Jurnal Strategi
Volume 3 Nomor 1 Mei 2021

Perancangan Aplikasi Enkripsi Messenger dengan Menggunakan Metode RC-6 Berbasis Android

Sehat Alprianto S.[#1], Hapnes Toba[*2]

[#]*Program Studi Teknik Informatika, Universitas Kristen Maranatha*
Jl.Prof. drg. Surya Sumantri No.65, Bandung
[1]sehatsitomvul@gmail.com
[2]hapnestoba@it.maranatha.edu

Abstract — **Current technological developments allow humans to communicate and exchange information remotely. Along with that, the demand for the confidentiality of information exchanged is increasing. Therefore, a branch of science was developed that studies ways of securing data or better known as cryptography. The RC6 algorithm is included in cryptography that has strong security and includes symmetric cryptography (has the same key when encryption and decryption). This application performs cryptography on text in the form of letters, the results of this study are in the form of an Android-based chat application that can send messages that have been encrypted using the RC6 algorithm, so that the confidentiality of these messages can be maintained.**

Keywords— ***Firebase; RC6; Cryptography; Android; Chat.***

I. PENDAHULUAN

Memasuki era teknologi saat ini, kebutuhan akan fasilitas penyebaran media informasi yang cepat dan mudah semakin meningkat terutama pada aplikasi Android. Salah satu fasilitas media komunikasi sekaligus tempat bertukarnya informasi yang paling populer dan paling banyak digunakan masyarakat adalah *chat*. Aplikasi *chatting* menjadi populer karena digunakan setiap hari oleh pengguna. Saat ini banyak pengguna smartphone yang lebih memilih menggunakan aplikasi chatting dibandingkan SMS, ini dikarenakan murahnya tarif internet yang ditawarkan berbagai operator, kecepatan menerima pesan pada aplikasi chatting dibandingkan sms yang terkadang memiliki gangguan jaringan sehingga pesan yang dikirim tidak bisa langsung sampai ke tujuan.

Di samping berbagai kemudahan yang diberikan aplikasi *chatting*, masih terdapat permasalahan di antaranya mengenai keamanan. Keamanan telah menjadi aspek yang sangat penting dari suatu sistem informasi. Sebuah informasi umumnya hanya ditujukan bagi segolongan individu atau komunitas tertentu. Oleh karena itu sangat penting untuk mencegah jatuh kepada pihak-pihak lain yang tidak berkepentingan, maka pada akhirnya orang-orang pun mengembangkan berbagai cara untuk mengatasi persoalan keamanan data agar orang-orang yang tidak berhak tidak mungkin dapat membaca atau bahkan merusak data yang bukan ditujukan kepadanya.

Salah satu cara untuk melindungi data adalah dengan teknik enkripsi (*encryption*), yaitu sebuah proses yang melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti, sehingga data yang sudah dienkripsi tersebut tidak akan dapat dimanipulasi oleh orang yang tidak berhak untuk mengakses data tersebut. Karena teknik enkripsi merupakan suatu sistem yang telah siap untuk diautomasi, maka teknik ini digunakan untuk mengamankan yang akan ditransmisikan.

Metode yang digunakan adalah Rivest Chiper 6 (RC-6). RivestChiper 6 (RC-6) adalah salah satu kandidat Advanced Encryption Standard (AES) yang diajukan oleh RSA Security Laboratories kepada NIST. Algoritma ini adalah pengembangan dari algoritma RC5 dan telah memenuhi semua persyaratan yang diajukan oleh NIST dan menggunakan ukuran blok hingga 128 bit. Metode enkripsi yang diterapkan adalah RC-6 untuk mendapatkan pesan yang sudah diacak atau disebut dengan chipper text yang menghasilkan pengacakan pesan yang lebih kompleks dan sulit dipecahkan. [1]

Fokus yang menjadi permasalahan utama dari penelitian ini adalah bagaimana cara membangun aplikasi *chatting* yang aman pada platform Android dengan teknologi kriptografi RC-6. Berdasarkan fokus permasalahan tersebut, tujuan dari penelitian yang dilakukan adalah untuk membangun aplikasi messenger pada platform Android yang aman dari pihak yang tidak berkepentingan dengan menggunakan teknik kriptografi RC-6, sehingga kerahasiaan data yang akan dikirim akan tetap terjaga. Manfaat penelitian adalah memberi kemudahan bagi pengguna mobile phone Android untuk saling bertukar informasi dan memberikan keamanan terhadap informasi yang dikirim sehingga pengguna tidak perlu khawatir apabila

terjadi penyadapan data dan informasi.

II. KAJIAN TEORI

A. Kriptografi

Kriptografi merupakan ilmu sekaligus seni untuk menjaga keamanan pesan. Selain itu, terdapat pemahaman tentang kriptografi, yaitu kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berkaitan dengan keamanan informasi (seperti kerahasiaan, integritas data dan verifikasi identitas). Istilah "seni" dalam definisi di atas berarti memiliki cara unik untuk menjaga kerahasiaan pesan. Kata "grafi" dalam "kriptografi" adalah seni itu sendiri. Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi ,yaitu: [2]

1. *Convidential* (kerahasiaan), yaitu memberikan kerahasiaan pesan dan menyimpan data dengan menyembunyikan data dengan menyembunyikan informasi lewat teknik-teknik enkripsi.
2. *Message integrity* (integritas data), yaitu memberikan jaminan bahwa dari setiap bagian tidak mengalami perubahan dari saat data dibuat/ dikirim sampai dengan saat data tersebut di buka.
3. *Non-repudiation* (nirpenyangkalan), yang memberikan cara untuk membuktikan bahwa suatu dokumen datang dari setiap seseorang apabila ia mencoba menyangkal memiliki dokumen tersebut.
4. *Autentication* (autentikasi), yang memberikan dua layanan. Pertama adalah untuk mengidentifikasi keaslian dari suatu pesan dan memberikan jaminan keotentikannya. Kedua, untuk menguji identitas seseorang apabila ia akan memasuki sebuah system yang ada.

1) Komponen Kriptografi

Pada dasarnya kriptografi memiliki beberapa komponen penting dalam pelaksanaannya seperti [3]:

a. Enkripsi. Dalam kriptografi, sangat penting untuk menjaga kerahasiaan data yang dikirimkan dengan cara tertentu. Enkripsi dapat diartikan sebagai kata sandi atau kode. Ketika Anda tidak mengerti sebuah kata atau kalimat, kamus akan digunakan untuk menjelaskannya. Tetapi ini berbeda dengan enkripsi, yang menggunakan algoritma yang dapat menyandikan data yang dibutuhkan untuk mengubah teks biasa menjadi kode.
b. Dekripsi. Ini kebalikan dari enkripsi, yang mengembalikan pesan terenkripsi ke bentuk aslinya. Algoritma ini juga berbeda dengan algoritma yang digunakan saat mengenkripsi data.
c. Kunci / *Key*. Merupakan parameter yang digunakan untuk melakukan proses enkripsi maupun dekripsi. Kunci hanya diketahui oleh pengirim dan penerima karena kerahasiaannya untuk mengamankan data. Biasanya berupa string atau deretan bilangan. Kunci dalam kriptografi dibagi menadi dua yaitu kunci rahasia (*private key*) dan kunci public (*public key*).
d. *Ciphertext*. Algoritma kriptografi dapat disebut juga dengan cipher, sedangkan ciphertext merupakan pesan hasil dari enkripsinya. Pesan yang ada pada *ciphertext* tidak dapat dibaca karena berupa karakter-karakter yang tidak mempunyai makna atau arti.
e. Pesan / *Plaintext*. Pesan adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain dari pesan adalah *plaintext* atau *cleartext*. Pesan yang tersimpan tidak hanya berupa *text*, tetapi juga dapat berupa citra (*image*), suara (*audio*), arsip (*document*) , maupun video.
f. *Cryptanalysis* dan kriptologi. *Cryptanalysis* adalah ilmu dan seni untuk memecahkan *ciphertext* menjadi *plaintext* tanpa mengetahui kunci atau algoritma yang digunakan. Pelaku yang melakukan *cryptanalysis* disebut kriptanalis. Sedangkan kriptologi merupakan sebuah studi mengenai kriptografi dan kriptanalisis, karena kriptografi maupun kriptanalisis keduanya saling bersangkutan dalam kriptografi [4].

2) Teknik Kriptografi

Teknik Kriptografi pada umumnya dibedakan menjadi dua yaitu *symmetric key* (kunci simetrik) dan *asymmetric key* (kunci asimetris atau kunci publik).

a. *Symmetric Algorithm*. Algoritma kriptografi simetris atau algoritma kriptografi konvensional adalah algoritma yang menggunakan kunci enkripsi yang sama dengan kunci dekripsi. Jika kunci enkripsi sama dengan kunci dekripsi maka skema tersebut disebut dengan kunci simetris. Dalam implementasinya, pengirim dan penerima harus memilih salah satu kunci yang akan digunakan untuk enkripsi atau dekripsi. Kunci yang digunakan haruslah rahasia, karena hanya pengirim dan penerima yang mengetahui kuncinya. Keunggulan dari algoritma ini adalah pesan yang dikirim dari pengguna yang berbeda memiliki kunci yang berbeda, sehingga jika pengirim dan penerima tidak mendistribusikan data, maka kerahasiaan data tersebut akan terjamin. Contoh algoritma terkenal yang menggunakan kunci simetris adalah *Data Encryption Standard* (DES), *Advance Encryption Standard* (AES), dan *TwoFish*.

b. *Asymmetric Algorithm*. Algoritma kriptografi asimetris sangatlah berbeda dengan algoritma simetris dikarenakan kunci yang digunakan pada proses enkripsi berbeda dengan proses dekripsi. Proses enkripsi memiliki kunci yang disebut Proses enkripsi memiliki kunci umum disebut public key yaitu kunci yang diketahui semua orang, namun pada proses dekripsi pada algoritma ini hanya diketahui oleh penerima, biasanya disebut private key atau kunci pribadi. Kelebihan dari algoritma ini adalah pendistribusian kunci yang lebih aman serta jumlah kunci yang lebih sedikit. Sedangkan kekurangannya ada pada pemrosesannya karena algoritmanya lebih panjang dibandingkan kunci simetris. Contoh algoritma yang memakai *asymmetric-key* adalah *Rivest-Shamir-Adleman* (RSA), *Digital Signature Algorithm* (DSA), serta El-Gamal.

B. Rivest Code (RC-6)

Algoritma RC-6 merupakan salah satu kandidat *Advanced Encryption Standard* (AES) yang diajukan oleh RSA Laboratoriest kepada NIST. Dirancang oleh Ronald L Rivest, M.J.B. Robshaw, R. Sidney dan Y.L. Yin, algoritma ini merupakan pengembangan dari algoritma sebelumnya yaitu RC5 dan telah memenuhi semua kriteria yang diajukan oleh NIST. RC6 ini berhasil menjadi finalis dan menjadi kandidat kuat untuk menjadi AES walaupun pada akhirnya algoritma ini tidak terpilih menjadi AES melainkan algoritma Rinjdael. Versi 1.1 dari RC6 mulai dipublikasikan pada tahun 1998. Dasar desain dari algoritma RC6 ini didasarkan pada pendahulunya yaitu algoritma RC5 [3].

Algoritma RC6 adalah versi yang dilengkapi dengan beberapa parameter, sehingga dituliskan sebagai RC6-w/r/b, dimana parameter w merupakan ukuran kata dalam satuan bit, r adalah bilangan bulat bukan negatif yang menunjukan banyaknya iterasi selam proses enkripsi, dan b menunjukan ukuran kunci enkripsi dalam byte. Ketika algoritma ini masuk sebagai kandidat AES, maka ditetapkan nilai parameter w = 32, r = 20 dan b bervariasi antara 16, 24, dan 32 byte [5]. Cara kerja dari algoritma RC6 adalah menggunakan 4 buah register dan menggunakan prinsip *Iterated Block Cipher* yang mengunakan iterasi, dalam algoritma ini tidak digunakan S-box. Tingkat keamanan pada algoritma ini terletak pada kekuatan rotasi yang berdasarkan data, penggunaan eksklusif OR yang bergantian, fungsi modulo dan fungsi persamaan yang menggunakan rotasi yeng tetap. Dengan menghilangkan salah satu atau beberapa aspek di atas, maka *cipher* yang dihasilkan akan menjadi lebih lemah terhadap beberapa serangan yang sudah diketemukan sebelumnya. Beberapa jenis serangan modern terhadap algoritma ini hanya dapat dilakukan secara teori tanpa praktek serangan sesungguhnya.

III. DESAIN ALGORITMA

Dalam penelitian ini, pembangunan aplikasi *messenger* pada *platform* Android yang aman dari pihak yang tidak berkepentingan dilakukan dengan menggunakan teknik kriptografi RC-6, sehingga kerahasiaan data yang akan dikirim akan tetap terjaga. Terdapat beberapa penelitian terdahulu yang menjadi referensi pendukung terkait dengan penelitian ini. . Rancangan dan proses enkripsi serta deskripsi secara utuh dapat dilihat dalam Gambar 1 dan 2.

Dalam penelitian [5] diusulkan sebuah sistem keamanan pesan dengan algoritma rivest code-6 (RC6) menggunakan Java pada *smartphone* berbasis *Android*. Dalam penelitian tersebut dikatakan bahwa algoritma RC6 adalah versi yang dilengkapi dengan beberapa parameter, sehingga dituliskan sebagai RC6-w/r/b, dimana parameter w merupakan ukuran kata dalam satuan bit, r adalah bilangan bulat bukan negatif yang menunjukan banyaknya iterasi selam proses enkripsi, dan b menunjukan ukuran kunci enkripsi dalam *byte*. Ketika algoritma ini masuk sebagai kandidat *AES*, maka ditetapkan nilai parameter w = 32, r = 20 dan b bervariasi antara 16, 24, dan 32 *byte*.

Enkripsi RC-6 akan diimplementasikan sendiri dengan memecah blok 128 bit menjadi 4 buah blok 32 bit, maka algoritma ini bekerja dengan 4 buah register 32-bit A, B, C, D. Byte yang pertama dari plainteks atau cipher teks ditempatkan pada byte A, sedangkan byte yang terakhirnya ditempatkan pada byte D. Dalam prosesnya akan didapatkan (A, B, C, D) = (B, C, D, A) yang diartikan bahwa nilai yang terletak pada sisi kanan berasal dari register di sisi kiri. Sementara itu, untuk melakukan dekripsi, hal yang harus dilakukan menerapkan algoritma yang sama dengan enkripsi, dengan tiap iterasi menggunakan sub kunci yang sama dengan yang digunakan pada saat enkripsi, hanya saja urutan sub kunci yang digunakan terbalik.

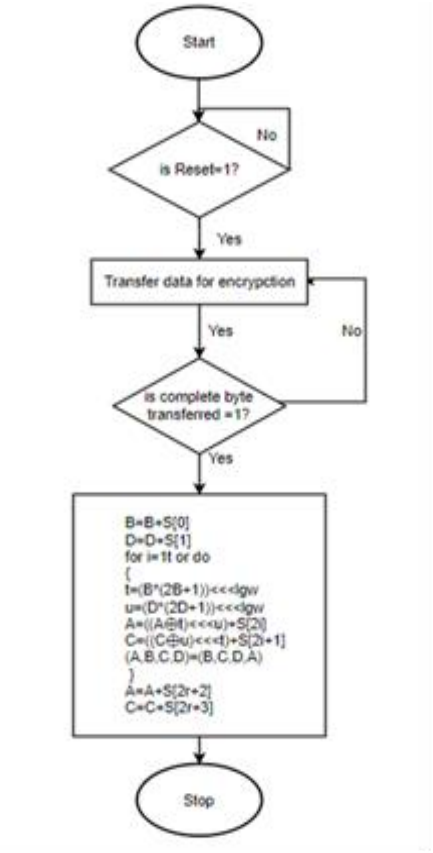
IV. IMPLEMENTASI ALGORITMA

A. Kunci

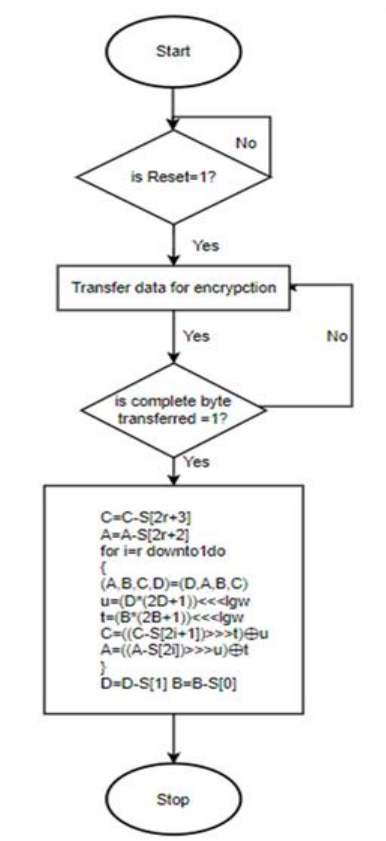
Pada tahap inisialisasi kunci didapatkan 44 kunci inisialisasi yang akan digunakan pada tahap kombinasi kunci. Kunci yang diinput pengguna yang ditampung pada array L dan inisialisasi kunci yang di tampung dalam array S. Algoritma inisialisasi kunci dapat dicontohkan secara manualnya seperti ini:

1. S[0] = Pw = b7e15163 = -1209970333
2. Qw = 9e3779b9 = -1640531527

3. S[1] = S[i - 1] + Qw penjumlahan ini dilakukan sampai 43 kali



Gambar 1. Flowchart Enkripsi RC-6



Gambar 2. Flowchart Dekripsi RC-6

B. Enkripsi RC6

Untuk mengenkripsi suatu pesan SMS dalam bentuk karakter terlebih dahulu kita merpresentasikan pesan tersebut dalam bentuk matriks yang mempunyai nilai ASCII per karakter, besar baris dan kolomnya sesuai banyaknya karakter yang akan dienkrip.

Misal: plaintext yang akan dienkrip berukuran 21 karakter

Plaint Text = TEKNIK INFORMATIK UKM

Algoritma RC6 mengola data perblok, setiap blok algoritma RC6 yang telah direvisi memiliki sebanyak 8 byte. Sehingga plaintext TEKNIK INFORMATIK UKM menjadi 2 buah blok dapat dilihat pada Tabel I.

TABEL I

KARAKTER DALAM BENTUK ASCII PADA SETIAP BLOK ALGORITMA RC6

Blok 1	**Blok 2**
84	75
69	65
75	32
78	77
73	65

Blok 1	**Blok 2**
75	82
32	65
73	78
78	65
70	84
79	72
82	65
77	0
65	0
84	0
73	0

Untuk melakukan Enkripsi ambil plaintext per8 byte data per kolomnya (blok),jika blok terakhir kurang dari 8 byte panggil function padding, sehingga blok data menjadi 8 byte. Dalam Tabel II diberikan langkah demi langkah proses enkripsi RC6 yang telah direvisi pada blok pertama dari karakter *plaintext*.

a. Tempatkan 16 byte pertama plaintext kedalam 4 buah register A dan B

70 73 75 79 77 32 77 69 84 72 79 68 73 83 84 32

TABEL II

PENEMPATAN BLOK PLAINTEXT KEDALAM REGISTER A, B, C, DAN D

Blok 1	*Binner*	*Register*
70	01000110	A 1229804627
73	01001001	
75	01001011	
79	01001111	
77	01001101	B 1128865867
32	00100000	
77	01001101	
69	01000101	
84	01010100	C 1312904008
72	01001000	
79	01001111	
68	01000100	
73	01001001	D 1396789280
83	01010011	
84	01010100	
32	00100000	

b. Lakukan langkah *whitening* awal
B = B + S[0]
1128865867 + -1204938796 = -76072929
D = D + S[1]
1396789280 + -935145979 = 461643301

c. Kemudian lakukan langkah transformasi, *mixing*, *swap register* sebanyak r = 20 iterasi. Berikut adalah langkah transformasi, *mixing*, *swap register* pada iterasi pertama.
• Transformasi
t = (B* (2B + 1)) <<< 5)
= (-76072929 * (2 * -76072929 + 1) <<< 5
= 919335968

u = (D * (2D + 1)) <<< 5)
= (461643301 * (2 * 461643301 + 1) <<< 5
= 1571822333
• *Mixing*
A = ((A ⊕ t) <<< u) + S [2]
= ((1229804627 ⊕ 919335968) <<< 1571822333) + -1495192834
= 382861068
C = ((C ⊕ u) <<< t) + S[3]
= ((1312904008 ⊕ 1571822333) <<< 919335968) + 1464888195
= 1799468344
• *Swap Register*
Sebelum di-*swap* register
A = 382861068
B = -76072929
C = 1799468344
D = 461643301
A B C D = B C D A
Setelah di-*swap* register
A = -76072929
B = 1799468344
C = 461643301
D = 382861068

Lakukan langkah transformasi, *mixing*, *swap register* sebanyak r = 20 iterasi. Sehingga register A,B,C,D menjadi
A = -1831660251
B = -291228099
C = -1080852968
D = -1430705437

d. Lakukan langkah terakhir yaitu *whitening* akhir
A = A + S[42]
-1831660251 + 1092415879 = -739244372
C = C + S[43]
-1080852968 + 1187139694 = 106286726

Proses enkripsi ini terus berlangsung hingga semua blok *plaintext* tersandikan. Sehingga didapatkanlah nilai register A dan B pada setiap blok, dalam Tabel III.

TABEL III

NILAI CHIPERTEXT PERBLOK ALGORITMA RC6

Blok	*Register*	Hex
1	A -1800259675	A5 37 B2 94
	B 1816013460	94 2A 3E 6C
	C 1672076907	6B DE A9 63
	D 2019114275	23 3D 59 78
2	A -739244372	AC 06 F0
	B - 291228099	D3 3D 36 A4
	C 106286726	EE 86 CE 55
	D -1430705437	06 E3 2A B9 AA

Setelah itu dapatlah *chipertext*-nya dalam bentuk *hexadecimal* dengan panjang 64 karakter di bawah ini:

A537B294942A3E6C6BDEA963233D5978AC06F0D33D36A4EE86CE5506E32AB9AA.

C. Dekripsi RC6

Untuk mendekripsikan *chipertext* karakter, ubah terlebih dahulu karakter dalam bentuk *hexadecimal* ke *decimal* dan tempatkan setiap 8 byte *chipertext* kedalam dua buah register A dan B sesuai aturan. Lalu lakukan langkah *whitening* akhir, iterasi, dan *whitening* awal sesuai dengan algoritma dekripsi. Proses dekripsi terus berlangsung sehingga semua blok data *chipertext* dikembalikan menjadi blok data *plaintext*.

D. Source Code Enkripsi Dan Dekripsi RC-6

Di bawah ini merupakan *source code* dari enkripsi dan dekripsi algoritma dari RC-6. Sebelum melakukan enkripsi dan dekripsi, teks yang berformat string harus di diubah menjadi byte oleh class *Converter*, dan setelah proses enkripsi dan dekripsi teks yang berformat byte akan diubah kembali menjadi *string* oleh *Converter*. Hal ini bertujuan agar class RC-6 dapat memproses enkripsi dan dekripsi pada teks pesan.

Converter.java

```
public class Converter {
    public static String static_byteArrayToString(byte[] data) {
        String res = "";
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < data.length; i++) {
            int n = (int) data[i];
            if (n < 0) {
                n += 256;
            }
            sb.append((char) n);
        }
        res = sb.toString();
        return res;
    }

    public static byte[] static_stringToByteArray(String s) {
        byte[] temp = new byte[s.length()];
        for (int i = 0; i < s.length(); i++) {
            temp[i] = (byte) s.charAt(i);
        }
        return temp;
    }
}
```

RC6.java

```
public class RC6 {

          private static int w=32, r=20;
          private static final double e = Math.E;
          private static final double goldenRatio = 1.6180339887496482;

          private static int Pw=0xb7e15163, Qw=0x9e3779b9;

          private static int[] S;

          private static int[] convBytesWords(byte[] key, int u, int c) {
                    int[] tmp = new int[c];
                    for (int i = 0; i < tmp.length; i++)
                              tmp[i] = 0;

                    for (int i = 0, off = 0; i < c; i++)
                              tmp[i] = ((key[off++] & 0xFF)) | ((key[off++] & 0xFF) << 8)
                                        | ((key[off++] & 0xFF) << 16) | ((key[off++] & 0xFF) << 24);

                    return tmp;
          }
```

```
private static int[] generateSubkeys(byte[] key) {

          int u = w / 8;
          int c = key.length / u;
          int t = 2 * r + 4;

          int[] L = convBytesWords(key, u, c);


          int[] S = new int[t];
          S[0] = Pw;
          for (int i = 1; i < t; i++)
                    S[i] = S[i - 1] + Qw;

          int A = 0;
          int B = 0;
          int k = 0, j = 0;

          int v = 3 * Math.max(c, t);

          for (int i = 0; i < v; i++) {
                    A = S[k] = rotl((S[k] + A + B), 3);
                    B = L[j] = rotl(L[j] + A + B, A + B);
                    k = (k + 1) % t;
                    j = (j + 1) % c;

          }

          return S;
}

private static int rotl(int val, int pas) {
          return (val << pas) | (val >>> (32 - pas));
}
private static int rotr(int val, int pas) {
          return (val >>> pas) | (val << (32-pas));
}

private static byte[] decryptBloc(byte[] input){
          byte[] tmp = new byte[input.length];
          int t,u;
          int aux;
          int[] data = new int[input.length/4];
          for(int i =0;i<data.length;i++)
                    data[i] = 0;
          int off = 0;
          for(int i=0;i<data.length;i++){
                    data[i] =          ((input[off++]&0xff))|
                                                  ((input[off++]&0xff) << 8) |
                                                  ((input[off++]&0xff) << 16) |
                                                  ((input[off++]&0xff) << 24);
          }


          int A = data[0],B = data[1],C = data[2],D = data[3];

          C = C - S[2*r+3];
          A = A - S[2*r+2];
          for(int i = r;i>=1;i--){
                    aux = D;
                    D = C;
                    C = B;
                    B = A;
                    A = aux;

                    u = rotl(D*(2*D+1),5);
                    t = rotl(B*(2*B + 1),5);
                    C = rotr(C-S[2*i + 1],t) ^ u;
                    A = rotr(A-S[2*i],u) ^ t;
          }
```

```
		D = D - S[1];
		B = B - S[0];

		data[0] = A;data[1] = B;data[2] = C;data[3] = D;


		for(int i = 0;i<tmp.length;i++){
			tmp[i] = (byte)((data[i/4] >>> (i%4)*8) & 0xff);
		}

		return tmp;
	}

	private static byte[] encryptBloc(byte[] input){

		byte[] tmp = new byte[input.length];
		int t,u;
		int aux;
		int[] data = new int[input.length/4];
		for(int i =0;i<data.length;i++)
			data[i] = 0;
		int off = 0;
		for(int i=0;i<data.length;i++){
			data[i] =		((input[off++]&0xff))|
							((input[off++]&0xff) << 8) |
							((input[off++]&0xff) << 16) |
							((input[off++]&0xff) << 24);
		}

		int A = data[0],B = data[1],C = data[2],D = data[3];

		B = B + S[0];
		D = D + S[1];
		for(int i = 1;i<=r;i++){
			t = rotl(B*(2*B+1),5);
			u = rotl(D*(2*D+1),5);
			A = rotl(A^t,u)+S[2*i];
			C = rotl(C^u,t)+S[2*i+1];

			aux = A;
			A = B;
			B = C;
			C = D;
			D = aux;
		}
		A = A + S[2*r+2];
		C = C + S[2*r+3];

		data[0] = A;data[1] = B;data[2] = C;data[3] = D;

		for(int i = 0;i<tmp.length;i++){
			tmp[i] = (byte)((data[i/4] >>> (i%4)*8) & 0xff);
		}

		return tmp;
	}

	private static byte[] paddingKey(byte[] key){
		int l = key.length%4;
		for(int i=0;i<l;i++)
			key[key.length+i] = 0;
		return key;
	}

	public static byte[] encrypt(byte[] data) {
		String keySeed ="12345678";
		byte[] key = keySeed.getBytes();

		byte[] bloc = new byte[16];
		key = paddingKey(key);
		S = generateSubkeys(key);
```

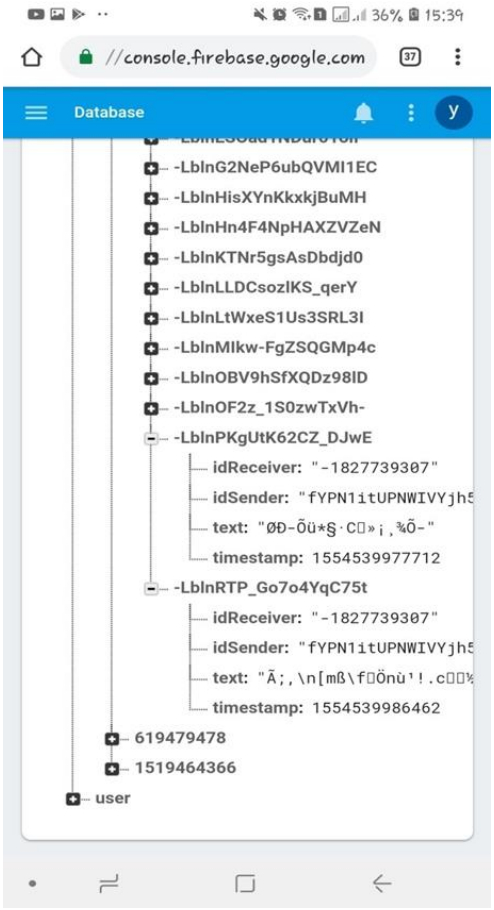
```
            int lenght = 16 - data.length % 16;
            byte[] padding = new byte[lenght];
            padding[0] = (byte) 0x80;

            for (int i = 1; i < lenght; i++)
                    padding[i] = 0;
            int count = 0;
            byte[] tmp = new byte[data.length+lenght];
            //afiseazaMatrice(S);
            int i;
            for(i=0;i<data.length+lenght;i++){
                    if(i>0 && i%16 == 0){
                            bloc = encryptBloc(bloc);
                            System.arraycopy(bloc, 0, tmp, i-16, bloc.length);
                    }

                    if (i < data.length)
                            bloc[i % 16] = data[i];
                    else{
                            bloc[i % 16] = padding[count];
                            count++;
                            if(count>lenght-1) count = 1;
                    }
            }
            bloc = encryptBloc(bloc);
            System.arraycopy(bloc, 0, tmp, i - 16, bloc.length);
            return tmp;
    }

    public static byte[] decrypt(byte[] data) {
            String keySeed ="12345678";
            byte[] key = keySeed.getBytes();

            byte[] tmp = new byte[data.length];
            byte[] bloc = new byte[16];
            key = paddingKey(key);
            S = generateSubkeys(key);

            int i;
            for(i=0;i<data.length;i++){
                    if(i>0 && i%16 == 0){
                            bloc = decryptBloc(bloc);
                            System.arraycopy(bloc, 0, tmp, i-16, bloc.length);
                    }

                    if (i < data.length)
                            bloc[i % 16] = data[i];
            }

            bloc = decryptBloc(bloc);
            System.arraycopy(bloc, 0, tmp, i - 16, bloc.length);

            tmp = deletePadding(tmp);
            return tmp;
    }

    private static byte[] deletePadding(byte[] input){
            int count = 0;

            int i = input.length - 1;
            while (input[i] == 0) {
                    count++;
                    i--;
            }

            byte[] tmp = new byte[input.length - count - 1];
            System.arraycopy(input, 0, tmp, 0, tmp.length);
            return tmp;
```

```
            }
}
```

V. HASIL PENELITIAN

A. Pengujian Menampilkan Hasil Enkripsi Pesan pada Database

Pada pengujian aplikasi di uji untuk menampilkan isi *database* untuk menampilakan apakah pesan yang dikirim terenkripsi. Pada tahap ini aplikasi mampu mengirimkan pesan yang terenkripsi pada *database* (lihat Gambar 3).



Gambar 3. Isi *database* dengan pesan yang sudah dienkripsi

B. Pengujian Perangkat Lunak

Setelah program diuji dan bekerja dengan baik pada emulator dan *device*, maka selanjutnya aplikasi diuji coba dengan berbagai *device* yang berbeda. Hal ini betujuan untuk mengetahui apakah aplikasi juga dapat berjalan pada smartphone lainnya yang memiliki resolusi layar dan versi Android yang berbeda. Untuk lebih jelasnya dapat dilihat pada tabel berikut.

TABEL IV

TABEL PENGUJIAN APLIKASI DI BERBAGAI DEVICE

No	Handphone	Resolusi Layar	Versi Android	Keterangan
1	Samsung Galaxy Grand Prime	960 x 540 px	4.4	Berjalan Dengan Baik

No	Handphone	Resolusi Layar	Versi Android	Keterangan
	Duos			
2	Smartfren Andromax	854 x 480 px	5.0	Berjalan Dengan Baik
3	Oppo Neo 7	960 x 540 px	5.0	Berjalan Dengan Baik
4	Vivo V5	1280 x 720 px	6.0	Berjalan Dengan Baik
5	Xiomi 4x	1280 x 720 px	7.0	Berjalan dengan Baik
6	Asus Zenfone 5	1280 x 720 px	5.0	Berjalan Dengan Baik
7	Huawei GR3	1280 x 720 px	6.0	Berjalan Dengan Baik

VI. SIMPULAN DAN SARAN

Berdasarkan pada hasil tercapai dalam penelitian ini, dapat diambil kesimpulan bahwa aplikasi ini telah dapat mengamankan informasi berupa teks sehingga tidak semua orang dapat mengambil pesan yang dikirimkan tersebut dengan sembarangan. Dengan adanya aplikasi ini *user* dapat menyebarkan informasi tekstual, sekaligus memberikan *user* keamanan dalam mengirim pesan dengan proses enkripsi-deskripsi.

Setelah mengembangkan aplikasi enkripsi pesan *chatting* tersebut, ada beberapa saran yang sebagai langkah pengembangan sistem selanjutnya, antara lain: aplikasi ini dapat juga melakukan pengamanan pada data digital lain nya, seperti citra digital, audio, dan video. Disarankan pula agar aplikasi menerapkan mekanisme *message digest untuk two-step encryption*. Aplikasi ini juga dapat dikembangkan untuk enkripsi pada data *stream* sehingga dapat juga digunakan untuk melakukan panggilan suara (*voice over Internet*).

DAFTAR PUSTAKA

[1] R. Mangundap and W. Kristiana, "Aplikasi Secure Message Menggunakan Algoritma CR6 Berbasis Android", *E-Jurnal SPIRIT PRO PATRIA*, Vol. 1 No. 2, 96-110, 2015.
[2] R. L. Rivest, M. J. Robshaw, R. Sidney, and Y.L. Yin, The RC6 block cipher. In *in First Advanced Encryption Standard (AES) Conference*, 1998.
[3] C. Rachmawati and A. Candra, "Implementasi Kombinasi Caesar dan Affine Cipher untuk keamanan Data Teks", *Jurnal Edukasi dan Penelitian Informatika (JEPIN), Nomor, 2*, 60-63, 2015.
[4] I. Gunawan, "Kombinasi algoritma Caesar cipher dan algoritma RSA untuk pengamanan file dokumen dan pesan teks," *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, Vol. *2* No. 2, 124-129, 2018.
[5] R. M. Yunus and H. Sujadi, "Sistem Keamanan Pesan Dengan Algoritma Rivest Code 6 (RC-6) Menggunakan Java Pada Smartphone Berbasis Android", *J-ENSITEC*, Vol. *2*, No. 1, 2015.