombol mengirim pesan.



Gambar 3.10 Rancangan Antarmuka Halaman Tulis Pesan

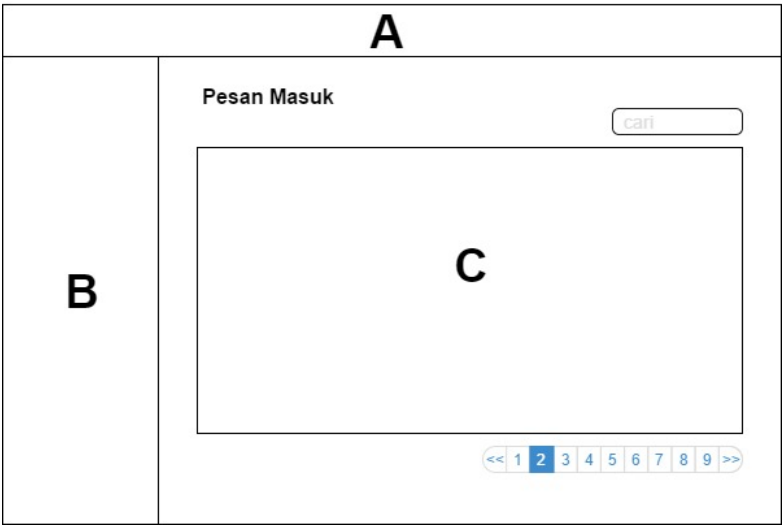
3.6.2 Halaman Pesan Masuk

Halaman ini merupakan halaman pertama yang ditampilkan setelah pengguna *login*. Pada halaman ini akan ditampilkan daftar seluruh pesan masuk. Informasi yang akan ditampilkan adalah tanggal pengiriman pesan, pengrim pesan, dan subyek pesan. Rancangan antarmuka halaman pesan masuk ditunjukkan pada Gambar 3.11.

Adapun komponen yang terdapat dalam Gambar 3.11 antara lain:

A. Bagian A merupakan *navigation bar*. Pada *Navigation bar* terdapat menu kembali ke halaman utama, *toggle* untuk *side menu* dan perintah *logout.*
B. Bagian B merupakan *side menu*. Pada *side menu* terdapat foto profile dan username pengguna yang sedang aktif, menu untuk mengedit profile, menulis pesan, melihat pesan masuk, dan melihat pesan keluar.
C. Bagian C merupakan daftar pesan masuk. Pada daftar ini akan ditampilkan tanggal pengiriman pesan, pengirim

pesan, dan subyek pesan. Pada bagian ini juga terdapat fitur halaman dan pencarian.



Gambar 3.11 Rancangan Antarmuka Halaman Pesan Masuk

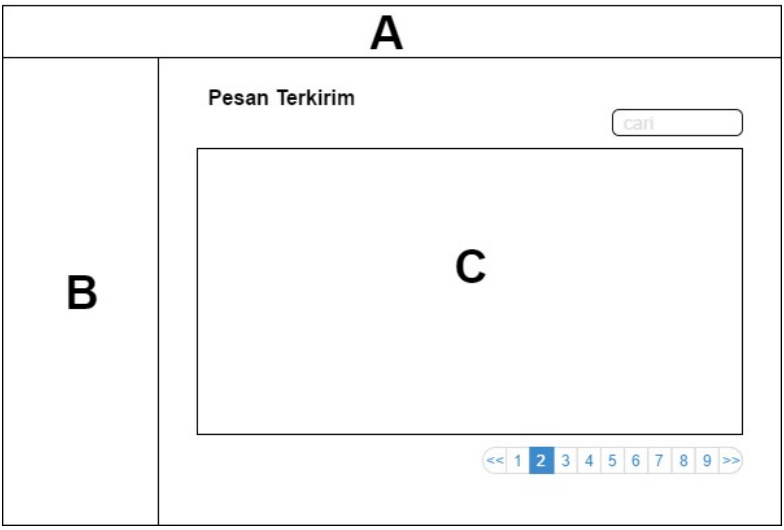
3.6.3 Halaman Pesan Terkirim

Pada halaman ini akan ditampilkan daftar seluruh pesan yang telah terkirim. Informasi yang akan ditampilkan adalah tanggal pengiriman pesan, pengrim pesan, dan subyek pesan. Rancangan antarmuka halaman pesan terkirim ditunjukkan pada Gambar 3.12.

Adapun komponen yang terdapat dalam Gambar 3.12 antara lain:

A. Bagian A merupakan *navigation bar*. Pada *Navigation bar* terdapat menu kembali ke halaman utama, *toggle* untuk *side menu* dan perintah *logout.*
B. Bagian B merupakan *side menu*. Pada *side menu* terdapat foto profile dan username pengguna yang sedang aktif, menu untuk mengedit profile, menulis pesan, melihat pesan masuk, dan melihat pesan keluar.

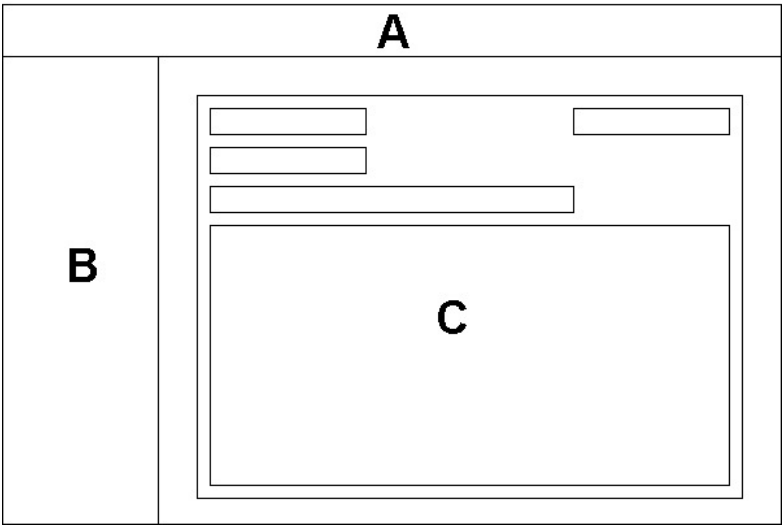
C. Bagian C merupakan daftar pesan masuk. Pada daftar ini akan ditampilkan tanggal pengiriman pesan, pengirim pesan, dan subyek pesan. Pada bagian ini juga terdapat fitur halaman dan pencarian.



Gambar 3.12 Rancangan Antarmuka Halaman Pesan Terkirim

3.6.4 Halaman Membaca Pesan

Halaman ini merupakan halaman yang digunakan pengguna untuk membaca sebuah pesan. Pada halaman ini akan ditampilkan pesan dalam *plaintext*. Informasi yang akan ditampilkan adalah pengirim pesan, penerima pesan, tanggal pengiriman pesan, subyek pesan, isi pesan dan lampiran. Rancangan antarmuka halaman baca pesan ditunjukkan pada Gambar 3.13.



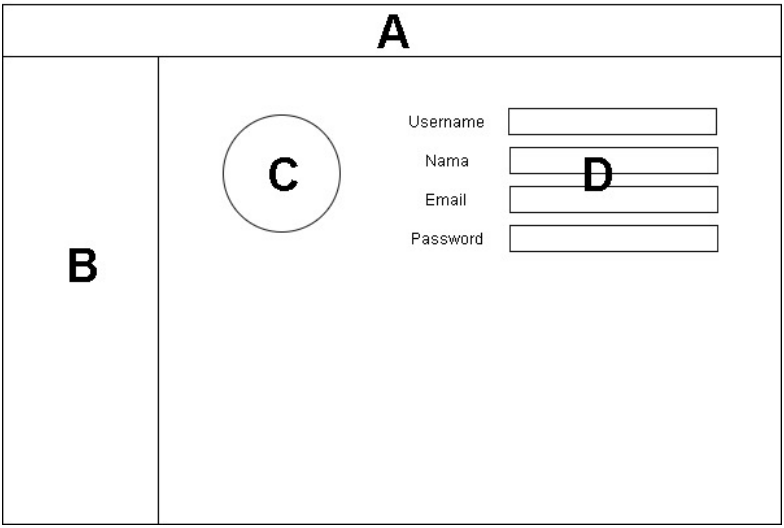
Gambar 3.13 Rancangan Antarmuka Halamn Membaca Pesan

Adapun komponen yang terdapat dalam Gambar 3.13 antara lain:

A. Bagian A merupakan *navigation bar*. Pada *Navigation bar* terdapat menu kembali ke halaman utama, *toggle* untuk *side menu* dan perintah *logout.*
B. Bagian B merupakan *side menu*. Pada *side menu* terdapat foto profile dan username pengguna yang sedang aktif, menu untuk mengedit profile, menulis pesan, melihat pesan masuk, dan melihat pesan keluar.
C. Bagian C merupakan tampilan isi dari pesan. Pada bagian ini akan ditampilkan pengirim pesan, penerima pesan, tanggal pengiriman, subyek pesan, isis pesan, dan lampiran.

3.6.5 Halaman Profile

Halaman ini merupakan halaman yang digunakan pengguna untuk melihat profile secara detail. Informasi yang akan ditampilkan adalah foto profile, username, nama lengkap, email. Pada halaman ini pengguna juga dapat melakukan perubahan pada data profile yang sudah ada. Rancangan antarmuka halaman pesan masuk ditunjukkan pada Gambar 3.14.



Gambar 3.14 Rancangan Antarmuka Halaman Profile

Adapun komponen yang terdapat dalam Gambar 3.14 antara lain:

A. Bagian A merupakan *navigation bar*. Pada *Navigation bar* terdapat menu kembali ke halaman utama, *toggle* untuk *side menu* dan perintah *logout.*
B. Bagian B merupakan *side menu*. Pada *side menu* terdapat foto profile dan username pengguna yang sedang aktif, menu untuk mengedit profile, menulis pesan, melihat pesan masuk, dan melihat pesan keluar.

C. Bagian C merupakan foto profile pengguna. Pada bagian ini juga dugunakan unutk mengubah foto profile pengguna.
D. Badian D merupakan detail dari data pengguna. Pada bagian ini akan ditampilkan username, nama, email dan password. Bagian ini juga digunakan unutk mengubah data pengguna.

3.6.6 Halaman Login

Halaman ini merupakan halaman pertama yang ditampilkan saat pengguna membuka website. Pada halaman ini akan ditampilkan *form* login yang berisi username dan password. Pengguna harus memasukkan username dan password yang sesuai untuk bisa masuk keddalah sistem. Rancangan antarmuka halaman pesan masuk ditunjukkan pada Gambar 3.15.

Gambar 3.15 Rancangan Antarmuka Halaman Login

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa *Pseudocode* untuk membangun program.

4.1 Lingkungan Implementasi

Implementasi perangkat lunak menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Inplementasi Perangkat Lunak

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-6500U @ 2.5 GHz (4 CPUs)
	Memori	8 GB 1600 MHz DDR3L
Perangkat Lunak	Sistem Operasi	Windows 10 64 bit
	Perangkat Pengembang	Intellij IDEA 2016.4.6
	Manajemen Basis Data	SQLyog Ultimate
	Bahasa Pemrograman	Javascript
	Web Server	NodeJS

4.2 Implementasi Proses Enkripsi

Pada sub bab implementasi ini menjelaskan mengenai implementasi penggunaan metode yang dipakai perangkat lunak secara mendetail dan menampilkan *Pseudocode* setiap proses. Pada tugas akhir ini, implementasi menggunkan bahasa pemrograman *javascript*.

4.2.1 Pembuatan *Public Key*

Pembuatan public key dilakukan saat pengguna pertama kali mendaftar pada sistem. Pada perangkat lunak ini, kurva yang digunakan adalah secp256k1. Masukan yang dibutuhkan adalah *private key* yang dimasukkan oleh pengguna. *Private key* dilakukan fungsi hash SHA512 yang kemudian akan digunakan untuk menentukan titik dari *public key*. Implementasi fungsi pembuatan *public key* terdapat pada Gambar 4.1.

Set Key Pair	
Masukan: *privateKey, publicKey*	
1	privateKey <- new Buffer(privateKey);
2	privateKey <- Hash.sha256(privateKey).slice(0,32);
3	privateKey <- new PrivateKey(privateKey);
4	$.checkArgument(privateKey, 'no private key provided')
5	this.privateKey <- privateKey
6	if (publicKey)
7	$.checkArgument(publicKey, 'no public key provided')
8	this.publicKey <- publicKey
9	else
10	this.publicKey <- null
11	return this
Keluaran: -	

Gambar 4.1 Fungsi Pembuatan *Public Key*

4.2.2 Implementasi Algoritma AES Counter Mode

Fungsi ini digunakan untuk enkripsi menggunakan algoritma AES dengan mode *counter*.

4.2.2.1 AES Cipher Block

Fungsi AES Cipher Block digunakan untuk enkripsi tiap block. i*nput* akan dikonversikan menjadi *array* 2 dimensi dan disimpan dalam variable *state*. *State* akan dilakukan iterasi sebanyak jumlah round - 1. Sebelum memasuki iterasi pertama, State akan dilakukan fungsi *addRoundKey*.

Pada iterasi permata akan dilakukan fungsi *subBytes*, *shiftRows*, *mixColumns* dan *addRoundKey* secara berurutan. Pada iterasi terakhir akan dilakukan *subBytes*, *shiftRows*, dan *addRoundKey* secara berurutan. Setalah semua iterasi selesai dilakukan, *state* akan dikembalikan menjadi *array* 1 dimensi. Ouput dari fungsi ini adalah *state* dalam bentuk array 1 dimensi. Implementasi fungsi AES Cipher Block ditunjukkan pada Gambar 2.1.

Fungsi AES Cipher Block	
Masukan: input, key	
1	`Nblock <- 4`
2	`Nround <- key.lenght/Nblock - 1`
3	`for (i=0; i<4*Nblock; i++)`
4	`  state[i%4][Math.floor(i/4)] <- input[i]`
5	`state <- Aes.addRoundKey(state, key, 0, Nblock)`
6	`for (round = 1; round<Nround; round++)`
7	`  state <- Aes.subBytes(state, Nblock)`
8	`  state <- Aes.shiftRows(state, Nblock)`
9	`  state <- Aes.mixColumns(state, Nblock)`
10	`  state <- Aes.addRoundKey(state, key, round, Nblock)`
11	`state <- Aes.subBytes(state, Nblock)`

12	`state <- Aes.shiftRows(state, Nblock)`
13	`state <- Aes.addRoundKey(state, key, Nround, Nblock)`
14	`for (i=0; i<4*Nblock; i++)`
15	`  output[i] <- state[i%4][Math.floor(i/4)]`
16	`return output`
`Keluaran: output`	

Gambar 4.2 Fungsi AES Cipher Block

4.2.2.2 Key Expansion

Key expansion dilakukan pada key sebelum digunakan pada fungsi *AES Cipher Block*. Key expansion akan menggunakan AES *sBox* dan AES *rCon*. *Key* yang digunakan pada enkripsi AES adalah karakter acak sepanjang 32 byte yang selalu berbeda setiap kali fungsi enkripsi akan dilakukan. *Key* awal akan dikembangkan menjadi array 2 dimensi berukuran 60x4. Untuk 7 index pertama, akan di-set sesuai rumus yang bidasarkan nomor index.

Setiap index kellipatan 8, akan dilakukan transformasi khusus. Key pada index tersebut akan dilakukan fungsi *rotWord* yaitu rotasi 4 *byte word* ke kiri sebanyak 1 byte. Kemudian dilakukan fungsi *subWord* yaitu penerapan Sbox. Fungsi *subWord* juga akan dilakukan setiap index ke 4. Kemudian dilakukan operasi xor dengan index sebelumnya. Keluaran dari fungsi ini adalah *keySchedule*. Implementasi fungsi AES key expansion ditunjukkan pada Gambar 4.3

`Fungsi Aes key expansion`	
`Masukan: key`	
1	`Nblock <- 4`
2	`keyLenght <- key.length/4`
3	`Nround <- keyLenght + 6`
4	`for (i=0; i<keyLenght; i++)`
5	`  newKey[i] <- [key[4*i],key[4*i+1],key[4*2],key[4*i+3]]`

```
    for (i = keyLenght; i<(Nblock*(Nround+1));
6   i++)
7     for (j=0; j<4; j++)
8       temp[j] <- newKey[i-1][j]
9     if (i% keyLenght == 0)
10      temp <- Aes.subWord(Aes.rotWord(temp))
11      for (j=0; j<4; j++)
12        temp[j] ^= Aes.rCon[i/keyLenght][j]
13    if (keyLenght>6 && i%keyLenght == 4)
14      temp <- Aes.subWord(temp)
15    for (j=0; j<4; j++)
      newKey[i][j] <- newKey[i-keyLenght][j] ^
16  temp[j]
17  return newKey
Keluaran: newKey
```

Gambar 4.3 Fungsi AES *Key Expansion*

4.2.2.3 Enkripsi Counter Mode

Pada awal fungsi enkripsi *counter mode* akan diinnisialisasi ukuran block (*blockSize)* dan (jumlah bit) *nBits*. Untuk AES ukuran *blockSize* adalah tetap yaitu 16 dan *nBits* yang digunakan adalah 256. Kunci yang digunakan sebagai masukan akan dikonversikan menjasi code ASCII terlebih dahulu. Berikutnya adalah pembuatan nonce sepanjang 8 byte yang dibuat berdasarkan waktu saat fungsi ini dipanggil. 2 byte pertama adalah milidetik, 2 byte berikutnya adalah angka acak, dan 4 byte terakhir adalah detik. Nonce ini akan digunakan sebagai 8 index pertama *counterBlock*. 8 index pertama ciphertext dibuat dari 8 *counterBlock* pertama yang telah dikonversikan mejadi karakter.

Sebelum iterasi counter dilakukan, *key* akan dikembangkan menggunakan fungsi *key expansion*. Kemudian plaintext dibagi sepanjang 16 byte setiap bagian untuk dilakukan fungsi AES *cipher block* degan *keySchedule*

Implemantasi fungsi enkripsi AES CTR ditunjukkan pada Gambar 4.4.

	Fungsi Enkripsi AES Counter mode
	Masukan: Plaintext, key
1	blockSize <- 16
2	nBits <- 256
3	pwBytes <- password.charCodeAt(key)
4	nonce <- (new Date()).getTime()
5	nonceMs <- nonce mod 1000
6	nonceSec <- Math.floor(nonce/1000)
7	nonceRnd <- Math.floor(Math.random()*0xffff)
8	for (i=0; i<2; i++)
9	counterBlock[i] <- (nonceMs >>> i*8) & 0xff
10	for (i=0; i<2; i++)
11	counterBlock[i+2] <- (nonceRnd >>> i*8) & 0xff
12	for (i=0; i<4; i++)
13	counterBlock[i+4] <- (nonceSec >>> i*8) & 0xff
14	for (var i=0; i<8; i++)
15	ctrTxt += String.fromCharCode(counterBlock[i])
16	keySchedule <- Aes.keyExpansion(pwBytes)
17	blockCount <- Math.ceil(plaintext.length/blockSize)
18	for (b=0; b<blockCount; b++)
19	for (var c=0; c<4; c++)
20	counterBlock[15-c] <- (b >>> c*8) & 0xff
21	for (var c=0; c<4; c++)
22	counterBlock[15-c-4]<- (b/0x100000000 >>> c*8)

23	cipherCntr <- Aes.cipher(counterBlock,keySchedule)
24	blockLenght <- b<blockCount-1 ? blockSize : (plaintext.length-1)%blockSize+1
25	For (var i=0; i<blockLength; i++)
26	cipherChar[i] <- cipherCntr[i] ^ plaintext.charCodeAt(b*blockSize+i)
27	cipherChar[i] <- String.fromCharCode(cipherChar[i])
28	ciphertext += cipherChar.join('')
29	if (typeof WorkerGlobalScope != 'undefined' && self instanceof WorkerGlobalScope)
30	if (b%1000 == 0) self.postMessage({ progress: b/blockCount })
31	ciphertext = Ctr.base64Encode(ctrTxt+ciphertext)
32	return ciphertext;
Keluaran: ciphertext	

Gambar 4.4 Fungsi Enkripsi AES *Counter Mode*

4.2.2.4 Dekripsi Counter Mode

Masukan untuk proses dekripsi AES *counter mode* adalh *ciphertext* dan *key*. Kunci yang digunakan sebagai masukan akan dikonversikan menjasi code ASCII terlebih dahulu. Implementasi fungsi dekripsi AES *Counter mode* ditunjukkan pada Gambar 4.5

Dekripsi AES	
Masukan: cipherText, key	
1	blockSize = 16
2	nBits = 256
3	nBytes = nBits/8
4	pwBytes <- key.charCodeAt()
5	ctrTxt = ciphertext.slice(0, 8)
6	for (i=0; i<8; i++)

```
 7    counterBlock[i] = ctrTxt.charCodeAt(i)
 8  keySchedule = Aes.keyExpansion(key)
 9  nBlocks = Math.ceil((ciphertext.length-8) /
     blockSize)
10  for (b=0; b<nBlocks; b++)
11    ct[b] = ciphertext.slice(8+b*blockSize,
     8+b*blockSize+blockSize)
12  ciphertext = ct
13  for (b=0; b<nBlocks; b++)
14    for (c=0; c<4; c++)
15      counterBlock[15-c] = ((b) >>> c*8) &
     0xff
16    for (c=0; c<4; c++)
17      counterBlock[15-c-4] =
     (((b+1)/0x100000000-1) >>> c*8) & 0xff
18    cipherCntr = Aes.cipher(counterBlock,
     keySchedule)
19    for (i=0; i<ciphertext[b].length; i++)
20      plaintxtByte[i] = cipherCntr[i] ^
     ciphertext[b].charCodeAt(i)
21      plaintxtByte[i] =
     String.fromCharCode(plaintxtByte[i])
22    plaintext += plaintxtByte.join('')
23    if (typeof WorkerGlobalScope !=
     'undefined' && self instanceof
     WorkerGlobalScope)
24      if (b%1000 == 0) self.postMessage({
     progress: b/nBlocks })
25  plaintext = Ctr.utf8Decode(plaintext)
26  return plaintext
Keluaran: plaintext
```

Gambar 4.5 Fungsi Dekripsi AES *Counter Mode*

4.2.3 Implementasi Algoritma ECIES

Pada subbab ini akan dijelaskan implementasi algoritma ECIES. Fungsi-fungsi yang akan dijelaskan yaitu inisiali kelas ECIES, fungsi *setKeyPair*, fungsi *setProperty*, fungsi enkripsi, dan fungsi dekripsi.

4.2.3.1 Inisialisasi Kelas ECIES

Pada kelas ECIES akan dibutuhkan fungsi PublicKey, Hash, precondition, dan AES CTR. Implementasi inisialisasi kelas ECIES ditunjukkan pada Gambar 4.6.

Inisialisasi Kelas ECIES	
Masukan: -	
1	`bitcore <- require('bitcore-lib');`
2	`PublicKey <- bitcore.PublicKey;`
3	`var Hash <- bitcore.crypto.Hash;`
4	`$ <- bitcore.util.preconditions;`
5	`ctr <- require('./ctr');`
6	`var ECIES <- function ECIES()`
7	`  if (!(this instanceof ECIES))`
8	`    return new ECIES()`
Keluaran: -	

Gambar 4.6 Inisialisasi Kelas ECIES

4.2.3.2 Fungsi setKeyPair

Fungsi setKeyPair digunakan untuk menentukan pasangan *private key* dan *public key*. Masukan yang diperlukan adalah *private key* (*string hex*) dan *public key* (*string hex*). Fungsi setKeyPair ini akan mengembalikan kelas ECIES. Implementasi fungsi *setKeyPair* ditunjukkan pada Gambar 4.7.

Set Key Pair
Masukan: *privateKey, publicKey*

1	$.checkArgument(privateKey, 'no private key provided')
2	this.privateKey <- privateKey
3	if (publicKey)
4	$.checkArgument(publicKey, 'no public key provided')
5	this.publicKey <- publicKey
6	else
7	this.publicKey <- null
8	return this
Keluaran: -	

Gambar 4.7 Fungsi *setKeyPair* ECIES

4.2.3.3 Fungsi setProperty

Fungsi *setProperty* digunakan untuk menetapkan attribute kelas ECIES yang akan digunakan pada fungsi enkripsi dan dekripsi. Implementasi fungsi *setProperty* ditunjukkan pada Gambar 4.8. *Attribute* kelas ECIES yang ditetapkan pada sungsi ini adalah DER public key dan hash dari *shared secret*.

DER adalah hasil perkalian *private key* dengan *Generator point* pada kurva. Kemudian mendapatkan titik dari hasil perkalian *private key* dengan titik dari *public key* dan *Generator Point*. *Shares secret* adalah nilai X dari titik tersebut. Shared secret kemudian akan dilakukan fungsi hash SHA512 dan menghasilkan buffer dengan 64. Hasil dari fungsi hash akan dibagi menjadi 2 bagian masing-masing sepanjang 32. Bagian pertama adalah *KE* yang akan digunakan sebagai *key* untuk proses *symmetric encryption*. Bagian kedua adalah *KM* yang digunakan unutk pembuatan *Message Authentication Code* (MAC).

Set Property	
Masukan: -	
1	this.R <- this.privateKey.publicKey.toDER(true)

2	`r <- this.privateKey.bn`
3	`KB <- this.publicKey.point`
4	`P <- KB.mul(r)`
5	`S <- P.getX()`
6	`S <- S.toBuffer({size: 32})`
7	`KEKM <- Hash.sha512(S)`
8	`this.KE <- KEKM.slice(0, 32)`
9	`this.KM <- KEKM.slice(32,64)`
Keluaran: -	

Gambar 4.8 Fungsi setProperty ECIES

4.2.3.4 Fungsi Enkripsi

Implementasi fungsi fungsi enkripsi ECIES ditunjukkan pada Gambar 4.9. Untuk proses enkripsi harus inisialisasi kelas ECIES dan menetapkan pasangan *private key* pengirim dan *public key* penerima. Masukan dari proses enkripsi adalah *key* yang digunakan pada enkripsi AES CTR sepanjang 32 *byte*. Hasil dari proses enkripsi adalah gabungan dari *public key* pengirim (*string hex*), *cipher* (*string*), dan *Message Authentication Code* (MAC) (*string hex*).

Enkripsi	
Masukan: *plaintext*	
1	`this.setProperty()`
2	`cipher <- ctr.encrypt(plaintext, this.KE.toString('hex').slice(0,32))`
3	`cipherBuf <- new Buffer(cipher)`
4	`MAC <- Hash.sha256hmac(cipherBuf, this.KM)`
5	`ciphertext <- this.R.toString('hex') + cipher + MAC.toString('hex')`
6	`return ciphertext`
Keluaran: *ciphertext*	

Gambar 4.9 Fungsi Enkripsi ECIES

4.2.3.5 Fungsi Dekripsi

Implementasi fungsi dekripsi ECIES ditunjukkan pada Gambar 4.10. Fungsi dekripsi Dapat dilakukan setelah inisialisasi kelas ECIES seperti pada proses enkripsi. Namun pada proses dekripsi tidak memerlukan public key saat inisialisasi awal. Public key dari proses dekripsi didapat dari atribut yang terdapat pada pesan. Pada proses dekripsi harus dilakukan pemisahan atribut dalam pesan, yaitu *publicKey*, pesan, dan *Message Authentication Code* (MAC).

- *publicKey* diambil dari 66 karakter pertama.
- Pesan diambil dari karakter ke 65 sampai ke panjang ciphertext-64
- MAC diambil dari 64 karakter terakhir

publicKey akan digunakan untuk membuat *sharedSecret* yang kemudian digunakan untuk autentitikasi pesan. Jika nilai dari *sharedSecret* sama dengan MAC, pesan akan didekripsi menggunakan AES CTR. Hasil keluaran dari fungsi ini dalah AES *cipher key* yang akan digunakan untuk enkripsi pesan menggunakan algoritma AES CTR.

Dekripsi	
Masukan: *ciphertext*	
1	`publicKey <- ciphertext.slice(0, 66)`
2	`publicKey <- new Buffer (publicKey, 'hex')`
3	`this.publicKey <- PublicKey.fromDER(publicKey)`
4	`setProperty <- this.setProperty()`
5	`cipher <- ciphertext.slice(66, ciphertext.length - 64)`
6	`cipherBuf <- new Buffer(cipher)`
7	`MAC <- ciphertext.slice(ciphertext.length - 64, ciphertext.length)`
8	`checkMAC <- Hash.sha256hmac(cipherBuf, this.KM)`

```
 9  if (checkMAC.toString('hex') == MAC)
10    return ctr.decrypt(cipher,
   this.KE.toString('hex').slice(0,32))
11  else
12    return ("Invalid Input")
Keluaran: plaintext
```

Gambar 4.10 Fungsi Dekripsi ECIES

4.2.4 Pengiriman Pesan

Data pesan yang dapat dikirimkan dapat berupa text dan file. Keduanya akan dienkripsi terlebih dahulu sebelum dikirimkan pada server. Enkripsi ini akan menggunakan algotirma AES CTR menggunakan kunci yang sama.

Untuk proses enkripsi memerlukan *public key* penerima dan pengirim pesan yang telah disediakan oleh REST API. Selelah mendapatkan *public key* penerima dan pengirim, sistem akan memvalidasi *private key* yang dimasukkan oleh pengguna terlebih dahulu. Jika *private key* valid, proses enkripsi baru akan dilakukan. Pesan akan dienkripsi menggunakan aloritma AES CTR menggunakan kunci yang dibangkitkan secara acak. Kemudian kunci tersebut dan dienkripsi menggunakan algoritma ECIES sebanyak dua kali. Proses enkripsi pertama menggunakan *private key* pengirim dan *public key* penerima yang akan digunakan penerima untuk membaca pesan. Enkripsi kedua menggunakan *private key* pengirim dan *public key* pengirim yang akan digunakan oleh pengirim untuk membaca pesan yang telah dikirimkan. Implementasi fungsi pengiriman pesan dapat dilihat pada Gambar 4.11.

```
Fungsi Pengiriman pesan
Masukan: text, key, privateKey
1  data <- HTTP GET request(pbkey)
2  publickeyPengirim <- data[0].pbkey
3  publickeyPenerima <- data[1].pbkey
```

4	if (cekPrivateKey(privateKey, publickeyPengirim))
5	senderPrivate <- privateKey
6	if(text)
7	ciphertext = Ctr.encrypt(text, key)
8	recieverKey = crypt(key, senderPrivate, publickeyPenerima)
9	senderKey = crypt(key, senderPrivate, publickeyPengirim)
Keluaran: ciphertext, recieverKey, senderKey	

Gambar 4.11 Fungsi pengiriman pesan

Untuk pengiriman lampiran yang berupa *file*, *file* lampiran akan terlebih dahulu dibaca sebagai data URL atau base64 menggunakan *library FileReader*. Kemudian file akan dienkripsi menggunakan kunci yang sama yang digunakan pada enkripsi pesan text. File yang telah dienkripsi akan dikirimkan bersamaan dengan pesan text dan disimpan dengan nama file dan ekstensi yang sama dengan file asli pada server. Implementasi enkripsi file terdapat pada Gambar 4.12.

Fungsi Enkripsi File Lampiran	
Masukan: file, key	
1	reader <- new FileReader()
2	reader.onload <- function(e) {
3	result <- reader.result;
4	cipherFile <- Ctr.encrypt(result, key);
5	};
6	reader.readAsDataURL(file);
Keluaran: cipherFile	

Gambar 4.12 Fungsi Enkripsi File Lampiran

4.2.5 Membaca Pesan

Pada perangkat lunak ini, pesan hanya bisa dibaca oleh penerima yang valid dan pengirim pesan. Pesan yang diterima olek pengguna adalah berupa *ciphertext* yang akan didekripsi setalah diterima pada *browser*. Browser akan melakukan HTTP GET request untuk mendapatkan public key pengirim pesan. Implementasi proses dekripsi pesan dapat dilihat pada Gambar 4.13.

Untuk membaca pesan, pertama akan dilakukan dekripsi *cipherKey* yang sesaui. Untuk penerima pesan akan menggunakan *recieverKey* seperti yang ditunjuukan pada baris 8. Jika pembaca adalah pengirim pesan itu sendiri akan menggunakan *senderKey* yang ditunjukkan pada baris 5. Proses dekripsi cipher key ini menggunakan algoritma ECIES. Setelah mendapatkan kunci, proses selanjutnya adalah dekripsi pesan menggunakan kunci tersebut menggunakan algoritma AES CTR. *Plaintext* hasil dari proses dekripsi AES inlah yang akan ditampikan pada *browser*.

Fungsi membaca pesan	
Masukan: text, privatekey	
1	data <- HTTP GET request(pbkey)
2	publickeyPengirim = data[0].pbkey
3	publickeyPenerima = data[1].pbkey
4	if (activeUser == pengirim)
5	cipherkey = $("#senderkey").text()
6	publickey = publickeyPengirim
7	else
8	cipherkey <- $("#receiverkey").text()
9	publickey <- publickeyPenerima
10	ENDIF
11	if (cekPrivateKey(privatekey, publickey))
12	key <- decrypt(cipherkey, privatekey, publickeyPengirim)

13	`plaintext = Ctr.decrypt(text, key)`
14	`return plaintext`
`Keluaran: plaintext`	

Gambar 4.13 Funsi Membaca Pesan

Jika terdapat file lampiran pada pesan, untuk membacanya, pertama browser akan melakukan HHTP GET rewuqst untuk mendapatkan file lampiran tersebut karena file lampiran tidak dikirimkan bersamaan dengan pesan teks. File lampiran kemudian akan didekripsi menggunakan kunci yang telah didekripsi menggunakan algoritma ECIES pada saat membuka pesan teks. Implementasi untuk mengunduh lampiran terdapat pada Gambar 4.14.

`decrypt file`	
`Masukan: privateKey, key`	
1	`file <- HTTP GET request(Lampiran)`
2	`fileLampiran = Ctr.decrypt(file, key)`
3	`download(fileLampiran)`
`Keluaran: -`	

Gambar 4.14 Fungsi Unduh Lampiran

4.3 Implementasi Basis Data

Pada subbab ini akan dijelaskan mengenai implementasi setiap tabel. Berdasarkan perancangan basis data, terdapat 2 tabel yang diimplementasikan pada perangkat lunak. Implementasi basis data menggunakan MySQL.

4.3.1 Tabel Pengguna

Tabel pengguna digunakan untuk menyimpan data dari pengguna perangkat lunak. Implementasi pembuatan tabel Pengguna menggunakan MySQL ditunjukkan pada Kode Sumber 4.1

```
CREATE TABLE `pengguna` (
    `username` varchar(30) NOT NULL,
    `password` varchar(32) NOT NULL,
    `nama` varchar(60) NOT NULL,
    `email` varchar(60) DEFAULT NULL,
    `avatar` longtext,
    `pbkey` varchar(255) NOT NULL,
    PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Kode Sumber 4.1 Query untuk membuat Tabel Pengguna

4.3.2 Tabel Pesan

Tabel Pesan digunakan untuk menyimpan semua data pesan yang dikirimkan oleh pengguna. Implementasi pembuatan tabel Pengguna menggunakan MySQL ditunjukkan pada Kode Sumber 4.2

```
CREATE TABLE `pesan` (
    `id_pesan` varchar(10) NOT NULL,
    `pengirim` varchar(50) NOT NULL,
    `penerima` varchar(50) NOT NULL,
    `subyek` text,
    `pesan` text,
    `receiverkey` text,
    `senderkey` text,
    `waktu` timestamp NOT NULL DEFAULT CURRENT_TIME
STAMP ON UPDATE CURRENT_TIMESTAMP,
    `lampiran` text,
    `isRead` tinyint(1) NOT NULL DEFAULT '0',
    `isDelete` tinyint(1) NOT NULL DEFAULT '0',
    `isSenderDelete` tinyint(1) NOT NULL DEFAULT '0
',
    PRIMARY KEY (`id_pesan`),
    KEY `FK_pesan_pengirim` (`pengirim`),
    CONSTRAINT `FK_pesan_pengirim` FOREIGN KEY (`pe
ngirim`) REFERENCES `pengguna` (`username`) ON DELE
TE NO ACTION ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Kode Sumber 4.2 Query untuk membuat Tabel Pesan

4.4 Implementasi Website

Terdapat beberapa hal yang yang diimplementasikan pada pengembangan website, yaitu implementasi teknologi, implementasi fungsionalitas, dan implementasi antarmuka. Pada subbab berikutnya akan dijelaskan ketiga implementasi tersebut.

4.4.1 Implementasi Teknologi

Pada Implementasi penggunaan teknologi akan dibahas mengenai dependency yang diperlukan oleh perangkat lunak, pengaturan koneksi ke basis data, dan fungsi autentikasi.

4.4.1.1 *Dependency* NodeJS

Teknologi yang dibutuhukan pada aplikasi website yaitu NodeJS, Pada Node JS terdapat file bernama *package.json* yang menyimpan pengaturan awal aplikasi dependensi yang dibutuhkan oleh aplikasi. Isi dari file *package.json* ditunjukkan pada Kode Sumber 4.3.

```
{
  "name": "app",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "bitcore-lib": "^0.14.0",
    "body-parser": "~1.17.1",
    "cookie-parser": "^1.4.3",
    "debug": "~2.6.3",
    "ejs": "~2.5.6",
    "express": "~4.15.2",
    "express-session": "^1.15.2",
    "morgan": "~1.8.1",
    "multer": "^1.3.0",
    "mysql": "^2.13.0",
```

```
    "serve-favicon": "~2.4.2"
  }
}
```

Kode Sumber 4.3 File Package.json

Untuk instalasi setiap dependensi dilakukan menggunakan NPM dengan NodeJS yang sudah terinstal. Untuk melakukan instalasi jalankan command berikut pada terminal:

```
Npm install
```

Kode Sumber 4.4 Perintah Instalasi NodeJS Package

4.4.1.2 Pengaturan Koneksi Basis Data

Basis data menggunakan MySQL

```
1  var mysql = require('mysql');
2
3  var connection = mysql.createConnection({
4      host     : 'localhost',
5      user     : 'root',
6      password : '',
7      database : 'ta'
8  });
9
10 connection.connect();
11
12 module.exports = connection;
```

Kode Sumber 4.5 Implementasi Koneksi ke Basis Data

4.4.1.3 Autentikasi

Autentikasi digunakan untuk memvalidasi pengguna saat masuk ke sistem. Autentikasi akan diterapkan pada seluruh halaman website kecuali halaman login dan register. Implementasi proses autentikasi dapat dilihat pada Kode Sumber 4.6.

```
1  module.exports = function (req, res, next){
2      if (req.session.user)
3          next();
4      else {
```

```
5           res.redirect('/');
6       }
7   };
```

Kode Sumber 4.6 Implementasi Autentikasi

4.4.2 Implementasi Fungsionalitas

Pada subbab ini akan dibahas mengenail implementasi fungsionalitas website sesuai dengan rancangan kasus penggunaan pada bab 3.4.

4.4.2.1 Implementasi Mendaftarkan Pengguna Baru

Pada halaman pendaftaran akan terdapat form yang berisi username, nama, email, password, dan *private key*. Sistem akan melakukan pengecekan username apakah username belum dipaki sebelum data dikirimkan. Jika username tersedia, maka proses akan dilanjutkan ke pembuatan *public key* berdasarkan *private* key yang dimasukkan penggnua. Data yang akan dikirimkan ke server adalah username, nama, email, password, dan *public key* yang telah dibangkitkan menggunakan *private key*. Sistem akan menyimpan data yang dikirimkan tersebut ke basis data. Jika proses ini berhasil, pengguna akan diarahkan ke halaman *login*, jika gagal pengguna akan dikembalikan ke halaman *register*.

```
1   router.get('/', function(req, res, next) {
2       var sess = req.session;
3       if(sess.user) res.redirect('/pesanmasuk');
4       res.render('register', {});
5   });
6
7   router.post('/', function(req, res, next) {
8       var username = req.body.username;
9       var name = req.body.nama;
10      var email = req.body.email;
11      var password = req.body.password;
12      var pbkey = req.body.public;
13
14      var insertdata = {
15          username: username,
16          nama: name,
```

```
17          email: email,
18          pbkey: pbkey
19      };
20
21      var query = connection.query('INSERT INTO
22         pengguna SET ?, password = MD5(?)',
23         [insertdata, password], function (error,
24         results, fields) {
25          if (error) {
26              res.redirect('/register');
27          }
28          else {
29              res.redirect('/');
30          }
31      });
32  });
```

Kode Sumber 4.7 Implementasi Mendaftarkan Pengguna Baru

4.4.2.2 Implementasi Manajemen Profile Pengguna

Pengguna yang sudah terdaftar dapat melihat dan melakukan perubahan pada data diri yang sudah ada. Pengguna juga dapat menambalkan foto profil. Untuk menampilkan detail data pengguna, sistem akan melakukan *query* ke tabel pengguna sesuai dengan *username* yang sedang aktif. Pengguna dapat melakukan perubahan dengan mengisi form yang tersedia dan sistem akan melakukan *query update*.

```
1   Var username = sess.user;
2   var name = req.body.nama;
3   var email = req.body.email;
4   var password = req.body.password;
5   var avatar = req.body.avatar;
6
7   var updatedata = {
8     password: password,
9     nama: name,
10    email: email,
11    avatar: avatar
12  };
13
```

```
14  var query = connection.query('UPDATE pengguna
15  SET ? where username = ?', [updatedata,
16  username], function (error, results, fields) {
17    if (error) {
18      console.log(query.sql);
19    }
20    else {
21      edited = 1;
22      res.redirect('/profile/');
23    }
24  });
```

Kode Sumber 4.8 Implementasi Manajemen Profile Pengguna

4.4.2.3 Implementasi Menulis Pesan Baru

Sebelum pesan yang ditulis oleh pengguna dikirimkan ke server, pesan akan dienkripsi terlebih dahulu seperti yang sudah dijelaskan pada bab 4.2.4. Data yang dirimkan adalah penerima, subyek, pesan yang telah dienkripsi, *senderKey*, *recieverKey*. Dan lampiran

```
1   var penerima = req.body.penerima;
2   var subyek = req.body.subyek;
3   var pesan = req.body.pesan;
4   var receiverkey = req.body.receiverkey;
5   var senderkey = req.body.senderkey;
6   var lampiran = req.body.lampiran;
7   var filePath = '';
8
9   if (lampiran) {
10    filePath = './../public/uploads/' +
11  lampiran.split('.encrypted')[0];
12    var write = fs.writeFile(filePath,
13  lampiran.split('.encrypted')[1], function (err)
14  {
15       if (err) throw err;
16    });
17  }
18
19  var insertdata = {
20    id_pesan: randomString10(),
21    pengirim: sess.user,
22    penerima: penerima,
```

```
23    subyek: subyek,
24    pesan: pesan,
25    receiverkey: receiverkey,
26    senderkey: senderkey,
27    lampiran: filePath.split('./..')[1]
28  };
29  var sql = 'insert into pesan set ?';
30  var sqlcheck = 'select 1 as status from pesan
31  where id_pesan = ?';
32
33  function insertFunction(sqlcheck, checkid) {
34    var query1 = connection.query(sqlcheck,
35  checkid, function (error, result, field) {
36      if (!result[0]) {
37        var query = connection.query(sql,
38  insertdata, function (error, result, field) {
39          if (error) console.log(query.sql);
40            res.redirect('/pesanterkirim');
41          });
42        } else {
43        insertdata.id = randomString10();
44        insertFunction(sqlcheck,
45  insertdata.id_pesan);
46      }
47    });
48    return 0;
49  }
50  var insert = insertFunction(sqlcheck,
51  insertdata.id_pesan);
```

Kode Sumber 4.9 Implementasi Menulis Pesan Baru

4.4.2.4 Implementasi Menampilkan Daftar Pesan Masuk

Pada implementasi kasus penggunaan menampilkan daftar pesan masuk, dilakukan query untuk mendapatkan data pesan yang ditujukan untuk pengguna yang sedang akttif. Informasi yang akan ditampilkan pada *browser* adalah pengirim pesan, tanggal pengiriman pesan, dan subyek.

```
1   var sql = 'SELECT p.* from pesan p where
2   p.penerima = ?';
3   var query = connection.query(sql, [sess.user],
4   function (error, rows, f) {
```

```
5     if (error) throw error;
6       res.render('pesanmasuk', {
7         userID: sess.user,
8         unread: unread,
9         avatar: avatar,
10        d: rows
11      }, function (err, html) {
12        if (err) res.redirect('/');
13          res.send(html);
14      });
15  });
```

Kode Sumber 4.10 Implementasi Menampilkan Daftar Pesan Masuk

4.4.2.5 Implementasi Menampilkan Daftar Pesan Terkirim

Pada implementasi kasus penggunaan menampilkan daftar pesan terkirim, dilakukan query untuk mendapatkan data pesan yang telah dikirimkan oleh pengguna yang sedang aktif. Informasi yang akan ditampilkan pada *browser* adalah penerima pesan, tanggal pengiriman pesan, dan subyek.

```
1   var sql = 'SELECT p.* from pesan p where
2   p.pengirim = ?';
3   var query = connection.query(sql, [sess.user],
4   function (error, rows, f) {
5     if (error) throw error;
6       res.render('pesanmasuk', {
7         userID: sess.user,
8         unread: unread,
9         avatar: avatar,
10        d: rows
11      }, function (err, html) {
12        if (err) res.redirect('/');
13          res.send(html);
14      });
15  });
```

Kode Sumber 4.11 Implementasi Menampilkan Daftar Pesan Terkirim

4.4.2.6 Implementasi Membaca Pesan

Pada implementasi pengiriman pesan, data yang akan dirimkan oleh Sistem adalah data pesan sesuai dengan *id_pesan* yang diplih. Informasi yang akan dikirimkan ke *client* adalah pengirim, penerima, subyek, pesan dalam bentuk *ciphertext*, *senderKey*, *recieverKey*, tanggal pengiriman, dan nama lampiran.

```
1  var sql = 'SELECT * from pesan where id_pesan =
2  ?';
3    connection.query(sql, [pesanID], function
4  (error, rows, f) {
5      if (error) throw error;
6        if (rows[0].penerima == sess.user) {
7          var sql0 = 'UPDATE pesan SET isRead = 1
8  WHERE id_pesan = ?';
9          connection.query(sql0, [pesanID],
10 function (err, rows, f) {
11         });
12       }
13     var sql0 = 'SELECT COUNT(*) JUMLAH FROM
14 pesan WHERE penerima = ? AND isRead = 0';
15       connection.query(sql0, [sess.user],
16 function (err, rows, f) {
17         if (err) res.redirect('/');
18         unread = rows[0].JUMLAH;
19       });
20       var valid;
21       if (rows[0].penerima != sess.user) valid
22 = false;
23       else valid = true;
24
25       res.render('view', {
26         userID: sess.user,
27         unread: unread,
28         avatar: avatar,
29         valid: valid,
30         d: rows[0]
31       }, function (err, html) {
32     if (err) res.redirect('/');
33     res.send(html);
34   });
35 });
```

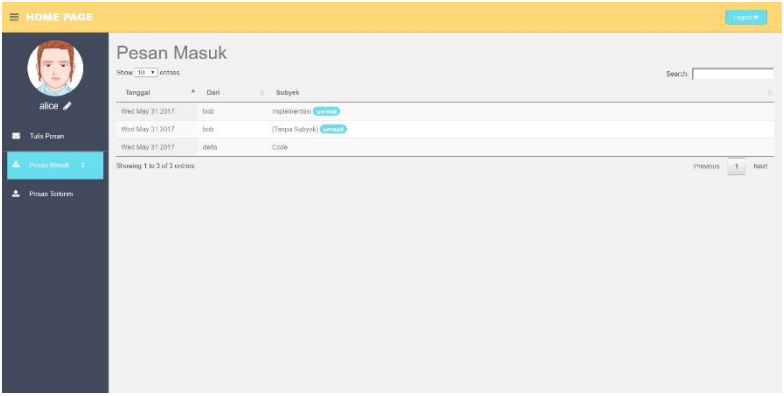
Kode Sumber 4.12 Implementasi Membaca Pesan

4.4.3 Implementasi Antarmuka

Pada implementasi antarmuka akan ditunjukkan tampilan antarmuka yang telah dibuat sesaui dengan perangcangan perangkat lunak.

4.4.3.1 Implementasi Halaman Pesan Masuk

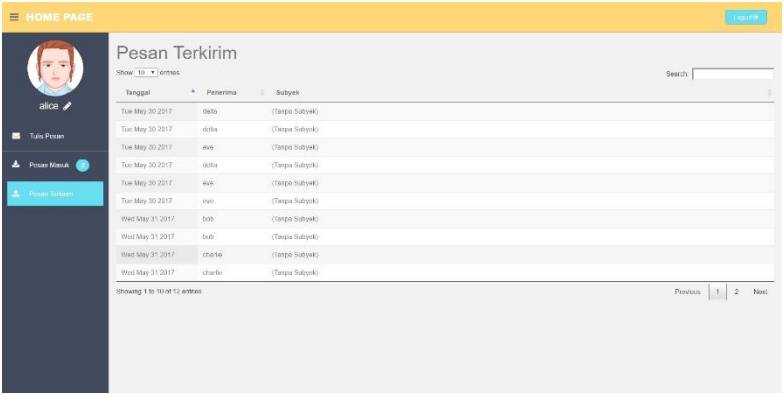
Implamentasi halaman pesan masuk dapat diakses dengan *route /pesanmasuk.* Tampilan antarmuka halaman pesan masuk ditunnjukkan pada Gambar 4.15. Informasi yang ditampilkan adalah daftar pesan yang ditujukan kepada pengguna yang sedang aktif. Setiap pesan akan menampilkan detail waktu pengiriman, pengirim, subyek pesan, dan status pesan belum dibaca.



Gambar 4.15 Antarmuka Halaman Pesan Masuk

4.4.3.2 Implementasi Halaman Pesan Terkirim

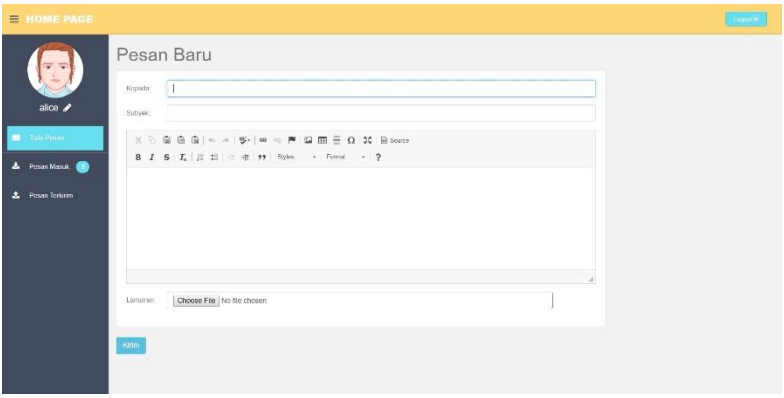
Implamentasi halaman pesan terkirim dapat diakses dengan *route /pesanterkirim.* Tampilan antarmuka halaman pesan terkirim ditunnjukkan pada Gambar 4.16. Informasi yang ditampilkan adalah daftar pesan yang dikirimkan oleh pengguna yang sedang aktif. Setiap pesan akan menampilkan detail waktu pengiriman, penerima, subyek pesan, dan status pesan belum dibaca.



Gambar 4.16 Antarmuka Halaman Pesan Terkirim

4.4.3.3 Implementasi Halaman Menulis Pesan

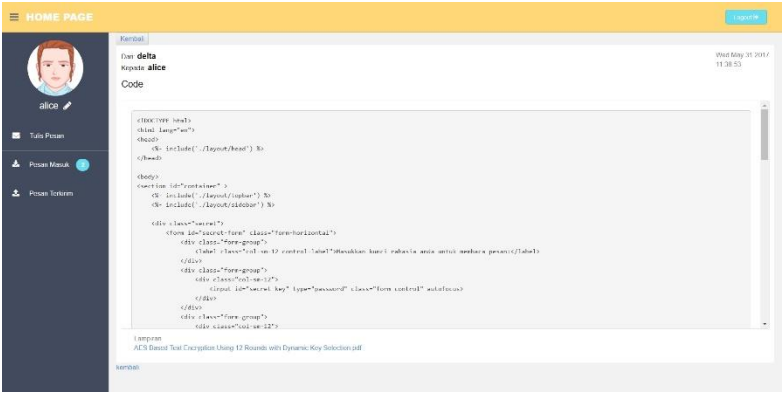
Implamentasi halaman tulis pesan dapat diakses dengan *route /tulispesan.* Tampilan antarmuka halaman menulis pesan ditunnjukkan pada Gambar 4.17. Pada halaman ini akan ditampilakan form pesan baru yang berisi penerima, subyek, isi pesan, dan file lampiran.



Gambar 4.17 Antarmuka Halaman Menulis Pesan

4.4.3.4 Implementasi Halaman Membaca Pesan

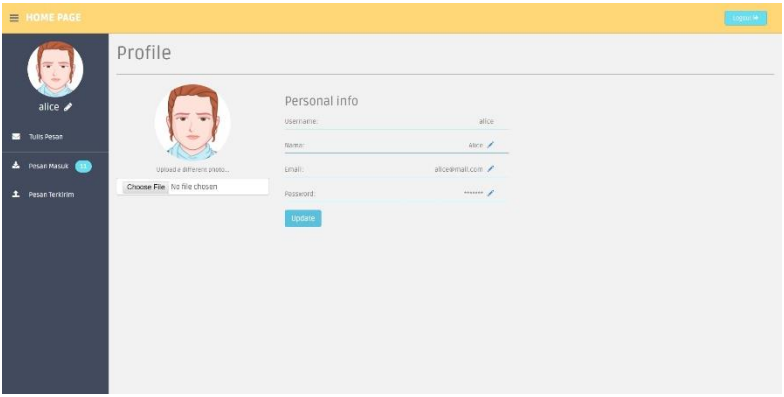
Implamentasi halaman membaca pesan dapat diakses dengan *route /view/IDpesan.* Tampilan antarmuka halaman view ditunnjukkan pada Gambar 4.18. Informasi yang ditampilkan adalah detail waktu pengiriman, pengirim, penerima, subyek pesan, isi pesan dan file lampiran.



Gambar 4.18 Antarmuka Halaman Membaca Pesan

4.4.3.5 Implementasi Halaman Profile

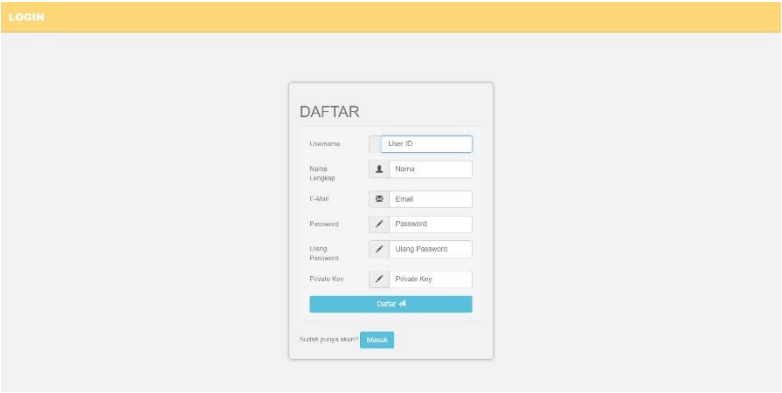
Implamentasi halaman profile dapat diakses dengan *route /profile*. Tampilan antarmuka halaman profile ditunnjukkan pada Gambar 4.19. Halaman ini akan digunakan untuk melihat dan mengubah data pengguna aktif. Informasi yang ditampilkan foto profile, username, nama, email, dan password.



Gambar 4.19 Antarmuka Halaman Profile

4.4.3.6 Implemenasi Halaman Pendaftaran

Implamentasi halaman pendaftaran dapat diakses dengan *route '/register'*. Tampilan antarmuka halaman *register* ditunnjukkan pada Gambar 4.20. Form yang akan ditampilkan adalah username, nama lengkap, email, password, konfirmasi password, dan *private key*.



Gambar 4.20 Antarmuka Halaman Pendaftaran

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi

5.1 Lingkungan Pengujian

Pengujian dilakukan menggunakan 2 perangkat yaitu computer server dan computer client. Spesifikasi perangkat keras dan perangkat lunak pada server ditunjukkan pada Tabel 5.1. Sedangkan spesifikasi perangkat keras dan perangkat lunak pada *client* ditunjukkan pada Tabel 5.2.

Tabel 5.1 Spesifikasi Lingkungan Server

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i3-3240 @ 3.4 GHz (4 CPUs)
	Memori	6 GB
Perangkat Lunak	Sistem Operasi	Windows 10 64 bit
	Manajemen Basis Data	phpmyadmin
	Web Server	NodeJS

Tabel 5.2 Spesifikasi Lingkungan *Client*

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-6500U @ 2.5 GHz (4 CPUs)
	Memori	8 GB 1600 MHz DDR3L
	Sistem Operasi	Windows 10 64 bit

Perangkat Lunak	Peramban web	Google Chrome versi 58.0.3029.110 (64-bit)

5.2 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, perangkat lunak akan diuji apakah sudah berjalan dengan benar. Akan ada dua skenario uji coba yan dilakukan, yaitu uji fungsionalitas dan uji performa.

5.2.1 Skenario Uji Fungsionalitas

Uji coba fungsionalitas dilakukan untuk melihat apakah perangkat lunak dapat melakukan operasi-operasi sesuai dengan kasus kegunaan pada bab 3.3. Selain itu akan terdapat uji coba pada pengiriman pesan. Pada uji coba ini akan dilakukan *sniffing* dan analisa menggunakan wireshark untuk melihat data yang dikirimkan dari *client* kepada server. Detail mengenai setiap scenario akan dijelaskan pada subbab berikutnya.

5.2.1.1 Uji Mendaftarkan Pengguna Baru

Pada uji mendaftarkan pengguna baru, sistem akan diuji fungsinalitas untuk mendaftarkan pengguna baru kedalam sistem. Detail mengenai skenario uji mendaftarkan pengguna baru ditunjukkan pada Tabel 5.3.

Tabel 5.3 Skenario Uji Mendaftarkan Pengguna Baru

ID	UC-01
Nama	Uji Coba Mendaftarkan Pengguna Baru
Tujuan	Menguji fungsionalitas mendaftarkan pengguna baru
Skenario	1. Pengguna membuka halaman *register* 2. Pengguna mengisi form 3. Pengguna me-*submit* form pendaftaran

Hasil harapan	Pengguna baru berhasil disimpan ke dalam sistem

5.2.1.2 Uji Manajemen Profile Pengguna

Uji manajemen profile pegguna didasarkan pada kasus penggunaan pada bab 3.4. Pada fungsionalitas ini sistem diharapkan mampu menampilkan informasi pengguna yang sedang aktif dan dapat melakukan perubahan data pengguna. Detail mengenai skenario uji manajemen profile pengguna ditunjukkan pada Tabel 5.4.

Tabel 5.4 Skenario Uji Manajemen Profile Pengguna

ID	UC-02
Nama	Uji Coba Manajemen Profile Pengguna
Tujuan	Menguji fungsinalitas manajemen profil pengguna
Skenario	1. Pengguna membuka halaman *profile* 2. Pengguna mengubah isian form 3. Pengguna me-*submit* form perubahan
Hasil harapan	1. Pengguna dapat melihat informasi yang telah disimpan. 2. Informasi pengguna yang baru berhasil disimpan ke dalam sistem.

5.2.1.3 Uji Menampilkan Daftar Pesan Masuk

Pada kasus penggunaan menampilkan daftar pesan masuk, sistem diharapkan dapat menampilkan pesan yang ditujukan kepada pengguna yang sedang aktif. Data yang akan ditampilkan adalah tanggal pengiriman pesan, pengirim pesan, dan subyek pesan. Detail mengenai skenario uji manajemen profil pengguna ditunjukkan pada Tabel 5.5.

Tabel 5.5 Skenario Uji Menampilkan Daftar Pesan Masuk

ID	UC-03
Nama	Uji Coba Menampilkan Daftar Pesan Masuk
Tujuan	Menguji fungsionalitas menampilkan daftar pesan masuk
Skenario	1. Pengguna *login* ke sistem 2. Pengguna membuka halaman *pesan masuk*
Hasil harapan	Pengguna dapat melihat daftar pesan masuk

5.2.1.4 Uji Menampilkan Daftar Pesan Terkirim

Pada kasus penggunaan menampilkan daftar pesan masuk, sistem diharapkan dapat menampilkan pesan yang dikirimkan oleh pengguna yang sedang aktif. Data yang akan ditampilkan adalah tanggal pengiriman pesan, pengirim pesan, dan subyek pesan. Detail mengenai skenario uji manajemen profil pengguna ditunjukkan pada Tabel 5.6.

Tabel 5.6 Skenario Uji Menampilkan Daftar Pesan Terkirim

ID	UC-04
Nama	Uji Coba Menampilkan Daftar Pesan Terkirim
Tujuan	Menguji fungsionalitas menampilkan daftar pesan terkirim
Skenario	1. Pengguna *login* ke sistem 2. Pengguna membuka halaman *pesan terkirim*
Hasil harapan	Pengguna dapat melihat daftar pesan terkirim

5.2.1.5 Uji Menulis Pesan Baru

Pada saat pengguna akan mengirimkan sebuah pesan baru, pesan tersebut akan dienkripsi terlebih dahulu pada browser sebelum dikirimkan pada server dan disimpan dalam basis data. Pengujian ini bertujuan untuk menguji fungsionalitas tersebut. Detail skenario uji menulis pesan ditunjukkan pada Tabel 5.7.

Tabel 5.7 Skenario Uji Menulis Pesan Baru

ID	UC-05
Nama	Uji Coba Menulis Pesan Baru
Tujuan	Menguji fungsionalitas menulis pesan baru
Skenario	1. Pengguna membuka halaman *tulis pesan* 2. Pengguna mengisi form pesan 3. Pengguna memasukkan *private key* 4. Pengguna me-*submit* form pesan
Hasil harapan	1. Pesan dienkripsi pada browser 2. Pesan baru disimpan ke dalam sistem

5.2.1.6 Uji Membaca Pesan

Pada saat pengguna membuka halaman *view*, sistem akan mengirimkan pesan dalam bentuk ciphertext. Pesan kemudian akan ditampilkan dalam bentuk plaintext. Skenario uji membaca pesan fitunjukkan pada Tabel 5.8

Tabel 5.8 Skenario Uji Membaca Pesan

ID	UC-06
Nama	Uji Coba Membaca Pesan
Tujuan	Menguji fungsionalitas membaca pesan
Skenario	1. Pengguna memilih pesan yang akan dibaca

	2. Pengguna memasukkan private key
Hasil harapan	1. Pesan yang diterima dienkripsi pada browser 2. Pesan ditampilkan dalam bentuk plaintext

5.2.1.7 Uji *Sniffing*

Sniffing digunakan untuk memantau data yang keluar dan masuk melalui jaringan. Uji *sniffing* dilakukan untuk melihat bagaimana data pesan yang dikirimkan dan diterima oleh *client*. Data pesan yang dikirmkan dari *client* ke *server* dan dari *server* ke *client* sesaui dengan perancangan perangkat lunak adalah berupa ciphertext. Pengujian ini akan dilakukan pada sisi client menggunakan wireshark sebagai alat dalam melakukan *sniffing*. Proses sniffing dilakukan pada protokol HTTP. Detail mengenai skenario uji *sniffing* dapat dilihat pada Tabel 5.9.

Tabel 5.9 Skenario Uji *Sniffing*

ID	UC-07
Nama	Uji *Sniffing*
Tujuan	Menguji pengiriman data melalui jaringan
Skenario	1. Pengguna melakukan aktivitas mengirim dan membaca pesan 2. Dilakukan *sniffing* menggunakan wireshark 3. Mencari informasi yang dikirimkan *client* dan *server*
Hasil harapan	Informasi isi pesan yang didapatkan dalam bentuk ciphertext.

5.2.2 Skenario Uji Performa

Uji coba performa dilakukan untuk melihat bagaimana kemampuan perangkat lunak dalam melakukan operasi

fungsiolitas. Pada skenario ini akan di uji proses enkripsi dan dekripsi yang dilakukan pada browser.

5.2.2.1 Uji Kecepatan Proses Enkripsi

Uji kecepatan proses enkripsi bertujuan untuk menguji seberapa cepat perangkat lunak melakukan proses enkripsi yang dijalankan pada browser. Perhitungan peoses menggunakan web API yaitu *performance.now()*. Enrkipsi akan dilakukan pada ext dan file pdf

Tabel 5.10 Skenario Uji Kecepatan Proses Enkripsi

ID	UP-01
Nama	Uji Kecepatan Proses Enkripsi
Tujuan	Menguji kecepatan proses enkrisi pada browser
Data	1. Text 2. File Pdf
Hasil Uji Coba	Waktu yang digunakann untuk melakukan proses enkripsi data masukan.

5.2.2.2 Uji Kecepatan Proses Dekripsi

Uji kecepatan proses dekripsi bertuuan untuk menguji seberapa cepat proses dekripsi dilakukan pada browser. Data yang digunakan dalam pengujian Proses dekripsi dilakukan

Tabel 5.11 Skenario Uji Kecepatan Proses Dekripsi

ID	UP-02
Nama	Uji Kecepatan Proses Dekripsi
Tujuan	Menguji kecepatan proses dekripsi pada browser
Data	1. Ciphertext 2. File Cipher

Hasil Uji Coba	Waktu yang diperlukan untuk melakukan dekripsi ciphertext dan file cipher.

5.2.2.3 Pengukuran Ukuran Ciphertext

Uji coba ini akan membandingkan ukuran text dan file asli dengan ciphertext setelah proses enkripsi dilakukan. Uji coba ini bertujuan untuk melihat seberapa besar perbedaan ukuran plainttext dengan ciphertext.

Tabel 5.12 Skenario Pengukuran Ukuran Ciphertext

ID	UP-03
Nama	Pengukuran ukuran ciphertext
Tujuan	Membandingkan ukuran ciphertext dengan plaintext
Data	1. Text 2. File pdf
Hasil Uji Coba	Ukuran ciphertext dan perbandingannya dengan plaintext.

5.3 Hasil Uji Coba dan Evaluasi

Pada subbab ini akan dijelaskan mengenai hasil pengujian yang dilakukan pada skenario uji coba fungsionalitas dan performa.

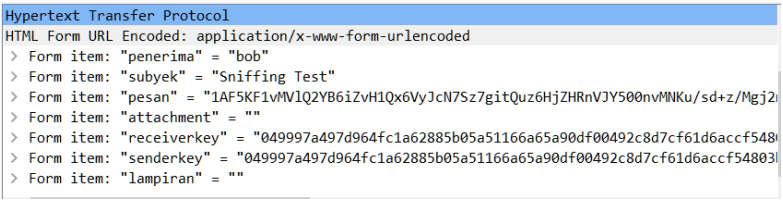
5.3.1 Uji Fungsionalitas

Sesuai dengan skenario uji coba fungsionalitas, berikut disampaikan hasil yang didapatkan dari uji coba fungsionalitas pada sistem yang telah dibangun. Hasil uji coba fungsionalitas ditunjukkan pada Tabel 5.13.

Tabel 5.13 Hasil Uji Fungsionalitas

ID	Nama Uji Coba	Hasil Uji Coba
UC-01	Uji Coba Mendaftarkan Pengguna Baru	Terpenuhi
UC-02	Uji Coba Manajemen Profile Pengguna	Terpenuhi
UC-03	Uji Coba Menampilkan Daftar Pesan Masuk	Terpenuhi
UC-04	Uji Coba Menampilkan Daftar Pesan Terkirim	Terpenuhi
UC-05	Uji Coba Menulis Pesan Baru	Terpenuhi
UC-06	Uji Coba Membaca Pesan	Terpenuhi
UC-07	Uji *Sniffing*	Terpenuhi

Uji *sniffing* pada protokol HTTP menggunakan Wireshark dilakukan untuk mendapatkan pesan yang kirimkan oleh *client* maupun *sever*. IP dari komputer server yang digunakan adalah 10.151.32.45. alamat ini hanya dapat diakses memalui jaringan lokal di Informatika ITS. Untuk mendapatkan pesan yang dikirimkan oleh *client* digunakan filter *ip.dst == 10.151.32.45 && http.request.method == POST*.

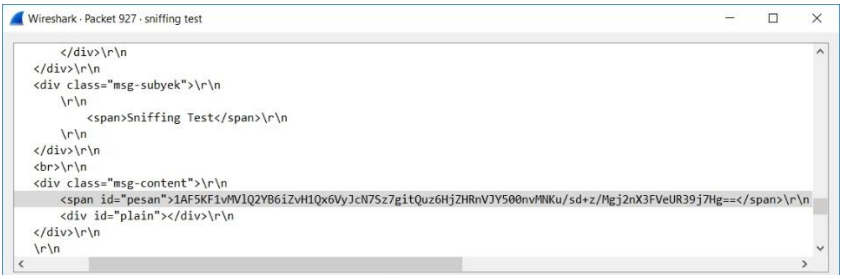


Gambar 5.1 Hasil Sniffing HTTP POST Request

Hasil sniffing terhadap pesan yang dikirimkan pengguna ditunjukkan pada Gambar 5.1. Dari hasil tersebut didapatkan pesan yang dikirimkan pengguna dalam bentuk ciphertext. Kunci AES yang telah dienkripsi menggunakan algoritma ECIES juga kiririmkan bersamaan dengan pesan tersebut. Berdasarkan hasil tersebut, dengan menggunakan protokol HTTP, keamanan pesan saat dikirimkan tetap terjadi. Penggunaan protokol HTTP tetap memungkinkan untuk serangan *sniffing*, tetapi data yang didapatkan adalah ciphertext yang tidak bisa dibaca tanpa mengggunakan *key* yang cocok. Dengan hasil ini, perangkat lunak dapat dikembangkan pada jaringan public tanpa menggunakan protokol HTTPS yang membutuhkan *Secure Socket Layer* (SSL).

Dengan menggunakan gabungan algoritma AES CTR dan ECIES, distribusi kunci AES menjadi lebih sederhana. Kunci AES dapat dikirimkan bersamaan dengan ciphertext melalui jaringan publik.

Untuk mendapatkan pesan yang dikirimkan oleh server menggunakan filter *ip.src == 10.151.32.45 && http.response.code == 200* pada Wireshark



Gambar 5.2 Hasil Sniffing HTTP GET Request

Hasil sniffing terhadap pesan yang dikirimkan server kepada client pada halaman view ditunjukkan pada Gambar 5.2. Dari hasil tersebut didapatkan pesan masih dalam bentuk *ciphertext* ketika dikirimkan kepada *client*. File html yang diterima

oleh pengguna berisikan *ciphertext*. Ciphertext ini kemudian akan dilakukan proses dekripsi ketika sudah diterima oleh *client*.

Pada bagian *backend*, Penyimpanan pesan pada basis data ditunjukkan pada Gambar 5.3. Pesan yang disimpan dalam basis data juga merupakan ciphertext. Dengan ini, isi dari pesan akan tetap aman jika terjadi serangan terhadap basis data.

id_pesan	pengirim	penerima	subyek		pesan	
qphGVntDJ1	delta	bob	new	3B	ZgI2uyNQLVlT3bqE0HHI1TavgQ==	28B
cywLI1OhYs	bob	alice	tes 8	5B	XQDi240KL1kNzzGzUcsYQmFBOGzVVYyGSg7S5DHG42ywiiHHoRUMF1SV7...	1K
rlHcrAhH6T	bob	alice	tes 10	6B	SQMgd9wML1lMBFxCvXr82jMTwFRf7KJ3Oe88d9O8QPh1BtSiCpkP4LXDQ...	2K
kTTHIkrgr7	bob	alice	tes 7	5B	rwHerDEKL1loVfbAtNlA+4PBwSs6gVDnRJ0bQZ89pDZdheea1xrHcU4hp...	1K
4Z1Fs9dcct	charlie	alice	tes 17	6B	nQIVdQ2xL1kLxaif2URxNBCtp9frqA==	32B
sQiUxsduDi	bob	alice	tes 14	6B	lgNNKeoOL1lkmOKd9Bd2u38genc7XpJB2WaWlhyN6bcWU9RpccrE001jK...	6K
dzeNiwt1Jf	bob	alice	tes 1	6B	KQKmdV0GL1lXkoCnFHSjsBbKmSRqAXQsYzr4c+aIOr8PAARjgLurh+xCu...	88B
MAuPWSFtNk	bob	alice	tes 13	6B	kQFeGjoOL1me5vrTNPe0N5r3AIuT45VDjXzkwsQDyZjtvb5tubIpZ0c63...	5K
pZ1JBob5PF	bob	alice	tes 3	5B	JgImyW8IL1kx2asE0yuhiErGXAOpCibxu81/Evs50PQ59CQ2azUjJHUCe...	368B
fJzqYahz24	bob	alice	tes 4	5B	HwNtV+gILlnW7DDz6YX9uuCuTxQjjn1YlH/Y/TfSnugIJVM0R4WDOpN/T...	488B
q98qw8vNUx	charlie	alice	Test 16	7B	GwFVWuSwL1ld/uzPD1x6rrn1XakVOxw=	32B
NQmdCm1xMy	bob	alice	tes 11	6B	GgDp2UsML1mP/DYsQeHTLETFDsxq1PEhSLZTLtG3YoXRuVHohaVqn5PN1...	3K
4eDOwG2Ggt	bob	alice	tes 9	5B	gALS+B8LL1nhpkHuazpjVtrk2Y28aXeA9foF659pybZ/J4wY9KisQsD3c...	2K
4U4wn7pIDh	bob	alice	tes 2	5B	FwO25CcIL1m/bHeSJa1uF2rwAVL2zE5IHpwfqqnIDaO3Izcz7QG9+3F3x...	224B
rcX1x35T3X	bob	alice	tes 12	6B	DgJ9uFINL1k1v3+X18cpSfAAqpFTB6F7M7KMxryNurT0bggAIbs9fgG1x...	4K
Hb7K1OnK4w	bob	alice	tes 5	5B	5AFGd0sJL1lulxm2Z5+1r7uoD174wZ/do8AjJrHYbImIW613NAfS7X126...	624B
vEugdLGruJ	bob	alice	tes 6	5B	2gGmn68JL1mNgpjOWze8H02CKe14TDB+Ligtx/pf0wz4g4wOJHbnRoUU1...	800B
WCM56o4Poi	bob	alice	tes 15	6B	1gFPUcIPL1k+Pq9rRsDk09sy6rIYxBajhB/BH3AvDMJtRrJizs42uy7KJ...	7K

Gambar 5.3 Penyimpanan Pada Basis Data

5.3.2 Uji Performa

Sesuai dengan skenario uji coba performa, pada uji coba performa akan didapatkan hasil kecepatan proses enkripsi, kecepatan proses dekripsi, dan perbandingan ukuran *plaintext* dengan *ciphertext*.

5.3.2.1 Uji Kecepatan Proses Enkripsi

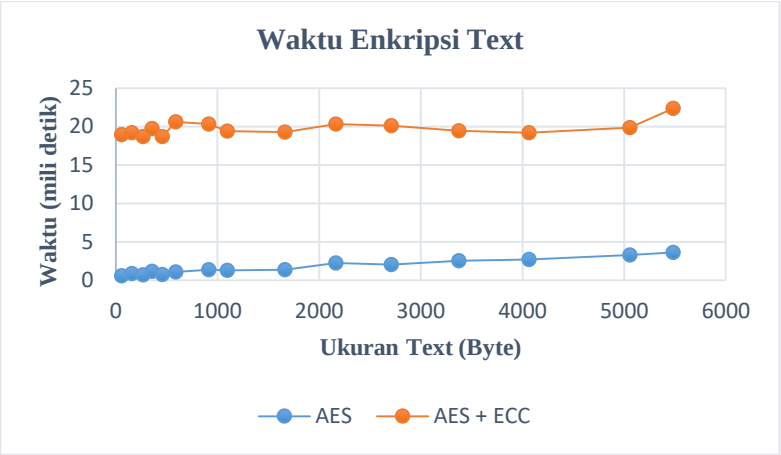
Pada uji kecepatan proses enkripsi akan dilakukan pengukuran waktu dimulai saat proses enkripsi dilakukan hingga proses enkripsi selesai. Pengiriman data tidak Setiap data test dilakukan uji coba sebanayak lima kali dan didapatkan rata-rata dari hasil tersebut. Kecepatan proses enkripsi juga akan dibandngkan dengan saat menggunakan algoritma AES CTR saja. Satuan waktu yang digunakan adalah milidetik. Hasil uji kecepatan

proses enkripsi dengan panjang text yang berbeda ditunjukkan pada Tabel 5.14.

Tabel 5.14 Hasil Uji Kecepatan Proses Enkripsi Text

No	Panjang text (Byte)	AES CTR	AES CTR + ECIES
1	58	0,557	18,966
2	158	0,886	19,191
3	268	0,691	18,690
4	358	1,150	19,739
5	458	0,757	18,716
6	590	1,087	20,603
7	914	1,348	20,329
8	1097	1,264	19,418
9	1663	1,354	19,297
10	2166	2,255	20,337
11	2711	2,044	20,102
12	3375	2,543	19,460
13	4064	2,703	19,192
14	5055	3,275	19,888
15	5481	3,634	22,363

Hasil uji enkripsi dalam bentuk grafik ditunjukkan pada Gambar 5.4. Dari grafik tersebut menunjukkan peningkatan ukuran text tidak selalu berbanding lurus dengan peningkatan waktu enkripsi. Namun grafik memiliki *trendline* naik.



Gambar 5.4 Grafik Waktu Enkripsi Text

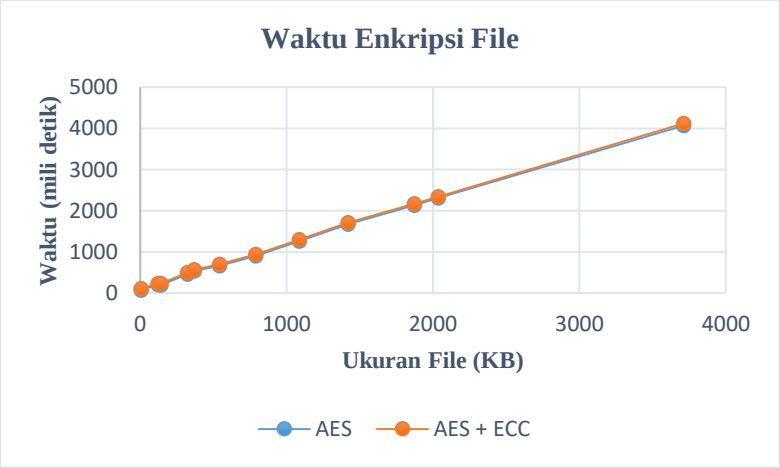
Pada uji kecepatan proses enkripsi file, tipe file yang digunakan adalah pdf. Hasil uji kecepatan proses enkripsi file ditunjukkan pada Tabel 5.15. Setiap file memiliki ukuran yang berbeda untuk melihat bagaimana pengaruhnya terhadap waktu enkripsi. Hasil tersebut juga akan dibandingkan dengan kecepatan proses enkripsi saat menggunakan algoritma AES saja.

Tabel 5.15 Hasil Uji Kecepatan Proses Enkripsi File

No	Ukuran File (KB)	AES CTR	AES CTR + ECIES
1	6	69,260	109,695
2	124	203,610	237,825
3	142	194,930	234,970
4	325	457,390	498,845
5	370	534,490	572,230
6	543	663,620	699,410
7	790	901,385	940,860
8	1087	1258,820	1300,900

No	Ukuran File (KB)	AES CTR	AES CTR + ECIES
9	1419	1674,060	1711,380
10	1873	2128,415	2168,330
11	2036	2307,675	2345,410
12	3713	4067,615	4125,970

Hasil uji kecepatan enkripsi file dalam bentuk grafik ditunjukkan pada Gambar 5.5. Dari grafik tersebut terlihat bahwa kenaikan waktu enkripsi secara linear.



Gambar 5.5 Grafik Waktu Enkripsi File

Pada proses enkripsi text dan file, waktu operasi meningkat jika dibandingkan dengan AES CTR saja. Selisih waktu ini adalah waktu yang dperlukan untuk melakukan proses enkripsi kunci AES menggunakan algoritma ECIES. Terdapat selisih dengan rata-rata 18,089 milidettik untuk enkripsi text dan rata-rata 40,380 milidetik untuk enkripsi file.

Kenaikan waktu yang diperlukan untuk melakukan proses enkripsi seccara keseluruhan tidak selalu sama pada setiap data tes. Perubahan waktu yang tidak konstan dipengaruhi faktor-faktor lain. Karena proses berjalan pada browser, waktu dipengaruhi oleh *service* lain yang sedang berjalan pada *browser*.

5.3.2.2 Uji Kecepatan Proses Dekripsi

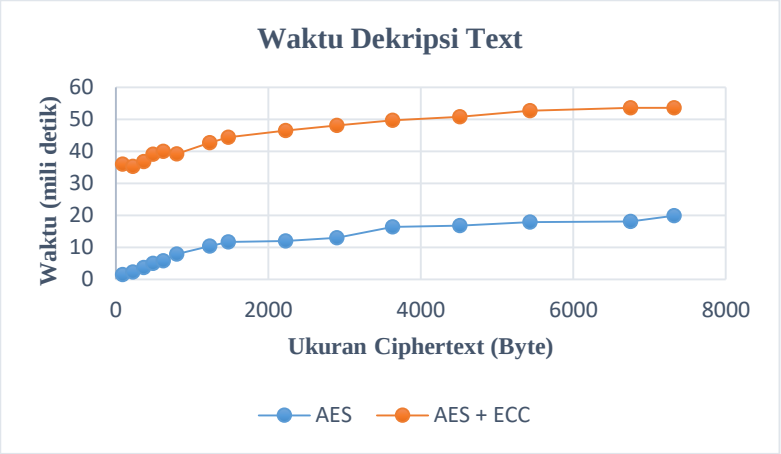
Pada uji kecepatan proses dekripsi akan dilakukan pengukuran waktu dimulai saat proses dekripsi dilakukan hingga proses dekripsi selesai. Perhitungan akan dilakukan pada browser. Satuan waktu yang digunakan adalah milidetik. Setiap ciphertext akan dilakukan uji coba sebanyak lima kali dan diambil rata-rata dari kelima hasil tersebut. Kecepatan proses enkripsi juga akan dibandngkan dengan saat menggunakan algoritma AES CTR saja. Hasil uji kecepatan proses dekripsi dengan panjang ciphertext yang berbeda ditunjukkan pada Tabel 5.16.

Tabel 5.16 Hasil Uji Kecepatan Proses Dekripsi Text

No	Panjang Text (KB)	AES CTR	AES CTR + ECC
1	88	1,428	35,978
2	224	2,246	35,227
3	368	3,656	36,808
4	488	5,008	39,086
5	624	5,742	39,931
6	800	7,830	39,185
7	1232	10,375	42,672
8	1476	11,624	44,425
9	2228	11,972	46,491
10	2900	12,983	48,118
11	3628	16,409	49,669
12	4512	16,730	50,814
13	5432	17,827	52,673

No	Panjang Text (KB)	AES CTR	AES CTR + ECC
14	6752	18,073	53,562
15	7320	19,823	53,535

Hasil uji kecepatan proses deripsi *text* dalam bentuk grafik ditunjukkan pada Gambar 5.6. Sama seperti proses enkripsi penambahan ciphertext tidak selalu berbanding lurus dengan penambahan waktu. Namun grafik tetap menunjukkan *trendline* naik.



Gambar 5.6 Grafik Waktu Dekripsi Text

Pada uji coba dekripsi file, setiap file juga akan dilakukan uji coba sebanyak lima kali. Hasil dari rata-rata waktu dekripsi setiap file ditunjukkan pada Tabel 5.17.

Tabel 5.17 Hasil Uji Kecepatan Proses Dekripsi Text

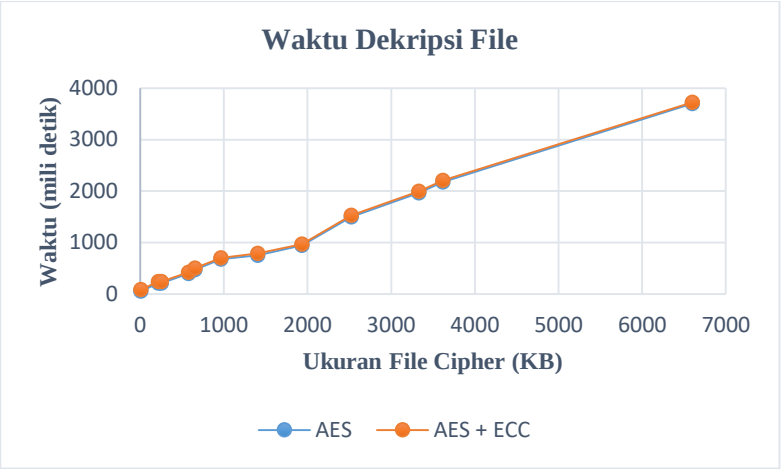
No	Ukuran File Cipher (KB)	AES CTR	AES CTR+ ECC
1	10	51,960	82,960

No	Ukuran File Cipher (KB)	AES CTR	AES CTR+ ECC
2	220	209,650	240,220
3	253	208,830	239,000
4	577	393,735	423,675
5	657	472,390	502,530
6	965	674,565	703,430
7	1405	753,800	786,805
8	1932	940,990	968,825
9	2522	1495,885	1529,800
10	3330	1963,420	1995,040
11	3619	2174,760	2205,940
12	6600	3698,320	3727,670

Hasil uji coba kecepatan proses dekripsi *file* dalam bentuk grafik ditunjukkan pada Gambar 5.7. Pada grafik terlihat bahwa semakin bersan ukuran cipher, semakin lama waktu yang diperlukan unutk proses dekripsi. Meskipun penambahan waktu tidak linier dan konstan.

Pada proses dekripsi text dan file, waktu operasi juga meningkat jika dibandingkan dengan AES saja seperti pada proses enkripsi. Selisih waktu ini adalah waktu yang dperlukan untuk melakukan proses dekripsi kunci AES menggunakan algoritma ECIES sebelum proses enkripsi pesan menggunakan algoritma AES dilakukan. Terdapat selisih dengan rata-rata 33,763 milidettik untuk dekripsi text dan rata-rata 30,632 miliddetik untuk dekripsi file.

Kenaikan waktu yang diperlukan untuk melakukan proses enkripsi seccara keseluruhan tidak selalu sama pada setiap data tes. Perubahan waktu yang tidak konstan dipengaruhi faktor-faktor lain. Karena proses berjalan pada browser, waktu dipengaruhi oleh *service* lain yang sedang berjalan pada *browser*.



Gambar 5.7 Grafik Waktu Dekripsi File

5.3.2.3 Pengukuran Ukuran Ciphertext

Hasil pengukuran ukuran ciphertext akan digunakan untuk membadingkan ukuran ciphertext awal dengan file asli. Untuk perbandingan ukuran ciphertext dengan masukan text ditunjukkan pada Tabel 5.18. Dari hasil perbardingan *plaintext* dengan ciphertext, presentase penambahan ukuran *ciphertext* adalah 36.297%. Penambahan ukuran ciphertext untuk *plaintext* yang pendek lebih besar dibangingkan plaintext yang panjang. Trendline yang ditunjukkan adalah turun.

Tabel 5.18 Perbandingan Ukuran Ciphertext

No	Panjang Text (Byte)	Panjang Ciphertext (Byte)	Penambahan Ukuran (%)
1	58	88	51,724
2	158	224	41,772
3	268	368	37,313
4	358	488	36,312

No	Panjang Text (Byte)	Panjang Ciphertext (Byte)	Penambahan Ukuran (%)
5	458	624	36,244
6	590	800	35,593
7	914	1232	34,792
8	1097	1476	34,548
9	1663	2228	33,974
10	2166	2900	33,887
11	2711	3628	33,825
12	3375	4512	33,688
13	4064	5432	33,661
14	5055	6752	33,570
15	5481	7320	33,552

Untuk hasil perbandingan ukuran file dengan cipher ditunjukkan pada Tabel 5.19. Berbeda dengan enkripsi text, penambahan ukuran cipher pada enkripsi file jauh lebih tinggi yaitu dengan rata-rata 76,834%. Semakin Besar ukuran file yang dienkripsi, presentase penambahan ukuran cipher juga lebih besar.

Tabel 5.19 Perbandingan Ukuran Cipher File

No	Ukuran File (KB)	Ukuran Cipher File (KB)	Penambahan Ukuran (%)
1	6	10	66,666
2	124	220	77,419
3	142	253	78,169
4	325	557	71,384
5	370	675	82,432
6	543	965	77,716
7	790	1405	77,848
8	1087	1932	77,736

No	Ukuran File (KB)	Ukuran Cipher File (KB)	Penambahan Ukuran (%)
9	1419	2522	77,730
10	1873	3330	77,789
11	2036	3619	77,750
12	3713	6660	79,369

Penambahan ukuran cipher adalah sama dengan menggunakan algoritma AES CTR saja karena pesan dienkripsi juga menggunakan algoritma AES CTR. Enkripsi ECIES hanya digunakan untuk proses enkripsi AES key sehingga tidak mempengaruhi ukuran ciphertext.

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak nantinya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba implementasi metode enkripsi *Advanced Encryption Standard* (AES) dan *Elliptic Curve Cryptography* (ECC) adalah sebagai berikut:

1. Penggabungan metode enkripsi AES dan ECC dilakukan dengan enkripsi pesan menggunakan algoritma AES dan enkripsi AES *key* menggunakan algoritma ECC.
2. Penggunaan gabungan algoritma AES CTR dan ECIES, distribusi kunci AES menjadi lebih mudah dan sederhana karena kunci AES dapat dikirimkan bersamaan dengan pesan yang telah terenkripsi. Namun memiliki penambahan waktu operasi yang membuat proses menjadi lebih lama.
3. Kecepatan proses enkripsi dan dekripsi pada browser dipengaruhi oleh service lain yang sedang berjalan pada *browser* tetapi memiliki kecerendungan naik seiring dengan ukuran text dan file yang bertambah. Browser juga akan menjadi sedikit tidak responsif ketika proses enkripsi dan dekripsi sedang berjalan.
4. Ukuran cipher mengalami penambahan yang cukup besar yaitu rata-rata 36,297% untuk text dan 76,835% untuk file. Penambahan ukuran yang cukup besar membuat pengiriman data dan penyimpan pada server memerlukan sumber daaya yang lebih besar.

6.2 Saran

Saran yang diberikan terkait pengembangan pada tugas akhir ini adalah sebagai berikut:

1. Pengembangan *service in background* untuk proses enkripsi dan dekripsi untuk menjaga reponsivitas browser saat melakukan proses tersebut.
2. Manajemen *private key* untuk mempermudah pengguna dalam memakai perangkat lunak.
3. Dilakukan kritanalisis untuk menguji ketahanan ciphertext.

DAFTAR PUSTAKA

[1] J. C. D. Q. W. W. Xiang Li, "Research and realization based on hybrid encryption algorithm of improved AES and ECC," *2010 International Conference on Audio, Language and Image Processing, Shanghai*, pp. 396-400, 2010.

[2] R. B. Nishtha Mathur, "AES Based Text Encryption Using 12 Rounds with Dynamic Key Selection," *Procedia Computer Science*, pp. 1036-1043, 2016.

[3] S. Gueron, "Intel® Advanced Encryption Standard (AES) New Instructions Set," *Intel Corporation*, 2010.

[4] U. Kretzschmar, "AES128 – A C Implementation for Encryption and Decryption," Texas Instruments, 2009.

[5] K. M. S. Laiphrakpam Dolendro Singh, "Implementation of Text Encryption using Elliptic Curve Cryptography," *Procedia Computer Science*, vol. 54, pp. 73-82, 2015.

[6] Y. A. Rawya Rizk, "Two-phase hybrid cryptography algorithm for wireless sensor networks," *Journal of Electrical Systems and Information Technology*, vol. 2, no. 3, p. 296–313, 2015.

[7] G. s. P. (. N. S. Abhinandan Aggarwal, "Implementation of AES algorithm," *International Journal of Engineering Research & Science*, vol. 2, pp. 112-116, 2016.

[8] J. D. Vincent Rijmen, "Advanced Encryption Standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, 2001.

[9] M. Dworkin, "Recommendation for Block Cipher Modes of Operation," National Institute of Standards and Technology, 2001.

[10] A. MS, "Elliptic Curve Cryptography," *An Implementation Guide*, 2007.

[11] D. Flanagan, JavaScript: The Definitive Guide (6th ed.), O'Reilly & Associates, 2011.

[12] [Online]. Available: https://nodejs.org/en/. [Accessed 5 March 2017].

[13] [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. [Accessed 16 May 2017].

[14] [Online]. Available: https://www.jetbrains.com/idea/. [Accessed 12 March 2017].

BIODATA PENULIS

Armirara Refa Aziz, lahir di Blitar, tanggal 24 Desember 1994. Penulis menempuh pendidikan formal mulai dari SDI Kardina Massa (2001-2007), SMP Negeri 1 Blitar (2007-2010), SMA Negeri 1 Blitar (2010-2013), hingga S1 jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember (2013-2017).

Selama masa perkuliahan di Teknik Informatika, penulis mengikuti kegiatan organisasi dengan berpartisipasi sebagai anggota Departemen Riset dan Teknologi HMTC (2014-2015). Selain itu penulis juga mengikuti kegiatan kepanitiaan yaitu ITS EXPO 2014 dan 2015, Schematics 2014 dan 2015. Penulis juga pernah melakukan kerja praktik di PT Krakatau Steel (Persero) Tbk. Penulis dapat dihubungi melalui email pribadi di *armirara.refa@gmail.com*.