PENERAPAN ENKRIPSI RABBIT STREAM CIPHER ALGORITHM UNTUK MENGAMANKAN FILE BERSIFAT RAHASIA DI POLSEK MANGKUBMI TASIKMALAYA

Muhamad Septian[1], Angga Setiyadi, S.Kom., M.Kom[2]
[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : me@mseptian.id[1], angga.setiyadi@email.unikom.ac.id[2]

ABSTRAK

Kebocoran informasi adalah salah satu masalah yang terjadi di dalam instansi pemerintahan terutama didalam kepolisian. Salah satu pencegahannya adalah dengan cara mengenkripsi informasi tersebut. Enkripsi sudah di gunakan oleh tentara Spartan pada tahun 46 hingga 120 masehi yang bertujuan untuk mengirim pesan tersembunyi ketika perang berlangsung. Dimasa sekarang ini enkripsi telah mengalami perkembangan pesat, salah satunya adalah *rabbit stream chipher*. *Rabbit stream cipher* pertama kali diperkenalkan kepada umum pada tahun 2003. Rabbit Stream Cipher menggunakan kunci 128 bit kunci rahasia dan 64 bit *Initialization Vector* (IV) sebagai masukan untuk digunakan dalam blok keluaran yang terdiri dari 128 bit acak semu (*pseudo-random*). Di dalam makalah ini akan di bahas mengenai *rabbit stream cipher*, implementasi kedalam program serta performansi dari *rabbit stream cipher* itu sendiri.

Kata kunci : enkripsi, dekripsi, keamanan informasi, *rabbit stream cipher*, kriptografi.

1. PENDAHULUAN

Kepolisian Republik Indonesia adalah suatu badan keamanan negara Indonesia yang mempunyai tugas yaitu memelihara keamanan dan ketertiban masyarakat, menegakkan hukum, memberikan perlindungan, pengayoman, dan pelayanan kepada masyarakat. Salah satu badan yang mempunyai tingkat kerahasiaan yang tinggi maka dari itu Kepolisian Republik Indonesia sangat berhati-hati dalam mengamankan seluruh arsip dan data miliknya. Dengan Polres dan Polsek yang berada di setiap wilayah Indonesia menjadikan Kepolisian adalah salah satu tempat yang sangat sensitif untuk urusan pentimpanan data.

Metode enkripsi adalah proses mengacak data atau informasi sehingga tidak dapat dibaca oleh pihak lain atau pihak yang tidak berhak menerima, kecuali untuk pihak yang berhak menerima serta dengan adanya metode enkripsi ini diharapkan dapat mencegah campur tangan dari orang – orang yang tidak berhak menghapus data atau informasi yang telah diterima. Informasi ini dibagi menjadi dua yakni informasi yang bersifat pribadi dan informasi yang bersifat umum. Informasi yang bersifat umum adalah informasi yang boleh diketahui oleh orang banyak atau sifatnya umum. Alur dari proses pengiriman sebuah data atau informasi tidak luput dari gangguan – gangguan dari pihak yang tidak berhak. Salah satu teknik untuk menjaga kerahasiaan data atau informasi adalah dengan menggunakan algoritma kriptografi. Salah satu cara untuk mengatasi masalah tersebut adalah dengan menggunakan enkripsi data. Yang diharapkan mengamankan data bersifat rahasia agar tidak ditemukannya penyampaian informasi bersifat rahasia kepada publik.

Ditemukannya penyimpanan dan penyebaran informasi bersifat rahasia kepada publik secara tidak terenkripsi adalah masalah yang dialami oleh Polsek Mangkubumi Tasikmalaya saat ini yang dikhawatirkan dapat menjadi celah bocornya informasi bersifat rahasia kepada publik yang dapat berakibat terhadap kinerja Polsek Mangkubumi Tasikmalaya Kota. Selain itu tidak terenkripsinya file yang disebarluaskan kedalam internal kepolisian turut menjadi masalah didalam polsek Mengkubumi Tasikmalaya.

2. Penerapan Enkripsi Dekripsi

2.1 Metode Penelitian

Metode penelitian yang digunakan makalah ini adalah Studi pustaka yaitu mengumpulkan refrensi baik dari makalah, artikel, internet maupun buku mengenai algoritma *stream cipher* serta pemograman untuk membuat aplikasinya.

Dalam tahap ini hasil analisis dibuat pemodelan sistem, menggunakan struktur program dan algoritma, rancangan antar muka dibuat untuk mempermudah sistem alur yang terjadi ketika pembuatan program dan analisa program. Dalam makalah ini penulis menggunakan Bahasa pemograman java untuk mempermudahkan pembuatan aplikasi pada *platform desktop*.

2.2 Analisis Algoritma Enkripsi Dekripsi

Menurut [1] dan [2], Enkripsi merupakan proses perubahan data asli (plaintext) menjadi ciphertext (data yang tidak dapat dimengerti) sedangkan Dekripsi kebalikan dari enkripsi yaitu proses pengembalian bentuk ciphertext menjadi plaintext kembali sehingga bisa dipahami. Enkripsi dan

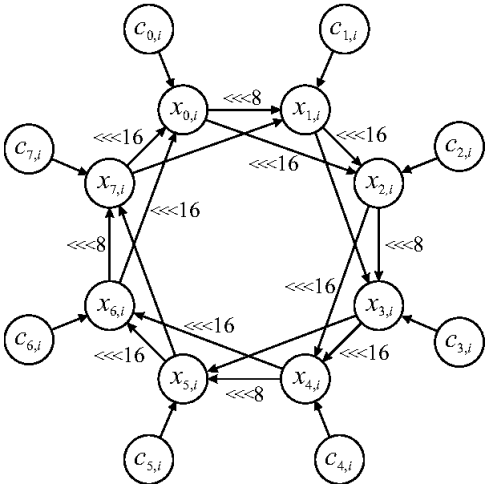
Dekripsi dilakukan menggunakan kunci yang sudah ditentukan. *Rabbit Stream Cipher* merupakan salah satu algoritma enkripsi dengan kunci simetris.

Rabbit merupakan salah satu dari sekian banyak algoritma cipher yang diperkenalkan pada tahun 2003. Algoritma Rabbit menggunakan 128 bit kunci rahasia atau sering di sebut KEY dan 64 bit Initialization Vector (IV) sebagai masukan untuk membangkitkan blok-blok keluaran yang terdiri dari 128 bit yang diacak semu (pseudo-random) yang merupakan kombinasi dari bit-bit pada status internal, untuk setiap perulangan.

Proses enkripsi/dekripsi dilakukan dengan mengXOR-kan blok acak semu tersebut dengan plainteks/cipherteks. Ukuran dari status internal adalah 513 bit dibagi menjadi 8 variabel status dengan panjang 32 bit, 8 counter dengan panjang 32 bit, dan 1 bit carry untuk counter. Kedelapan variabel status diupdate dengan 8 buah fungsi non-linear.

Rabbit dirancang sehingga lebih cepat daripada algoritma kriptografi yang biasa digunakan, dan menggunakan kunci berukuran 128 bit untuk mengenkripsi plainteks dengan jumlah blok hingga 2^{64}. Sehingga untuk mengetahui plainteks tanpa memiliki kunci hanya dapat dilakukan dengan pencarian exhaustive, yaitu dengan 2^{128} kemungkinan kunci.

Inti dari algoritma rabbit stream cipher adalah fungsi *nextstate* yang mengacak masukan bit menjadi keluaran bit semu yang disimpan untuk di jadikan hasil dari dekripsi. Berikut adalah gambar ilustrasi dari fungsi nextstate yang bisa di lihat pada gambar 1.



Gambar 1. Ilustrasi gambar dari fungsi *Next-state*

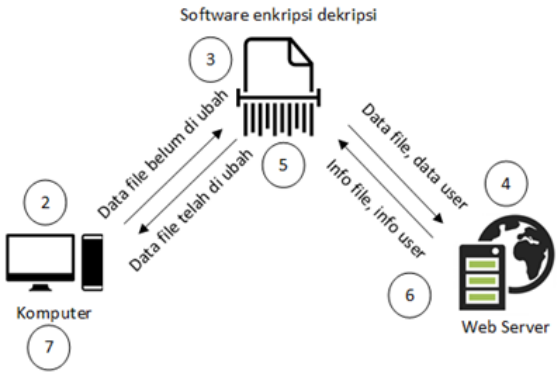
Kunci utama dari sebuah Algoritma Rabbit Stream Cipher ini adalah fungsi Next-State. Fungsi Next State berfungsi Ini didasarkan pada fungsi g, yang mengubah dua input 32-bit menjadi satu output 32-bit yang mana fungsi Next-State dapat digambarkan sebagai berikut:

Tabel 1. Tabel parameter pengujian algoritma

```
procedure nextState()
Kamus:
t: longint
i: integer
G,C,X : array[1..8] of integer
algoritma:
for i ← 0 to 7 do
    t ← (C[i] & 0xFFFFFFFFL) + (A[i] & 0xFFFFFFFFL) + b;
    b ← (byte) (t >>> 32);
    C[i] ← (int) (t & 0xFFFFFFFF);
endfor
for i ← 0 to 7 do
     t ← (X[i] + C[i]) & 0xFFFFFFFFL;
     G[i] ← (int) ((t *= t) ^ (t >>> 32));
endfor
for i ← 0 to 7 do
     X[i] ← G[i]+rtl(G[(i + 7)% 8], (i+1&1) + 1<<3) + ((i&1) == 0 ? rtl(G[(i+6)%8],16) : G[(i+6)%8]);
endfor
endprocedure
```

2.3 Alur Kerja Sistem yang diusulkan

Sistem yang akan dibangun bertujuan untuk memudahkan pihak Polsek Mangkubumi Tasikmalaya dalam mengamankan File. Adapun cara kerja sistem untuk proses enkripsi dan penyimpanan file terbagi menjadi dua, yaitu alur kerja sistem pada komputer yang bertujuan untuk mengenkripsi dan dekripsi sekaligus untuk mengunggah kedalam web server, dan alur kerja sistem pada smartphone yang bertujuan untuk mengambil list yang terdapat pada web server. Berikut adalah alur kerja sistem yang bertujuan mengenkripsi serta mengdekripsi file. pada computer yang akan dibangun adalah sebagai berikut pada gambar 2.



Gambar 2. Gambaran Sistem proses enkripsi dan penyimpanan file pada komputer

Berdasarkan pada gambar 2 dapat dijelaskan alur kerja sistem sebagai berikut:

1. Anggota kepolisian menggunakan computer yang sudah terinstall software enkripsi dekripsi file.

2. Komputer menampilkan form login yang mengharuskan anggota kepolisian mengisi field NRP dan password.

3. Software Enkripsi Dekripsi File Kepolisian mengirim data user ke web server untuk di validasi.

4. Web Server melakukan pengecekan terhadap inputan user yang diterima.

5. Jika data user yang dimasukkan anggota kepolisian sesuai maka Software Enkripsi Dekripsi File Kepolisian memproses enkripsi.

6. File hasil enkripsi lalu dikirim ke web service untuk disimpan kedalam server.

7. Komputer juga menyimpan File hasil Enkripsi dari Software Enkripsi Dekripsi File Kepolisian.

2.4 Implementasi Algoritma Enkripsi Dekripsi

Implementasi Algoritma Rabbit Stream Cipher akan digunakan pada fungsi utama dari aplikasi Enkripsi Dekripsi File Kepolisian. Algoritma Rabbit Stream Cipher Menggunakan Bahasa Pemograman Java yang akan di gunakan pada device destop dan smartphone. Berikut adalah Rabbit.java yang dijadikan fungsi utama dari program Enkripsi Dekripsi File Kepolisan Polsek Mangkubumi. Berikut adalah source code Rabbit.java

Tabel 2. Source Code Rabbit.java

```
import java.util.Arrays;

public class Rabbit {
   private static final int KEYSTREAM_LENGTH = 16;
   private static final int[] A = new int[]
         {
               0x4D34D34D, 0xD34D34D3, 0x34D34D34, 0x4D34D34D, 0xD34D34D3, 0x34D34D34, 0x4D34D34D,
               0xD34D34D3
         };

   private static final int rotl(final int value, final int shift)
   {
      return value << shift | value >>> 32 - shift;
   }

   private final int[] X = new int[8];
   private final int[] C = new int[8];
   private byte b;
   private int keyindex = 0;
   private byte[] keystream = null;

   public Rabbit()
   {
      b = 0;
   }

   public char decryptChar(byte byt)
   {
      byte [] b = {byt};
      //hack to get char from encrypted byte(array of lenght 1)
      return (new String(crypt(b))).charAt(0);
   }

   /**
    * Should be fed an array with a length that is a multiple of 16 for proper
    * key sequencing.
    */
   public byte[] crypt(final byte[] message)
   {
      int index = 0;
      while (index < message.length)
      {
         if (keystream == null || keyindex == KEYSTREAM_LENGTH)
         {
            keystream = keyStream();
            keyindex = 0;
         }
         for (; keyindex < KEYSTREAM_LENGTH && index < message.length; ++keyindex)
         {
            message[index++] ^= keystream[keyindex];
         }
      }
      return message;
   }

   /**
    * returns 16 bytes
    */
   private byte[] keyStream()
   {
      nextState();
      final byte[] s = new byte[16];
         /*
          * unroll
          */
      int x = X[6] ^ X[3] >>> 16 ^ X[1] << 16;
      s[0] = (byte) (x >>> 24);
      s[1] = (byte) (x >> 16);
      s[2] = (byte) (x >> 8);
      s[3] = (byte) x;
      x = X[4] ^ X[1] >>> 16 ^ X[7] << 16;
      s[4] = (byte) (x >>> 24);
      s[5] = (byte) (x >> 16);
      s[6] = (byte) (x >> 8);
      s[7] = (byte) x;
      x = X[2] ^ X[7] >>> 16 ^ X[5] << 16;
```

```
        s[8] = (byte) (x >>> 24);
        s[9] = (byte) (x >> 16);
        s[10] = (byte) (x >> 8);
        s[11] = (byte) x;
        x = X[0] ^ X[5] >>> 16 ^ X[3] << 16;
        s[12] = (byte) (x >>> 24);
        s[13] = (byte) (x >> 16);
        s[14] = (byte) (x >> 8);
        s[15] = (byte) x;
        return s;
    }

    private void nextState()
    {
            /*
             * counter update
             */
        for (int j = 0; j < 8; ++j)
        {
            final long t = (C[j] & 0xFFFFFFFFL) +
(A[j] & 0xFFFFFFFFL) + b;
            b = (byte) (t >>> 32);
            C[j] = (int) (t & 0xFFFFFFFF);
        }
            /*
             * next state function
             */
        final int G[] = new int[8];
        for (int j = 0; j < 8; ++j)
        {
            // TODO: reduce this to use 32 bits only
            long t = X[j] + C[j] & 0xFFFFFFFFL;
            G[j] = (int) ((t *= t) ^ t >>> 32);
        }
            /*
             * unroll
             */
        X[0] = G[0] + rotl(G[7], 16) + rotl(G[6], 16);
        X[1] = G[1] + rotl(G[0], 8) + G[7];
        X[2] = G[2] + rotl(G[1], 16) + rotl(G[0], 16);
        X[3] = G[3] + rotl(G[2], 8) + G[1];
        X[4] = G[4] + rotl(G[3], 16) + rotl(G[2], 16);
        X[5] = G[5] + rotl(G[4], 8) + G[3];
        X[6] = G[6] + rotl(G[5], 16) + rotl(G[4], 16);
        X[7] = G[7] + rotl(G[6], 8) + G[5];
    }

    /**
     * Clears all internal data. You must set the key
again to use this cypher.
     */
    public void reset()
    {
        b = 0;
        keyindex = 0;
        keystream = null;
        Arrays.fill(X, 0);
        Arrays.fill(C, 0);
    }
```

```
    /**
     * @param IV An array of 8 bytes
     */
    public void setupIV(final byte[] IV)
    {
        short[] sIV = new short[IV.length >> 1];
        for (int i = 0; i < sIV.length; ++i)
        {
            sIV[i] = (short) ((IV[i << 1] << 8) | IV[(2
<< 1) + 1]);
        }
        setupIV(sIV);
    }

    /**
     * @param iv array of 4 short values
     */
    public void setupIV(final short[] iv)
    {
            /*
             * unroll
             */
        C[0] ^= iv[1] << 16 | iv[0] & 0xFFFF;
        C[1] ^= iv[3] << 16 | iv[1] & 0xFFFF;
        C[2] ^= iv[3] << 16 | iv[2] & 0xFFFF;
        C[3] ^= iv[2] << 16 | iv[0] & 0xFFFF;
        C[4] ^= iv[1] << 16 | iv[0] & 0xFFFF;
        C[5] ^= iv[3] << 16 | iv[1] & 0xFFFF;
        C[6] ^= iv[3] << 16 | iv[2] & 0xFFFF;
        C[7] ^= iv[2] << 16 | iv[0] & 0xFFFF;
        nextState();
        nextState();
        nextState();
        nextState();
    }

    /**
     * @param key An array of 16 bytes
     */
    public void setupKey(final byte[] key)
    {
        short[] sKey = new short[key.length >> 1];
        for (int i = 0; i < sKey.length; ++i)
        {
            sKey[i] = (short) ((key[i << 1] << 8) |
key[(2 << 1) + 1]);
        }
        setupKey(sKey);
    }

    /**
     * @param key An array of 8 short values
     */
    public void setupKey(final short[] key)
    {
            /*
             * unroll
             */
        X[0] = key[1] << 16 | key[0] & 0xFFFF;
        X[1] = key[6] << 16 | key[5] & 0xFFFF;
```

```
        X[2] = key[3] << 16 | key[2] & 0xFFFF;
        X[3] = key[0] << 16 | key[7] & 0xFFFF;
        X[4] = key[5] << 16 | key[4] & 0xFFFF;
        X[5] = key[2] << 16 | key[1] & 0xFFFF;
        X[6] = key[7] << 16 | key[6] & 0xFFFF;
        X[7] = key[4] << 16 | key[3] & 0xFFFF;
            /*
             * unroll
             */
        C[0] = key[4] << 16 | key[5] & 0xFFFF;
        C[1] = key[1] << 16 | key[2] & 0xFFFF;
        C[2] = key[6] << 16 | key[7] & 0xFFFF;
        C[3] = key[3] << 16 | key[4] & 0xFFFF;
        C[4] = key[0] << 16 | key[1] & 0xFFFF;
        C[5] = key[5] << 16 | key[6] & 0xFFFF;
        C[6] = key[2] << 16 | key[3] & 0xFFFF;
        C[7] = key[7] << 16 | key[0] & 0xFFFF;
        nextState();
        nextState();
        nextState();
        nextState();
            /*
             * unroll
             */
        C[0] ^= X[4];
        C[1] ^= X[5];
        C[2] ^= X[6];
        C[3] ^= X[7];
        C[4] ^= X[0];
        C[5] ^= X[1];
        C[6] ^= X[2];
        C[7] ^= X[3];
    }
}
```

2.5 Implementasi Antar muka

Implementasi antarmuka berisi pemaparan dari setiap antarmuka yang telah diarancang pada tahap analisis. Implementasi antarmuka pada pembangunan sistem enkripsi dekripsi file kepolisian polsek mangkubumi tasikmalaya terdiri dari antarmuka desktop dan antarmuka smartphone. Adapun implementasi anatarmuka pada platform desktop dapat dilihat sebagai berikut. Implementasi Antarmuka platform desktop dapat dilihat pada tabel 4.3 sebagai berikut.

Tabel 3. Implementasi Antar Muka

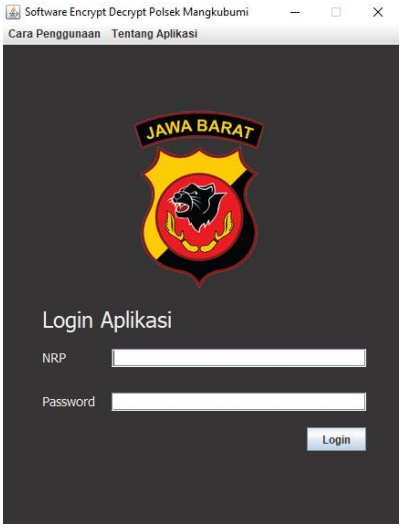
Menu	Deskripsi	Proses	File
Halaman Awal	Halaman Awal Aplikasi Enkripsi File	Menampilkan Halaman Utama	MainJFrame.form
Halaman Enkripsi	Halaman yang digunakan untuk mengenkripsi file	Enkripsi File	EncryptJFrame.form
Halaman Berhasil Enkripsi	Halaman yang digunakan untuk menampilkan notifikasi bahwa berhasil enkripsi	Notifikasi Enkripsi File	EncryptSuccess.form
Halaman Dekripsi	Halaman yang digunakan untuk mengdekripsi file	Dekripsi File	DecryptJFrame.form
Halaman Berhasil Dekripsi	Halaman yang digunakan untuk menampilkan notifikasi bahwa berhasil dekripsi	Notifikasi Dekripsi File	DecryptSuccess.form

2.6 Implementasi Program

Implementasi Program berisi hasil percobaan menjalankan program yang sudah di buat. Dengan mengimplemtasikan algoritma serta antar muka maka dibuatlah program untuk ujicoba enkripsi dan dekripsi file. Berikut adalah langkah-langkah enkripsi dan dekripsi file. Untuk langkah pertama pengguna diwajibkan login untuk bias menggunakan aplikasi enkripsi dekripsi berikut adalah langkah login.

Tabel 4. Percobaan Login

NRP	12345678
Password	12345678



Gambar 3. Halaman Login

Selanjutnya pengguna akan di tampilkan halaman home yang berisi dua tombol. Masing-masing tombol berguna untuk proses enkripsi dan dekripsi. Jika pengguna memilih enkripsi maka program akan menampilkan halaman enkripsi seperti gambar 4. Lalu pengguna bisa memasukan field inputan seperti pilih file yang akan di enkripsi, pilih folder hasil dan password atau key yang di gunakan untuk enkirpsi.

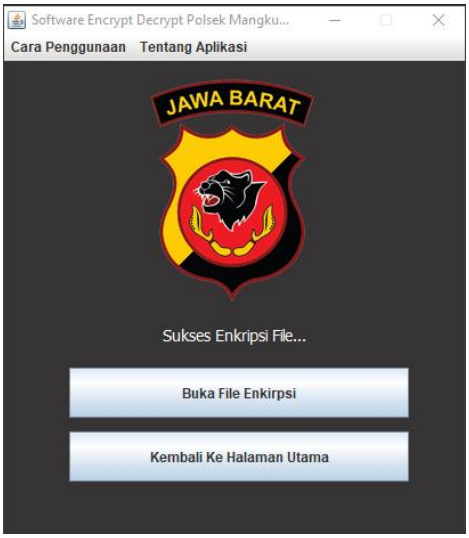
Tabel 5. Percobaan Enkripsi

Pilih File	Surat tugas.docx
Folder Output	Documents
Password	M4ngkubum1



Gambar 4. Tampilan Halaman Enkripsi

Setelah berhasil enkripsi maka pengguna akan di tampilkan halaman berhasil enkripsi yang bisa dilihat pada gambar 5.



Gambar 5. Tampilan Halaman Berhasil Enkripsi

3. PENUTUP

Dalam hal keamanan pada program enkripsi dekripsi file kepolisian, algoritma Rabbit sangat baik untuk diimplementasikan, karena hingga saat ini belum ada serangan terhadap algoritma Rabbit yang lebih cepat daripada pencarian exhaustive. Sedangkan pencarian exhaustive terhadap algoritma Rabbit adalah pencarian terhadap 2^{128} kemungkinan kunci, karena panjang kunci yang digunakan adalah 128 bit. Algoritma Rabbit dapat digunakan untuk mengenkripsi plainteks dengan ukuran hingga mencapai 2^{64} byte. Dalam hal kecepatan dan ukuran hasil keluaran, algoritma Rabbit dapat dikatakan lebih cepat daripada algoritma cipher aliran yang biasa digunakan dan lebih irit untuk pentimpanan dikarenakan inputan dan keluaran menghasilkan ukuran yang sama.

Saran yang dapat penulis sampaikan bahwasannya lebih baik algoritma rabbit stream cipher dipadukan dengan algoritma kompresi agar ukuran dari enkripsi semakin kecil. Dan sebaiknya bias di gabungkan dengan algoritma enkripsi yang lainnya.

DAFTAR PUSTAKA

[1] Ariyus, Doni, Pengantar Ilmu Kriptografi Teori, Analisis dan Implementasi, Penerbit Andi, 2008.

[2] Munir, Rinaldi., Kriptografi, Informatika, 2007.

[3] Boesgaard, Martin, et al, The Rabbit Stream Cipher – Design and Security Analysis, Cryptico, 2003.

[4] Boesgard, Martin, Mette Vesterager, Thomas Pedersen, Jesper Christiansen, and Ove Scavenius. Rabbit: A New High-Performance Stream Cipher. Cryptico A/S. 2003. http://www.cryptico.com/. 8 Agustus 2018 pukul 15:00.

[5] Suryana, Taryana. "Pengelolaan Dokumen sebagai Sarana Komunikasi Internal UNIKOM.". Jurnal Komputer dan Informatika (KOMPUTA), 2012.

[6] Paar, Prof. Dr.-Ing. Christof., Pelzl, Dr.-Ing. Jan., Understanding Cryptography: A Textbook for Students and Practitioners, Springer-Verlag, Berlin, 2010.

[7] Boesgaard, Martin, et al, The Rabbit Stream Cipher – Design and Security Analysis, Cryptico, 2003.

[8] Binanto, Iwan. 2005. Konsep Dasar Program. Jakarta: PT. Elex Media. Komputindo. Indrajani. 2007

[9] Jogiyanto. 2005. Analisis dan Desain Sistem Informasi. Yogyakarta: Penerbit Andi. Juju, Dominikus. 2006

[10] Wahana Komputer, The Best 40 Java Aplications, Penerbit PT. Gramedia Pustaka Utama, Jakarta, 2010.