# APLIKASI KLIEN SUREL DENGAN ALGORITMA RABBIT PADA PONSEL ANDROID

Muhammad Anwari Leksono
Institut Teknologi Bandung
Jalan Cisitu Lama 3B/160B Bandung
Jawa Barat, Indonesia
+62-852-86078737
If18037@students.if.itb.ac.id

Rinaldi Munir
Teknik Informatika ITB
Jalan Ganesha no. 10
Bandung
rinaldi@informatika.org

## ABSTRAK
Makalah tugas akhir ini membahas penerapan algoritma Rabbit untuk mengamankan surel dalam bentuk aplikasi untuk ponsel Android. Makalah ini bertujuan untuk memahami algoritma Rabbit dan surel beserta mekanismenya untuk mengamankan surel. Aplikasi ini menggunakan algoritma Rabbit untuk mengenkripsi dan mendekripsi konten surel yang bersifat rahasia. Aplikasi dapat membuat konten rahasia, mengirim surel dengan konten tersebut, menerima surel, dan membaca surel lengkap dengan konten rahasia. Aplikasi dibuat dengan bahasa Java pada *platform* Android. Setelah diuji aplikasi dapat digunakan sebagai aplikasi klien surel untuk Google Mail yang sederhana dengan algoritma Rabbit yang berfungsi sebagai pengaman konten rahasia pada surel.

## Kata Kunci
Surel, aplikasi klien surel, algoritma Rabbit, ponsel Android

## 1. PENDAHULUAN
Surel adalah salah satu sarana komunikasi formal. Surel yang formal dapat mengandung konten yang bersifat rahasia. Ancaman surel yang sering dijumpai adalah penyadapan. Selain itu dari mekanisme *store-and-forward*, sebuah surel dapat diambil tiruannya dari server yang dilewati surel ketika menuju penerima.

Cara untuk mengamankan surel salah satunya adalah dengan cara mengenkripsinya. Algoritma enkripsi yang dipilih adalah algoritma Rabbit karena algoritma ini berjenis algoritma simetri. Algoritma simetri lebih unggul dari algoritma kunci publik dari segi kecepatan (KetuWare, 2004).

Pengaksesan lewat ponsel pada masa kini dapat dilakukan melalui ponsel cerdas dan salah satu sistem operasi yang populer adalah Android (Titlow, 2011). Dengan demikian untuk mengamankan surel pada ponsel Android, perlu dibangun suatu aplikasi klien surel yang dapat mengirimkan surel dengan konten terenkripsi.

Ada beberapa masalah pokok yang ditemukan pada saat akan membangun aplikasi klien surel ini. Masalah-masalah tersebut adalah sebagai berikut: integrasi algoritma Rabbit dengan aplikasi klien surel; cara membuat konten terenkripsi dan mengirimkannya dalam bentuk surel; cara menerima surel dari internet dan membaca surel termasuk konten yang terenkripsi.

## 2. STUDI LITERATUR

### 2.1 Algoritma Rabbit
Algoritma Rabbit adalah algoritma yang temukan oleh Fast Software Encryption pada tahun 2003 oleh martin Boesgaard, Mette Vesterager, Jesper Christiansen, dan Ove Scavenius. Algoritma ini menerima kunci sepanjang 128 bit dan *initial vector* sepanjang 64 bit. Algoritma ini mengolah kunci dan *initial vector* tersebut menjadi sebuat deretan *bit* semu acak sepanjang 128 bit. Algoritma ini adalah algoritma kunci simetri.

Algoritma Rabbit memiliki tujuh belas peubah. Delapan peubah *inner state* dengan ukuran masing-masing 32 bit, delapan *counter* dengan ukuran maing-masing 32 bit, dan satu *carry* berukuran satu bit. Algoritma Rabbit memiliki beberapa skema. Skema-skema tersebut skema persiapan kunci, skema *next state function*, dan skema ekstraksi.

Skema persiapan kunci adalah skema untuk memberikan nilai awal pada semua *inner state*, *counter*, dan *carry*. Skema *next state function* adalah skema yang menjelaskan aturan pengubahan nilai-nilai semua peubah. Skema ekstraksi berfungsi untuk megambil bit-bit dari *inner state* dengan susunan tertentu menjadi rangkaian bit berukuran 128 dan semu acak.

### 2.2 Sistem Surel
Surel adalah sebuah metode untuk bertukar pesan digital dari satu orang ke orang yang lain. Surel dalam operasinya menggunakan jaringan internet atau jaringan komputer lokal (LAN). Pada masa ini pengguna surel dapat saling berhubungan tanpa pengirim dan penerima berada *online* secara bersamaan. Model surel yang mengharuskan pengirim dan penerima berada pada saat yang sama lebih dikenal dengan *instant messaging*.

Layanan surel masa kini menggunakan metode *store-and-forward*. Model ini membuat surel ketika dikirim tidak akan langsung sampai pada penerima. Surel ini akan disimpan di server penyedia layanan surel dan ketika penerima membuka server ini, surel dapat dikatakan sudah sampai.

Sistem surel terbagi menjadi dua bagian (Partige, 2008). Kedua bagian itu adalah MHS (*message handling system*) dan UA (*user agent*). MHS adalah bagian yang bertanggung jawab menangani masalah pengiriman surel dari pengirim kepada penerima. Bagian kedua, UA, berfungsi sebagai media interaksi pengguna surel dengan sistem surel. Melalui UA pengguna dapat membuat dan membaca surel. Salah satu contoh UA adalah aplikasi klien surel.

Surel dikirim dengan menggunakan protokol SMTP. Protokol ini didesain dengan model sebagai berikut (Postel, 1982): pihak

pengirim membuat jalur komunikasi dengan penerima; pihak pengirim mengirimkan perintah MAIL sebagai tanda bahwa ia adalah pengirim; penerima membalas dengan perintah OK jika layanan bisa dijalankan; pengirim mengirimkan perintah RCPT yang menjelaskan penerima surel; jika penerima dapat mengenai penerima surel maka balasan yang diberikan adalah OK; pengirim memberikan perintah DATA diikuti dengan *stream* surel secara lengkap dan diakhiri dengan tanda titik.

Surel dapat diterima dengan menggunakan protokol POP3 (*post office protocol*) atau IMAP (*internet message access protocol*). Perbedaan antara POP3 dengan IMAP adalah sebagai berikut. Surel yang diakses dengan protokol IMAP membutuhkan akses ke berkas surel di internet. Protokol IMAP memungkinkan pengguna surel dapat menyuting dan memodifikasi surel secara langsung di server. Dengan kata lain IMAP membutuhkan koneksi yang memadai (Crispin, 2003).

Berbeda dengan IMAP, protokol POP3 mengharuskan penerima mengunduh surel-surel dari internet sebelum dapat dibaca. Cara ini membuat protokol POP3 tidak memerlukan koneksi yang bersifat terus-menerus. Setelah protokol POP3 mengunduh semua surel dari kotak masuk di internet, surel-surel di kotak masuk tersebut kemudian dihapus.

Surel yang digunakan pada masa kini memiliki format MIME (*multipurpose internet mail extensions*). Format ini memungkinkan sebuah surel memiliki karakter-karakter selain karakter ASCII dan lampiran dengan jenis selain teks. Surel dengan format ini memiliki *header* dengan kelengkapan sebagai berikut: MIME-Version, Content-ID, Content-Type, Content-Disposition, dan Content-Transfer-Encoding.

## 2.3 Sistem Operasi Android

Sistem operasi Android adalah sistem operasi buatan Google yang dijalankan di lingkungan ponsel cerdas atau *smartphone*. Burns (2008) menjelaskan bahwa Android dibangun dengan *platform* Linux yang dikembangkan dengan Java. Android juga telah dioptimasi untuk berjalan di lingkungan ponsel.

Keunggulan sistem operasi Android adalah sebagai berikut. Android bersifat *root-able* yang berarti pengguna dapat mengubah *filesystem* dari Android itu sendiri. Android bersifat kode sumber terbuka atau *open source*. Dengan demikian celah keamanan dari sistem operasi Android akan lebih cepat ditemukan (Bryan, 2011).

Pengembangan aplikasi *third-party* di lingkungan Android dilakukan dengan bahasa Java. Dukungan untuk bahasa C dan C++. Penggunaan bahasa C atau C++ tidak meningkatkan performa aplikasi tetapi pasti akan membuat kode program menjadi lebih kompleks (Turner, 2009).

# 3. ANALISIS MASALAH DAN PERANCANGAN SOLUSI

## 3.1 Pemilihan Bahasa Pemrograman

Ada tiga bahasa yang didukung oleh Android, yaitu Java, C, dan C++. Bahasa yang dipilih adalah Java. Java dipilih memiliki keunggulan dalam aspek pustaka sehingga pemrograman akan lebih cepat dan sederhana karena pustaka-pustaka yang berisi definisi tipe data, fungsi, dan prosedur mudah ditemukan.

Solusi yang disediakan Java untuk mendukung pembuatan aplikasi klien surel adalah pustaka JavaMail. Pustaka ini berisi definisi protokol-protokol yang biasa ditemukan pada layanan surel seperti SMTP, IMAP, dan POP3.

## 3.2 Algoritma Rabbit

Masalah untuk algoritma Rabbit ada pada jenis tipe data yang digunakan untuk operasi matematika pada algoritma ini dan tipe data untuk menyimpan nilai peubah. Java menyediakan tiga tipe data bilangan, yaitu *int, long*, dan *big integer*.

Tipe data untuk menyimpan nilai pubah yang dipilih adalah *long*. Peubah pada algoritma Rabbit berukuran 32 bit sedangkan tipe data *int* pada Java hanya bisa menampung nilai sampai $2^{31}$ -1. Tipe data *long* dapat menyimpan data sampai nilai $2^{63}$-1. Karena nilai peubah hanya sampai $2^{32}$ -1, tiap operasi matematika yang melibatkan nilai peubah harus dikenakan operasi tambahan, yaitu modulus $2^{32}$. Hal ini dapat menjaga nilai peubah tetap sepanjang 32 bit saja.

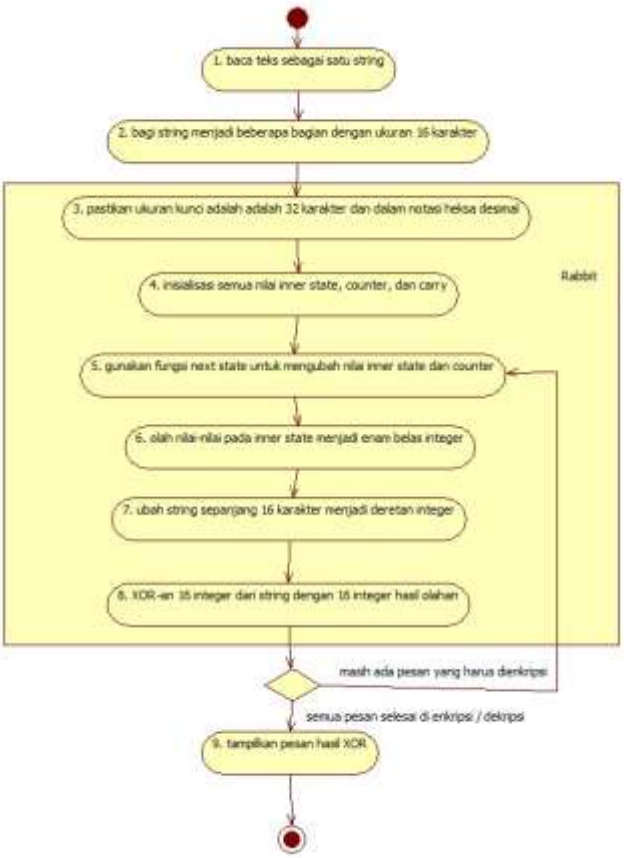
Algoritma Rabbit menerima kunci sepanjang 128 bit atau enam belas karakter. Akan tetapi kunci yang berikan untuk proses enkripsi atau dekripsi tidak bisa dipastikan berjumlah enam belas karakter. Untuk mengatasi hal ini, fungsi MD5 digunakan untuk menyeragamkan panjang kunci. Fungsi MD5 menerima input berupa *string* berukuran berapa saja dan mengembalikan sebuah untaian *byte* berjumlah enam belas. Enam belas *byte* dapat dibaca sebagai enam belas karakter.

Algoritma Rabbit untuk setiap saat akan hanya mengenkripsi *plaintext* sepanjang 16 karakter. Jika *plaintext* memiliki ukuran lebih panjang lebih dari 16 karakter maka *plaintext* akan dibagi menjadi beberapa blok *string* dengan panjang masing-masing adalah 16 karakter. Blok-blok ini dinotasikan dengan $B_1$, $B_2$, $B_3$, …, $B_n$. Jika blok terakhir berukuran kurang dari 16 karakter maka blok tersebut akan ditambahkan dengan spasi sebanyak 16 – (16 mod panjang $B_n$).

Penjelasan singkat mengenai cara kerja algoritma Rabbit adalah sebagai berikut. Pada awalnya semua *inner state*, *counter*, dan *carry* diberikan nilai awal sesuai dengan skema persiapan kunci. Kemudian nilai-nilai tersebut diubah dengan fungsi *next state*. Nilai-nilai yang telah diubah itu kemudian diekstrak dengan skema ekstraksi menjadi 16 bilangan bulat, misalnya $S_1$. Setelah itu blok pertama *plaintext*, $B_1$ di-XOR-an dengan $S_1$ dan menghasilkan blok ciphertext pertama dengan notasi $C_1$. Jika masih ada blok pesan yang akan dienkripsi maka fungsi *next state* dipanggil kembali, skema ekstraksi dipanggil untuk menghasilkan $S_2$, dan kemudian blok $B_2$ di-XOR-an dengan $S_2$ menjadi $C_2$. Hal ini berulang sampai $B_n$ di-XOR-an dengan $S_n$ dan menghasilkan $C_n$. Dengan demikian fungsi *next state* dan skema ekstraksi masing-masing akan digunakan sebanyak $n + 4$ kali dan $n$ kali karena pada skema persiapan kunci, fungsi *next state* dipanggil empat kali. Skema aktivitas algoritma Rabbit dalam melakukan enkripsi dan dekripsi pesan dapat dilihat pada gambar 1 di bawah ini.

Hasil dari operasi XOR antara $B_n$ dengan $S_n$ adalah $C_n$ yang berbentuk enam belas bilangan *integer*. Enam belas bilangan ini kemudian diterjemahkan menjadi 32 karakter dalam notasi heksadesimal. Dengan cara ini ukuran *ciphertext* adalah dua kali ukuran *plaintext*.

Proses dekripsi berjalan hampir sama dengan proses enkripsi. Perbedaannya adalah *ciphertext* dipecah menjadi beberapa bagian berukuran sama panjang, yaitu 32 karakter. Tiap-tiap bagian ini kemudian diubah menjadi enam belas bilangan bulat. Setelah itu semua bagian akan di-XOR-an dengan *string* semu acak hasil skema ekstrasi pada tiap iterasi.




**Gambar 1 Diagram Aktivitas Algoritma Rabbit**

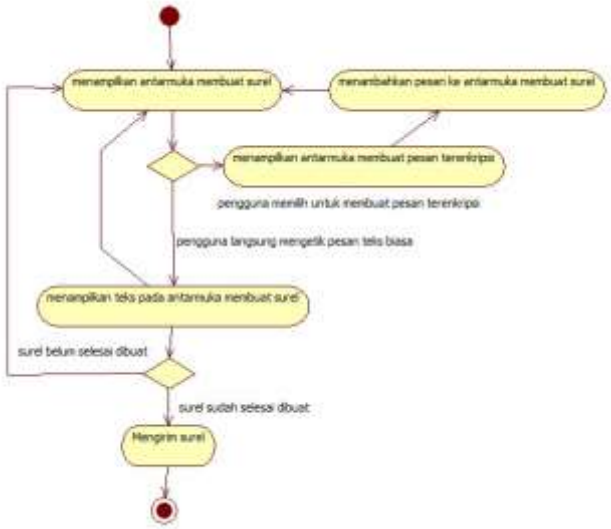
## 3.3 Proses Enkripsi dan Kaitannya dengan Pembuatan dan Pengiriman Surel

Sebelum mengirim surel, alamat tujuan, subjek, dan konten harus ditentukan lebih dulu. Konten surel dapat berupa konten terenkripsi, konten biasa, atau gabungan antar keduanya. Masalah yang ditimbulkan dari kemungkinan konten ini adalah cara membedakan antara konten biasa dengan konten terenkripsi dan cara untuk membaca keduanya.

Konten terenkripsi akan ditulis di antara dua buah label, yaitu <*enc*> dan </*enc*>. Penulisan *ciphertext* tidak dilakukan secara sekaligus tetapi *ciphertext* dibagi menjadi beberapa baris dan setiap baris terdiri dari 32 karakter. Kedua label harus ditulis pada baris baru dan pada baris itu tidak ada karakter lain selain label. Contoh penulisannya adalah sebagai berikut:

```
<enc>
34cf9bd6e25efd9a80b96dccb40c535b
f165551556be267fd4b69912dc0a98f0
68fa106eb93a8c0e7d9d8ba586ead0f1
</enc>
```

Konten yang tidak dienkripsi tidak ditulis dengan label apa pun dan tidak terikat dengan jumlah karakter untuk tiap barisnya. Diagram alur pembuatan konten surel dan pengiriman surel dapat dilihat pada gambar berikut ini.




**Gambar 2 Diagram Alur Pembuatan dan Pengiriman Surel**

Urutan penulisan konten terenkripsi dan konten biasa dapat dilakukan berbagai macam cara. Konten terenkripsi boleh mendahului konten biasa atau sebaliknya.

Surel dikirim dengan protokol SMTP. Pada pustaka JavaMail SMTP yan tersedia adalah SMTP dengan TLS. Protokol ini mengirim surel dalam keadaan terenkripsi. Sebelum surel dikirim, surel dibuat dalam format MIME. Konten surel yang digunakan adalah konten dengan tipe *text/plain* atau nilai Content-Type adalah *text*/*plain*. Tipe ini adalah tipe yang paling sederhana.

## 3.4 Proses Dekripsi dan Kaitannya dengan Penerimaan dan Pembacaan Surel

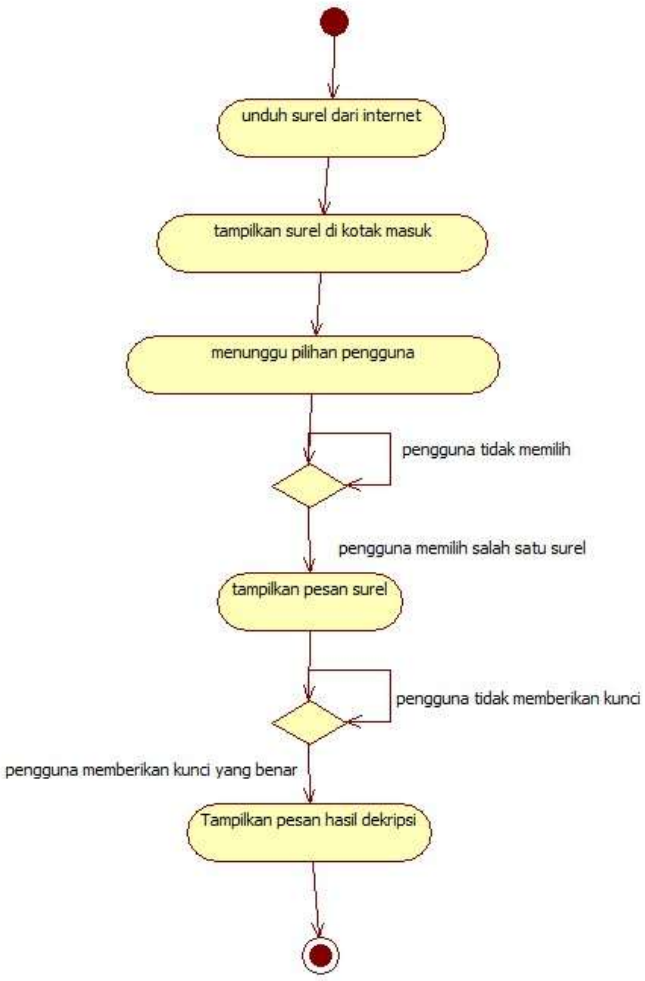
Sebelum surel dapat dibaca, surel harus diunduh lebih dulu dari internet. Cara menerima surel dari internet ada dua macam, yaitu dengan menggunakan fasilitas POP3 atau IMAP. Protokol yang dipilih adalah IMAP. Alasan protokol ini dipilih adalah protokol ini dapat mengambil surel dari internet tanpa harus menghapusnya sehingga surel yang sudah lama masih dapat dibaca kembali.

Surel yang diterima memiliki konten yang terdiri dari dua jenis pesan, yaitu pesan terenkripsi dan pesan biasa. Untuk membaca pesan secara utuh, pesan biasa dan terenkripsi harus dapat dipisahkan. Setelah dapat dipisah, pesan terenkripsi dapat didekripsi untuk kemudian dibaca.

Pesan terenkripsi dikenali dengan adanya label <*enc*> dan </*enc*>. Semua pesan yang berada diantara dua label ini akan dibaca sebagai pesan terenkripsi. Surel akan dibaca dengan cara per baris. Semua baris yang berada sebelum baris <*enc*> dan sesudah baris </*enc*> akan dibaca sebagai pesan biasa.

Diagram alur penerima dan pembaca surel apat dilihat pada gambar 3 berikut.

Urutan penempatan konten terenkripsi dan konten biasa pada surel dapat berbagai macam. Oleh karena itu pembacaan konten surel harus dapat mempertahankan urutan tersebut sehingga konten tidak kehilangan semantiknya.




**Gambar 3 Diagram Alur Penerimaan dan Pembacaan Surel**

Sebagai contoh, lihat pesan berikut ini.

```
The secret message is below
<enc>
34cf9bd6e25efd9a80b96dc0b44e1609
</enc>
This is the end of message
```

Ketika pesan di atas dekripsi dan dibaca, urutan pesan harus tetap sama, yaitu konten biasa diikuti oleh konten terenkripsi dan diikuti oleh konten biasa lagi. Cara pembacaan per baris yang dilakukan adalah sebagai berikut.

Konten di atas dibagi menjadi beberapa kelompok sesuai jenisnya dan kelompok ini diletakkan dalam suatu deret sesuai dengan urutan pada konten tersebut. Kelompok konten ini memiliki dua atribut, yaitu TYPE dan CONTENT. TYPE menyatakan jenis konten dengan aturan nilai 1 untuk jenis konten terenkripsi dan nilai 0 untuk jenisk konten biasa. CONTENT menyatakan isi dari konten itu sendiri dalam bentuk *string*. Jika contoh konten di atas dibaca maka jumlah kelompok pesan ada tiga dengan urutan seperti yang tertera pada tabel berikut ini.

**Tabel 1 Tabel Pengelompokan Contoh Konten**

| No Urut | TYPE | CONTENT |
|---|---|---|
| 1 | 0 | `The secret message is below` |
| 2 | 1 | `34cf9bd6e25efd9a80b96dc0b44e1609` |
| 3 | 0 | `This is the end of message` |

Setelah pembagian kelompok selesai dilakukan, proses dekripsi dapat dijalankan hanya kepada CONTENT kelompok dengan TYPE sama dengan 1. Kemudian CONTENT dari semua kelompok digabungkan menjadi satu. Dengan cara ini semantik konten tetap terjaga. Hasil yang diharapkan muncul dari cara seperti ini adalah sebagai berikut.

```
The secret message is below
--
Hello World
--
This is the end of message
```

## 3.5 Analisis Kebutuhan dan Perancangan

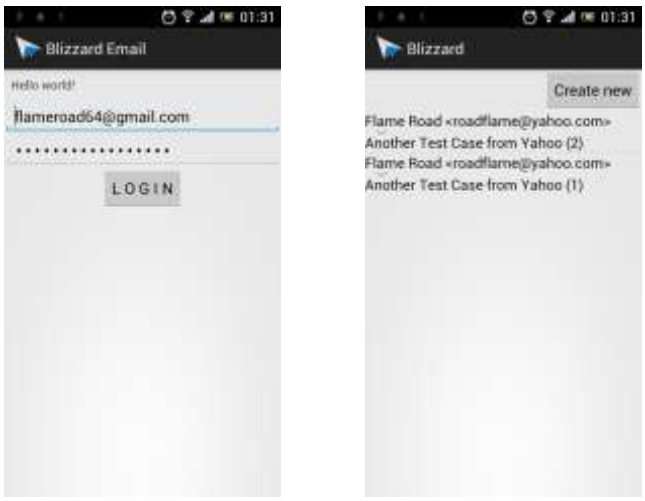
Aplikasi klien surel yang akan dibangun harus memenuhi beberapa spesifikasi, yaitu: aplikasi harus dapat membuat konten surel terenkripsi; aplikasi dapat membuat dan mengirim surel; aplikasi harus dapat menerima dan membaca surel.

# 4. IMPLEMENTASI

Implementasi dilakukan pada komputer jinjing dengan spesifikasi yaitu: prosesor Core i3 2.27 GHz, 4 GB RAM, dan 320 GB HDD. Perangkat lunak yang digunakan pada saat implementasi adalah sebagai berikut: Microsoft Windows 7 Ultimate 64-bit, Microsoft Word 2010, NetBeans 7.1, Eclipse 3.6 dengan Android Plugin, Android SDK rev. 20, Java Development Kit (JDK) 1.6, dan Java Runtime Environment 16.

Pengujian dilakukan pada ponsel Sony Ericsson Xperia Ray dengan sistem Android. Spesifikasi ponsel adalah sebagai berikut: prosesor Qualcomm Snapdragon 1 GHz, Adreno GPU, Android 4.0.3 Ice Cream Sandwich, dan 512 MB RAM.

Aplikasi terdiri dari lima antarmuka, yaitu antarmuka *login*, kotak masuk, pembacaan konten, membuat konten rahasia, dan membuat surel. Pada *platform* Android antarmuka didefinisikan dengan berkas XML. Ilustrasi untuk masing-masing antarmuka dapat dilihat pada gambar berikut.

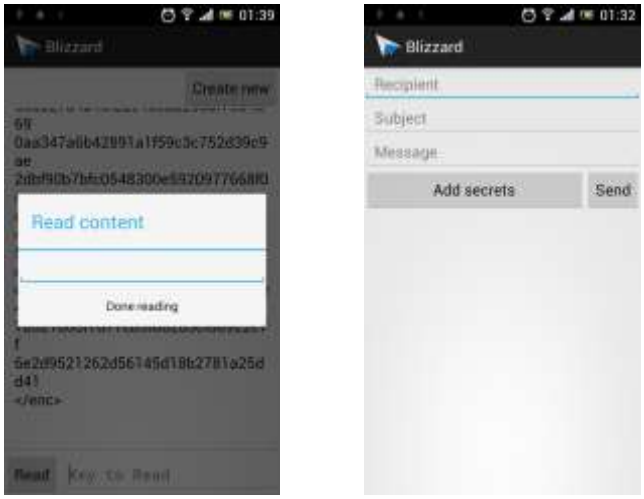



**(a)** **(b)**

**Gambar 4 a. Halaman *Login* b. Kotak Masuk**

Aplikasi terdiri dari sebelas kelas. Kelas-kelas tersebut diimplementasi dalam bahasa Java. Dengan menggunakan pustaka JavaMail, aplikasi dapat menerima *username* dan *password* dari pengguna untuk melakukan *login*. Setelah *login* selesai, aplikasi dapat menggunakan protokol IMAP untuk mengunduh surel yang kemudian ditampilkan pada halaman Kotak Masuk.

Di halaman kotak masuk pengguna dapat memilih surel yang ingin dibaca. Di halaman ini pula pengguna dapat mendekripsi pesan yang terenkripsi. Setelah itu pengguna dapat membuat surel di halaman Buat dan Kirim Surel.




**(a)** **(b)**

**Gambar 5 a. Halaman Baca Surel b. Buat dan Kirim Surel**


**Gambar 6 Halaman Buat Konten**

.

Pada halaman Buat dan Kirim surel pengguna dapat menyertakan alamat pengirim, subjek surel, dan konten surel. Konten surel dapat berupa konten yang dienkripsi atau konten biasa. Konten terenkripsi dapat dibuat pada halaman Buat Konten. Setelah pembuatan konten selesai, surel tersebut dapat dikirim. Protokol yang digunakan untuk mengirim adalah SMTP yang tersedia pada pustaka JavaMail.

# 5. PENGUJIAN

Tujuan pengujian ini adalah untuk mengetahui apakah aplikasi dapat bekerja sesuai dengan kebutuhan. Pengujian dilakukan dengan cara menggunakan aplikasi untuk mengenkripsi pesan dan mengirim pesan tersebut dengan surel dan membaca surel serta mendekripsi konten yang ada didalamnya. Selain itu surel juga disadap untuk membuktikan konten surel yang terenkripsi tetap dalam keadaan terenkripsi ketika dikirim.

Hasil pengujian dan ketercapaiannya dapat dilihat pada tabel 2 berikut ini.

**Tabel 2 Tabel Pengujian Aplikasi**

| Tujuan Pengujian | Ketercapaian |
|---|---|
| Enkripsi dan dekripsi teks | Tercapai |
| Membuat dan mengirim surel | Tercapai |
| Membaca dan mendekripsi surel | Tercapai |
| Konten surel yang dienkripsi tetap tidak terbaca ketika disadap | Tidak Tercapai |

# 6. ANALISIS

*Ciphertext* memiliki ukuran panjang yang lebih besar dibanding *plaintext*. Hal ini disebabkan oleh setiap karakter pada *plaintext* akan dikodekan menjadi dua karakter heksadesimal. Setelah *ciphertext* didekripsi hasil yang didapat adalah *plaintext* yang jika dibaca maka hasilnya adalah sama dengan teks asli tetapi ukurannya berbeda. Teks sebelum dienkripsi akan ditambahkan dengan beberapa spasi sehingga ukurannya mencapai kelipatan enam belas. Dengan demikian ukuran teks hasil dekripsi menjadi lebih panjang.

Penggunaan protokol IMAP menghasilkan surel yang sudah dibaca dapat diunduh ulang dan dibaca kembali. Berbeda dengan penggunaan protokol POP3, surel yang sudah pernah diunduh tidak akan diunduh lagi. Surel yang dapat dibaca hanya surel dengan Content-Type adalah *text/plain*. Hal ini mengakibatkan jika ada surel yang dibuat dengan peramban maka konten tersebut harus dibuat pada halaman *plain text* bukan *rich text*. Jika menggunakan *rich text* ketika surel dibaca melalui aplikasi, teks akan terbaca mentah sehingga semua *tag* HTML yang ada dalam konten akan terbaca.

Surel yang diunduh tidak disimpan di penyimpanan tetapi hanya disimpan di memori. Hal ini mengakibatkan jika aplikasi ditutup maka surel yang sudah diunduh hilang. Surel yang diunduh adalah semua surel yang ada di kotak masuk. Surel yang ada di kotak masuk dapat berjumlah sedikit atau banyak. Jika surel yang ada di kotak masuk terlalu banyak maka ponsel yang menerima surel akan *crash*. Dengan demikian jumlah surel yang dapat diunduh masih terbatas dan perlu dibuat mekanisme untuk mengatur pengunduhan surel sehingga surel yang diunduh tidak terlalu banyak.

Pengiriman yang dilakukan melalui akun surel Google Mail dengan protokol SMTP harus menggunakan jalur yang *secure*. Google Mail tidak menyediakan layanan SMTP yang tidak *secure* sehingga jika pengiriman dengan JavaMail tidak menggunakan TLS, protokol yang digunakan harus menggunakan SSL.

Penyadapan surel gagal dilakukan karena pengiriman surel dengan JavaMail menggunakan TLS (*transport layer security*). Protokol ini bertindak sebagai berikut. Jika setelah jalur komunikasi antar dua pihak berhasil dibuat maka jalur tersebut akan diubah menjadi jalur yang aman (*secure*) sehingga surel

tidak bisa disadap. Cara lain untuk melihat apakah konten tetap dalam keadaan terenkripsi adalah dengan membaca *log* perintah SMTP yang muncul saat transaksi surel. Pada protokol SMTP, perintah DATA diikuti dengan pengiriman teks berupa pesan surel. Dengan membaca teks-teks yang dikirim setelah perintah DATA, dapat diketahui bahwa teks-teks yang merupakan konten terenkripsi masih tetap pada keadaan terenkripsi.

## 7. SIMPULAN DAN SARAN

Pengerjaan tugas akhir ini memberikan simpulan sebagai berikut:

1. Penerapan algoritma Rabbit pada aplikasi klien surel dilakukan terpisah sehingga modul enkripsi-dekripsi dibuat berbeda dengan modul yang berhubungan dengan surel. Pembangunan dilakukan dengan bahasa Java.
2. Untuk membuat konten terenkripsi, aplikasi menyediakan halaman khusus untuk membuat konten ini dan konten ini kemudian dapat digabungkan dengan konten surel yang tidak dienkripsi.
3. Penggabungan konten surel dilakukan dengan cara meletakkan konten surel yang terenkripsi di antara dua buah label atau *tag*, yaitu label "<enc>" dan "</enc>". Setelah pesan selesai disiapkan, surel dapat dikirim. Pengiriman surel dilakukan dengan protokol SMTP.
4. Surel diterima dengan cara diunduh dari internet dengan menggunakan protokol IMAP dan kemudian ditampilkan di kotak masuk pada ponsel. Untuk membaca surel secara lengkap, pembaca harus menyertakan kunci dekripsi sehingga pesan yang dienkripsi dapat dibaca dan konten surel dapat dibaca dengan lengkap.

Aplikasi hasil pengembangan ini masih memiliki beberapa kekurangan. Untuk melengkapi kekurangan-kekurangan ini, ada beberapa saran yang dapat diberikan, yaitu:

1. Konten surel yang dibaca hendaknya tidak hanya berjenis *text/plain* saja.
2. Penyedia layanan surel diperluas sehingga pengguna layanan surel selain Google Mail dapat menggunakan aplikasi ini.
3. Enkripsi dan dekripsi hendaknya dapat dilakukan tidak hanya pada teks tetapi juga pada suara, gambar, dll.
4. Ada pengaturan jumlah surel yang diunduh sehingga jika surelnya terlalu banyak maka aplikasi tidak berhenti tiba-tiba.
5. Jumlah alamat surel yang dituju hendaknya dapat lebih dari satu.

## 8. REFERENCES


[1] Boesgaard, Martin., Pedersen, Thomas., Vesterager, Mette. (2004): The Rabbit Stream Cipher – Design and Security Analysis, http://eprint.iacr.org/2004/291.pdf, diturunkan pada 2 Desember 2011.

[2] Bryan, Randy. (2011): Top 4 Advantages of Android over the iPhone, http://randybryan.com/?p=671, Download(diturunkan, diunduh) pada 7 November 2011.

[3] Crispin, M. (2003): Internet Message Access Protocol - Version 4rev1, http://tools.ietf.org/html/rfc3501, diturunkan pada 29 Juli 2012.

[4] KetuWare. (2004) : Symmectic vs. Asymmectic Encryption, http://www.ketufile.com/Symmetric_vs_Asymmetric_Encryption.pdf, diunduh pada 25 Oktober 2011.

[5] Partige, Craig. (2009) : The Technical Development of Internet Email, http://www.ir.bbn.com/~craig/email.pdf, diunduh pada 9 November 2011.

[6] Postel, J.B. (1982): Simple Main Transfer Protocol, http://tools.ietf.org/html/rfc821, diturunkan pada 29 Juli 2012.

[7] Titlow, J.P. (2011): Android, the Fastest Growing Smartphone OS in Europe, Zooms Past iPhone, http://www.readwriteweb.com/archives/android_european_marketshare_beats_iphone.php, diturunkan pada 3 November 2011.

[8] Turner, David (2009) : Turner, David (2009) : Introducing Android 1.5 NDK, Release 1, http://android-developers.blogspot.com/2009/06/introducing-android-15-ndk-release-1.html, diturunkan pada 6 Desember 2011.