BAB 5 IMPLEMENTASI

Pada bab ini akan memaparkan mengenai implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Pada bab ini terdiri dari subbab implementasi algoritma dan implementasi antar muka. Implementasi algoritma bertujuan untuk menjabarkan proses algoritma yang telah dibuat pada tahap perancangan kedalam sistem. Implementasi antarmuka bertujuan untuk memberi gambaran antarmuka sistem yang telah dibuat berdasarkan proses perancangan sebelumnya. Implementasi lingkungan pengujian bertujuan guna mengetahui gambaran lingkungan pengujian yang dilakukan.

5.1 Implementasi Algoritma

Penulisan subbab ini akan mencantumkan implementasi dari perancangan proses algoritma yang telah dibuat sebelumnya. Implementasi algoritma yang dijabarkan yaitu implementasi proses tukar kunci, implementasi proses enkripsi, serta implementasi proses dekripsi. Implementasi ini akan direpresentasikan dalam bentuk *source code* dengan bahasa pemrograman Java dengan aplikasi Android Studio versi 2.2.

5.1.1 Implementasi Proses Tukar Kunci

Pada bagian ini akan dijabarkan implementasi dari proses tukar kunci yang telah dirancang pada bagian sebelumnya. Dalam proses ini akan didapat *shared secret* yang kemudian akan digunakan untuk proses enkripsi dan dekripsi. Tabel 5.1 merupakan potongan *source code* untuk implementasi proses tukar kunci.

Tabel 5.1 Implementasi Proses Tukar Kunci

```
1  public abstract class EcdhUtil {
2  public static KeyPair generateKeyPair() {
3     try {
4        KeyPairGenerator keyPairGenerator =
   getEcdhKeyPairGenerator();
5        Initialize
   keyPairGenerator((ELLIPTIC_CURVE_NAME_SECP256K1));
6        return keyPairGenerator.generateKeyPair();
7     } catch {
8     }
9     return null;
10 }
11 private static KeyPairGenerator getEcdhKeyPairGenerator() {
12    try {
13       return KeyPairGenerator.getInstance(KEY_AGREEMENT_ALGORITHM,
                CryptoUtil.getSpongyCastleProvider());
14    } catch (NoSuchAlgorithmException e) {
15       throw new RuntimeException(e);
16    } catch (NoSuchProviderException e) {
17       throw new RuntimeException(e);
```

```
18      }
19  }
20      private static KeyAgreement getEcdhKeyAgreement() {
21        try {
22           return KeyAgreement.getInstance(KEY_AGREEMENT_ALGORITHM,
                    CryptoUtil.getSpongyCastleProvider());
23        } catch (NoSuchAlgorithmException e) {
24                    throw new RuntimeException(e);
25        } catch (NoSuchProviderException e) {
26           throw new RuntimeException(e);
27        }
28      }
29  }
30
```

Penjelasan dari *source code* pada tabel 5.1 adalah sebagai berikut:

1. Pada baris 4-6 merupakan proses untuk melakukan *generate* kunci serta inisialisasi mode *elliptic curve* yang akan digunakan.
2. Pada baris 11-19 merupakan method dengan mengembalikan nilai proses KeyPairGenerator
3. Pada baris 20-26 merupakan method dengan mengembalikan nilai proses KeyAgreement

5.1.2 Implementasi Proses Enkripsi

Proses enkripsi dilakukan berdasarkan *plaintext* input dari pengguna untuk kemudian dilakukan proses enkripsi. Tabel 5.2 merupakan potongan *source code*untuk implementasi fitur enkripsi.

Tabel 5.2 Implementasi Fitur Enkripsi

```
1   public abstract class CryptoUtil {
2   public static byte[] encryptAes() {
3      try {
4         Cipher cipherOut = getAesCipher();
5         int uniqueNumber =
    InitializationVectorUtil.getNextUniqueNumber();
6         byte[] iv =
    ByteBufferUtil.getIntegerInEmptyArray(LENGTH_AES_IV,
    uniqueNumber);
7         cipherOut.init(Cipher.ENCRYPT_MODE, new
    SecretKeySpec(keyArray, CIPHER_ALGO_AES), new
    IvParameterSpec(iv));
8         byte[] enc = cipherOut.doFinal(clearText);
9         ByteBuffer bb = ByteBuffer.allocate(4+enc.length);
10        byte[] ivAndEnc = bb.putInt(uniqueNumber).put(enc).array();
11        return ivAndEnc;
12  }
```

```
13 public static byte[] genAesKey() {
14    try {
         KeyGenerator keyGen =
KeyGenerator.getInstance(CIPHER_ALGO_AES,
15 getSpongyCastleProvider());
16       keyGen.init(128);
17       return keyGen.generateKey().getEncoded();
18 }
19 private static Cipher getAesCipher() {
20       try {
            return Cipher.getInstance(CIPHER_MODE_AES_OFB,
21 getSpongyCastleProvider());
22    }
23 }
```

Penjelasan dari *source code* pada tabel 5.2 adalah sebagai berikut:

1. Pada baris 4-10 merupakan proses inisialisasi objek
2. Pada baris 14-17 merupakan proses untuk inisialisasi panjang kunci algoritma AES yang akan digunakan
3. Pada baris 20-21 merupakan proses untuk inisialisasi mode AES dan *library* yang digunakan.

5.1.3 Implementasi Proses Dekripsi

Proses dekripsi dilakukan saat sistem menerima *ciphertext* yang telah dikirim oleh pengguna lainya. Tabel 5.3 merupakan potongan *source code* untuk implementasi fitur dekripsi

Tabel 5.3 Implementasi Fitur Dekripsi

```
   public static byte[] decryptAes(final byte[] input, final byte[]
1  keyArray) {
2     try {
3        ByteBuffer bb = ByteBuffer.wrap(input);
         byte[] iv =
4  ByteBufferUtil.getIntegerInEmptyArray(LENGTH_AES_IV, bb.getInt());
         byte[] enc = ByteBufferUtil.getNextByteArray(bb,
5  bb.remaining());
6        Cipher cipherIn = getAesCipher();
         cipherIn.init(Cipher.DECRYPT_MODE, new
7  SecretKeySpec(keyArray, CIPHER_ALGO_AES), new IvParameterSpec(iv));
8        return cipherIn.doFinal(enc);
9  }
10 private static Cipher getAesCipher() {
11       try {
            return Cipher.getInstance(CIPHER_MODE_AES_OFB,
12 getSpongyCastleProvider());
13    }
14 }
```

Penjelasan dari *source code* pada tabel 5.3 adalah sebagai berikut:

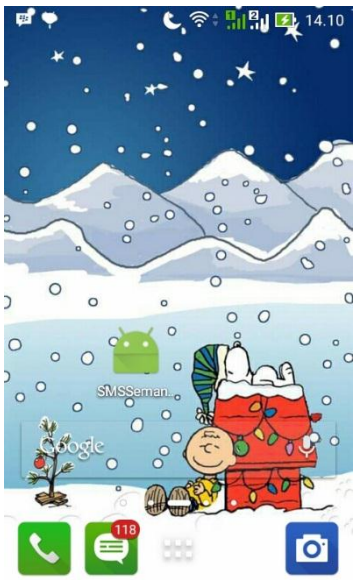
1. Pada baris 3-6 merupakan proses inisialisasi objek
2. Pada baris 7-8 merupakan perintah untuk melakukan dekripsi dengan menggunakan mode dekripsi serta kunci rahasia yang didapat
3. Pada baris 12 digunakan untuk menentukan jenis mode yang dipilih.

5.2 Implementasi Antar Muka

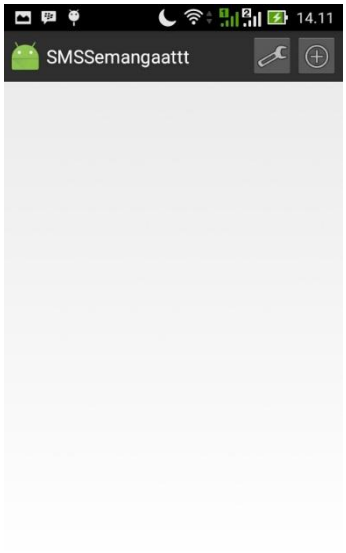
Antarmuka perangkat lunak yang dilakukan dalam pengujian dan analisis perbandingan algoritma AES untuk enkripsi SMS berbasis Android dibagi menjadi dua bagian, yaitu halaman awal dan buat pesan.

5.2.1 Antar Muka Halaman Awal

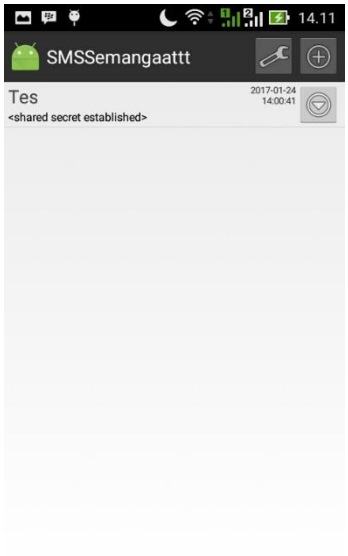
Antarmuka awal merupakan halaman pertama yang akan dibuka ketika perangkat lunak dijalankan. Dari antar muka awal pengguna akan melihat tampilan list obrolan yang kosong jika pengguna belum pernah melakukan obrolan atau list obrolan yang terisi jika pengguna telah melakukan obrolan.



Gambar 5.1 Antar Muka Halaman Awal



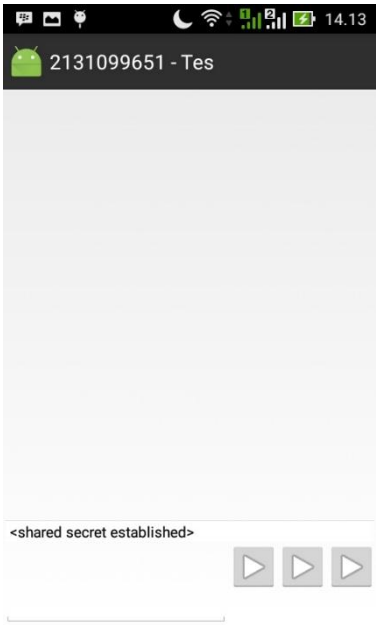
Gambar 5.2 Antar Muka Halaman Awal – Pengguna belum melakukan obrolan



Gambar 5.3 Antar Muka List Pesan

5.2.2 Antar Muka Buat Pesan

Antar muka buat pesan merupakan antar muka yang digunakan untuk melakukan proses enkripsi dan dekripsi, obrolan serta mengirim dan menerima pesan.

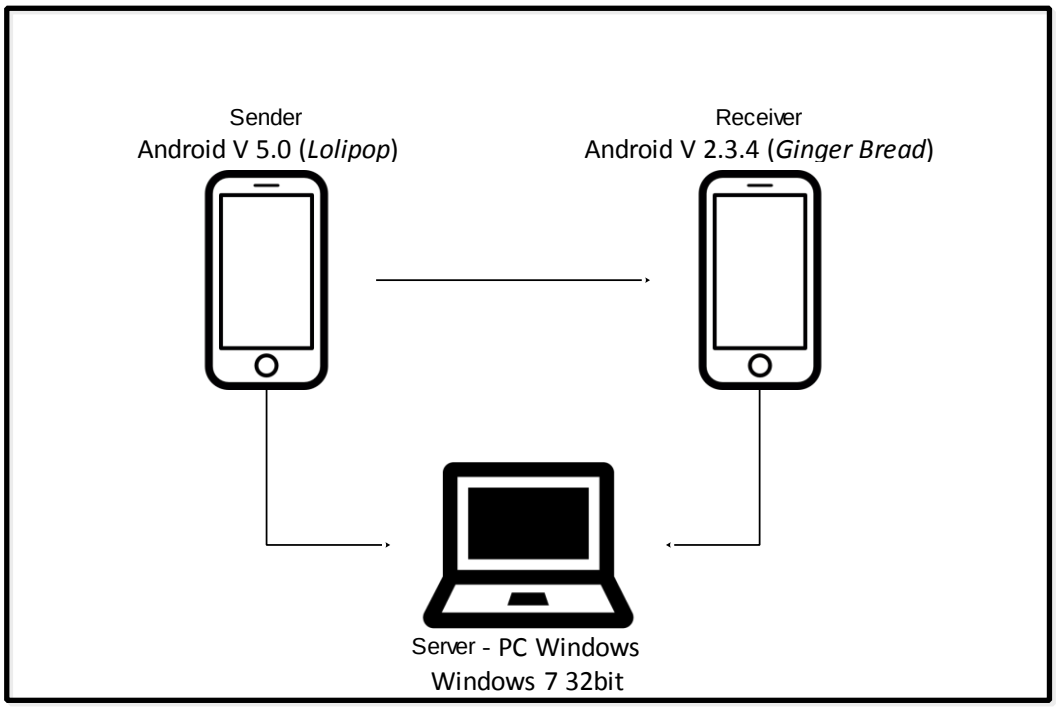


Gambar 5.4 Antar Muka Buat Pesan

5.3 Implementasi Lingkungan Pengujian

Pada implementasi lingkungan pengujian, perangkat lunak pengujian diinstall pada masing-masing host. Setiap host dihubungkan dengan server. Hal tersebut bertujuan untuk mengetahui proses yang berjalan pada setiap host dengan menggunakan fitur Android Monitor yang terdapat di Android studio yang digunakan pada server. Pada pelaksanaan implementasi berada di Laboratorium KCV gedung E Filkom Universitas Brawijaya.

Pada gambar 3.3 menunjukkan implementasi lingkungan pengujian dengan dua klien. Pengujian dengan dua klien bertujuan untuk mengetahui fungsi enkripsi dan dekripsi, performansi serta *avalanche effect*. Pada pengujian kedua ini *sender* maupun *receiver* dihubungkan dengan komputer *serve*r dan kemudian *sender* serta *receiver* menjalankan perangkat lunak pengujiannya. *Sender* akan melakukan proses enkripsi yang kemudian *ciphertext* hasil proses enkripsi tersebut akan di kirim ke *receiver* untuk selanjutnya dilakukan proses dekripsi. Dalam proses enkripsi dan dekripsi tersebut, *sender* dan *receiver* akan mengirim data pada *server* berupa panjang data, *plaintext* dan *ciphertext*, serta lama waktu proses enkripsi dan dekripsi untuk kemudian dilakukan analisa oleh peneliti.



Gambar 5.5 Implementasi Lingkungan Pengujian

5.3.1 Perangkat Lingkungan Pengujian

Perangkat keras yang digunakan dalam proses pengujian sistem enkripsi SMS pada telepon seluler berbasis Android dengan metode ECDH dan AES adalah satu buah laptop dan dua buah *smartphone* dengan spesifikasi yang dijelaskan sebagai berikut

a. Laptop 1

   Laptop 1 merupakan laptop yang bekerja sebagai server dalam setiap pengujian. Server digunakan untuk mengetahui proses yang berjalan pada host seperti waktu enkripsi dekripsi serta *ciphertext* hasil dar proses enkripsi.

b. Host 1 dan

   Host 1 merupakan piranti bergerak (*smartphone*) yang digunakan untuk melakukan proses enkripsi dan dekripsi maupun menerima dan mengirim pesan.

c. Host 2

   Host 2 merupakan piranti bergerak (*smartphone*) yang digunakan untuk melakukan proses enkripsi dan dekripsi maupun menerima dan mengirim pesan.

Beberapa perangkat lunak digunakan dalam pengujian ini. Perangkat lunak yang digunakan diantaranya:

- Android Studio sebagai *tools* pemrograman dan monitoring host
- Windows 7 Home Basic sebagai sistem operasi server
- Android versi 5.0 (Lolipop) dan Android versi 2.3.4 (Gingerbread) sebagai sistem operasi tiap host.