Available at https://jtsiskom.undip.ac.id (24 April 2020)
DOI:10.14710/jtsiskom.2020.13468

Jurnal Teknologi dan Sistem Komputer, 8(3), 2020, 171-177

Perbandingan kinerja RSA dan AES terhadap kompresi pesan SMS menggunakan algoritme Huffman

Performance comparison of RSA and AES to SMS messages compression using Huffman algorithm

Laurentinus*), Harrizki Arie Pradana, Dwi Yuny Sylfania, Fransiskus Panca Juniawan

Teknik Informatika, Fakultas Teknologi Informasi, Institut Sains dan Bisnis Atma Luhur
Jl. Jendral Sudirman, Kota Pangkal Pinang, Kepulauan Bangka Belitung, Indonesia 33123

Cara sitasi: L. Laurentinus, H. A. Pradana, D.Y.Sylfania, and F. P. Juniawan, "Perbandingan kinerja RSA dan AES terhadap kompresi pesan SMS menggunakan algoritme Huffman," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 3, pp. 171-177, 2020. doi: 10.14710/jtsiskom.2020.13468, [Online].

Abstract – Improved security of short message services (SMS) can be obtained using cryptographic methods, both symmetric and asymmetric, but must remain efficient. This paper aims to study the performance and efficiency of the symmetric crypto of AES-128 and asymmetric crypto of RSA with message compression in securing SMS messages. The ciphertext of RSA and AES were compressed using the Huffman algorithm. The average AES encryption time for each character is faster than RSA, which is 5.8 and 24.7 ms/character for AES and AES+Huffman encryption and 8.7 and 45.8 ms/character for RSA and RSA+Huffman, from messages with 15, 30, 60, and 90 characters. AES decryption time is also faster, which is 27.2 ms/character compared to 47.6 ms/character in RSA. Huffman compression produces an average efficiency of 24.8 % for the RSA algorithm, better than 17.35 % of AES efficiency for plaintext of 1, 16, 45, and 88 characters.

Keywords *– cryptography; Huffman compression; RSA; AES; SMS encryption*

Abstrak – Peningkatan keamanan layanan pesan singkat (SMS) dapat dilakukan dengan menggunakan metode kriptografi, baik simetris maupun asimetris, namun harus tetap efisien. Penelitian ini bertujuan untuk membandingkan kinerja dan efisiensi dari sistem kripto simetris AES-128 dan asimetris RSA terhadap kompresi data dalam mengamankan pesan SMS. Hasil enkripsi algoritme RSA dan AES dipadatkan menggunakan algoritme Huffman. Kinerja waktu enkripsi dan dekripsi rata-rata tiap karakter AES lebih cepat daripada RSA, yaitu 5,8 dan 24,7 mdetik/karakter untuk enkripsi AES dan AES+Huffman serta 8,7 dan 45,8 mdetik/karakter untuk enkripsi RSA dan RSA+Huffman, dari pesan dengan 15, 30, 60, dan 90 karakter. Waktu dekripsi AES lebih cepat, yaitu 27,2 mdetik/karakter dibandingkan 47,6 mdetik/karakter pada RSA. Kompresi Huffman menghasilkan efisiensi rata-rata sebesar sebesar 24,8 % untuk algoritme RSA yang lebih baik daripada efisisiensi 17,35 % pada AES untuk karakter plainteks 1, 16, 45, dan 88 karakter.

Kata kunci *– kriptografi; kompresi Huffman; RSA; AES; enkripsi SMS*

I. PENDAHULUAN

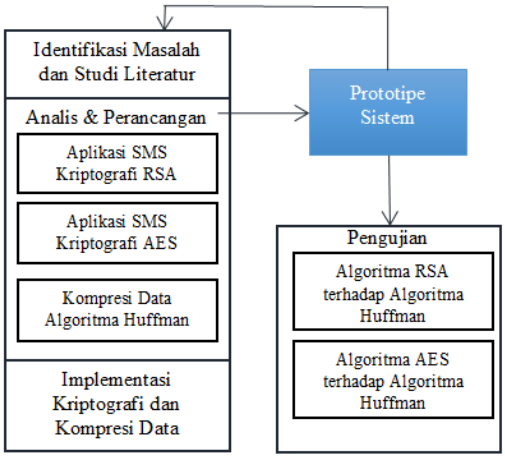
Telepon seluler menyediakan berbagai fitur, salah satunya adalah media *Short Message Service* (SMS). Layanan SMS dilaksanakan menggunakan telepon seluler untuk mengirim dan menerima pesan singkat. Layanan pesan ini telah digunakan di setiap bidang, di antaranya perbankan untuk transaksi *m-banking*, bidang *e-commerce*, dan *personal*. Suatu pesan SMS dikirim melalui telepon seluler melewati *Base Transceiver Station* (BTS) dan diarahkan ke *Short Message Service* (SMSC) [1]. Komunikasi melalui SMS memungkinkan data dicuri dan tidak aman. Salah satu cara dalam mengamankan informasi di komunikasi data adalah menggunakan teknik kriptografi, baik simetris, asimetris maupun hibrida.

Saat ini, telah banyak algoritme kriptografi yang dikembangkan dan diminati, seperti algoritme RSA sebagai kriptografi asimetris dan AES sebagai kriptografi simetris. Algoritme RSA memiliki keamanan yang baik yang terletak pada pemfaktoran bilangan yang besar menjadi faktor-faktor prima yang sulit dipecahkan. Pemfaktoran dilakukan untuk memperoleh kunci privat. Algoritme AES bekerja dalam bentuk byte atau karakter. Ukuran blok untuk algoritme AES-128 adalah sebesar 128 bit (16 byte) dalam mengenkripsi dan menghasilkan cipherteks.

Algoritme RSA dan AES telah banyak diterapkan untuk mengatasi masalah keamanan data dan telah telah banyak dikaji kekuatan, kelemahan, biaya, dan kinerja masing-masing algoritme tersebut [2]-[5]. Beragam parameter dipertimbangkan dalam analisis kinerja terrsebut, yaitu rata-rata penggunaan prosesor dan memori, waktu respons, dan konsumsi daya dalam [6]-

*) Penulis korespondensi (Laurentinus)
Email: laurentinus@atmaluhur.ac.id

Submitted: 20 Agustus 2019; Revised: 12 April 2020; Accepted: 19 April 2020; Published: 31 July 2020



Gambar 1. Kerangka penelitian secara keseluruhan

[8]. Lebih lanjut, RSA telah diterapkan dalam beragam aplikasi, di antaranya pada sistem tertanam [9], [10], layanan komputasi awan [11], dan citra optik [12]. Sementara itu, teknik kriptografi simetris seperti AES telah dikaji dalam [13], [14].

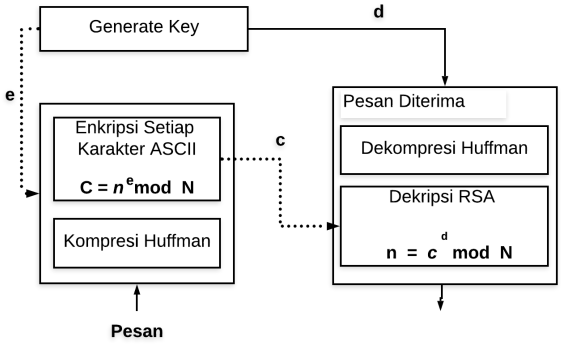
Namun, penelitian tersebut di atas hanya mencari algoritme kriptografi yang terbaik dalam mengamankan data, sedangkan permasalahan umum dalam penerapan teknik enkripsi, yaitu membesarnya cipherteks yang dihasilkan, tidak dikaji. Kajian ini bertujuan untuk menerapkan kompresi Huffman terhadap cipherteks AES-128 dan RSA dari pesan SMS seperti dalam [15] yang menggunakan RC6. Huffman diterapkan untuk memperkecil jumlah bit cipherteks tanpa kehilangan data (*lossless*). Kinerja waktu enkripsi dan dekripsi AES dan RSA tanpa kompresi dan dengan kompresi dikaji. Tingkat efisiensi kompresi data terhadap cipherteksAES dan RSA juga dikaji untuk menunjukkan kinerja terbaik teknik pengamanan pesan SMS.

II. METODE PENELITIAN

Penelitian ini menggunakan metode statistik deskriptif. Kerangka kajian yang menjelaskan tahapan penelitian ditunjukkan pada Gambar 1. Purwarupa yang dihasilkan adalah aplikasi pengiriman dan penerimaan SMS yang dilengkapi dengan teknik kriptografi RSA dan AES serta kompresi Huffman.

A. Kriptografi RSA

Teknik kriptografi asimetris yang digunakan dalam kajian ini adalah RSA. Proses RSA meliputi pembangkitan kunci privat (*d*) yang digunakan dalam dekripsi pesan dan kunci publik (*e*) untuk enkripsi, proses enkripsi+kompresi, dan proses dekripsi+dekompresi seperti ditunjukkan pada Gambar 2. Pesan terenkripsi atau cipherteks (*C*) dimampatkan dengan menggunakan kompresi Huffman. Pesan terenkripsi dan terkompresi (*c*) didekompresi dan didekripsi menggunakan kunci private *d*. Proses pembangkitan pasangan kunci RSA (*d*,*e*) dinyatakan dalam Algoritme 1. Proses enkripsi dan dekripsi RSA ditunjukkan dalam Algoritme 2 dan Algoritme 3.



Gambar 2. Tahapan algoritme RSA

Algoritme 1. Proses pembangkitan pasangan kunci RSA

```
1: bitlength ← 16;
2: blocksize ← 64;
3: bitlen ← 128;

4: Random r = new SecureRandom();
5: p ← probablePrime(bitlength, r);
6: q = new BigInteger(bitlength, 100, r);
7: n = p.multiply(q);

8: phi = p.subtract(BigInteger.ONE).multiply(q.subtract
   (BigInteger.ONE));

//Mencari nilai d menggunakan Euclid gcd(e, φ(n) ) = 1
9: e = probablePrime(bitlength, r);
10: while (phi.gcd(e).compareTo(BigInteger.ONE) > 0
    && e.compareTo(phi) < 0 ) {
11:     e.add(BigInteger.ONE);
}
12: d = e.modInverse(phi);
```

Algoritme 2. Proses enkripsi RSA

```
// Perulangan untuk melakukan enkripsi per karakter
1: for (int i = 0; i < text.length(); ++i) {
2:    c ← text.charAt(i)
3:    j ← (int) c
    //mengubah String menjadi ASCII
4:    ascii = new BigInteger(String.valueOf(j));

5:   do_enc = bb.enkripsi(ascii, E,N); //enkripsi
6:   ciphertext += do_enc;
}
function enkripsi(n , e , N  ) {
7:    C = n^e  mod  N;
8:    return C;
}
```

B. Kriptografi AES

Algoritme 3. Proses dekripsi RSA

```
1: for (int index = 0; index < ciphertextArray.length; +
    +index) {
2:    dec ← new BigInteger(ciphertextArray[index]);
3:    dec_key ← new BigInteger(Skunci);
4:    mod ← new BigInteger(Smodulus);
   //dekripsi RSA
5:    do_decr ← bb.decrypt(dec, dec_key, mod);
}
function decrypt( c ,d , N) {
6:    n = c^d  mod N;
7:    return n;
}
```

Teknik kriptografi simetris yang digunakan dalam kajian ini adalah AES-128 yang menggunakan kunci yang sama untuk melakukan enkripsi dan dekripsi pesan. Proses enkripsi dalam algoritme AES-128 dinyatakan dalam Algoritme 4. Proses *AddRoundKey* melakukan logika XOR atau ronde inisial antara plainteks / *state* awal dengan kunci. Jumlah putaran *Nr* dalam kajian ini adalah 10 kali putaran, sedangkan jumlah blok *Nb*-nya dan panjang kunci *Nk*-nya bernilai 4. Dalam setiap putaran sebanyak *Nr*-1 kali, dilakukan empat proses, yaitu *SubBytes* atau substitusi byte dengan menggunakan tabel substitusi (Sbox), *ShiftRows* atau melalui permutasi byte data dari kolom larik yang berbeda, *MixColumns* atau mengacak data di tiap kolom keadaan larik, dan *AddRoundKey* atau melakukan operasi XOR antara data dengan kunci. Putaran terakhir (*final round*) melakukan proses untuk putaran terakhir *SubBytes*, *ShiftRows*, dan *AddRoundKey*. Ekspansi kunci yang diperlukan AES-128 adalah menggunakan *Nb*(*Nr*+1) atau 44 word.

Proses dekripsi AES-128 dinyatakan dalam Algoritme 5. Dengan menggunakan kunci yang sama, proses dekripsi meliputi *KeyExpansion*, penambahan kunci inisial *AddRoundKey*, sembilan ronde putaran untuk inversi *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundKey* serta ronde final untuk melakukan inversi *InvShiftRows*, *InvSubBytes*, dan *AddRoundKey.*

C. Pengkodean Huffman

Kajian ini menggunakan pengkodean Huffman sebagai metode kompresi dan dekompresi. Proses pengkodean Huffman dimulai dari memasukkan teks dan menghitung frekuensi pohon (*tree*) dari graf tak berarah yang saling terhubung dan tidak mengandung lintasan. Proses membuat pohon Huffman dilakukan setiap karakter dimulai dari *root* ke cabang hingga menjadi satu pohon Huffman. Pada cabang kiri pohon biner diberi label 0, sedangkan pada cabang kanan diberi label 1. Rangkaian bit yang terbentuk pada setiap lintasan dari *root* ke cabang merupakan kode prefiks untuk karakter yang berpasangan sehingga menghasilkan pohon Huffman. Proses pengkodean dinyatakan dalam Algoritme 6, sedangkan pendekodean dinyatakan dalam Algoritme 7.

Algoritme 4. Proses enkripsi AES-128

```
function encryptSMS(Kunci, pesan) {
 try {
 1:    Nk ← 4, Nb ← 4, Nr ← 10
       // generate key
 2:    Key key ← generateKey(Kunci);
 3:    Key generateKey(Kunci)
       throws Exception {
 4:      Key key ← new SecretKeySpec(Kunci,"AES");
 5:      return key;
       }
     // enkripsi AES
 6:    Cipher c = Cipher.getInstance("AES");
 7:    KeyExpansion((Key) (Nr, Nb))
 8:    AddRoundKey (0, Nb)
 9:    for (i=1 in Nr) {
10:      SubBytes(Nb)
11:      ShiftRows(Nb)
12:      MixColumns(Nb)
13:      AddRoundKey(Nb, KeyExpansion, i)
       }
14:    SubBytes(Nb)
15:    ShiftRows(Nb)
16:    AddRoundKey(Nb, KeyExpansion, i+1)
17:  return CipherText;
   } catch (Exception e) {
18:    return null;
   }
}
```

Algoritme 5. Proses dekripsi AES-128

```
function decryptSMS( Kunci, PesanTerenkripsi)
 1:   Nk ← 4, Nb ← 4, Nr ← 10
 2:   Cipher c = Cipher.getInstance("AES");
 3:   c.init(Cipher.DECRYPT_MODE, key);
 4:   byte[] decValue = c.doFinal(PesanTerenkripsi);

 5:   KeyExpansion((Key) (Nr, Nb))
 6:   AddRoundKey(0, Nb)
 7:   for (i=1 in Nr) {
 8:     InvShiftRows(Nb)
 9:     InvSubBytes(Nb)
10:     InvMixColumns(Nb)
11:     AddRoundKey(Nb, KeyExpansion, i)
      }
12:   InvShiftRows(Nb)
13:   InvSubBytes(Nb)
14:   AddRoundKey(Nb, KeyExpansion, i+1)
15:   return PlainText;
}
```

III. Hasil dan Pembahasan

Kajian ini menghasilkan aplikasi kriptografi RSA dan AES berbasis Android dengan fungsi pembangkitan kunci, pengiriman dan penerimaan pesan SMS. Proses pembangkit kunci publik berfungsi untuk membangkitkan kunci enkripsi untuk berkomunikasi SMS dan pembangkit kunci privat berfungsi untuk kunci dekripsi serta modulus (Gambar 3). Setelah proses pembangkitan kunci selesai, maka kunci publik dan modulus dikirim kepada nomor telepon tujuan melalui media SMS.

Halaman pesan SMS berfungsi untuk menerapkan algoritme dalam pengacakan dan kompresi pesan. Gambar 4 memperlihatkan penerapan algoritme RSA dan kompresi Huffman dalam mengamankan data teks sebelum dikirimkan ke penerima melalui komunikasi data dan hasil penerapannya pada pesan SMS. Gambar 5 memperlihatkan SMS yang masuk di daftar pesan penerima dengan informasi yang telah teracak. SMS tidak teracak dari pengirim lainnya juga dapat diterima dengan baik. Hasil dekripsi dan dekompresi pesan SMS diperlihatkan dalam Gambar 6 untuk mendapatkan pesan SMS asli dari pengirim.

Pengujian dilakukan untuk mengukur efisiensi dan kinerja data terhadap kriptografi dan kompresi. Pengujian pada algoritme RSA dilakukan dengan kunci

Algoritme 6. Proses pengkodean Huffman

Input:
text ← input dari pesan

Begin
1: mendefinisikan sebuah simpul dengan karakter, frekuensi, child sebelah kiri dan kanan dari simpul untuk pohon Huffman
2: buat daftar 'freq' untuk menyimpan frekuensi setiap karakter
3: **for** (setiap karakter c pada text) {
4: menambah frekuensi untuk karakter ch dalam daftar freq
}
5: **for** (all type of character ch) {
6: **if** (the frequency of ch is non zero) {
7: tambah ch dan frekuensinya sebagai node dari priority queue Q.
}
}
8: **while** (Q is not empty) {
9: Hapus item dari Q dan tetapkan ke kiri node child
10: Hapus item dari Q dan tetapkan ke kanan node child
11: Traverse node untuk mencari assigned code
}

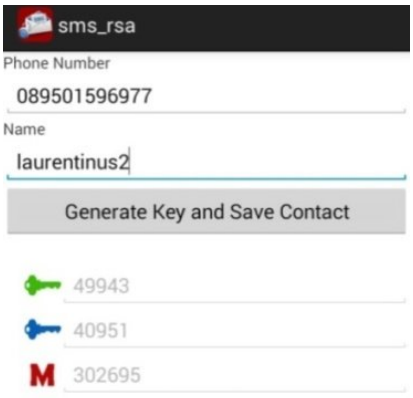
function traverse (n, code) {
12: **if**(a left child of node n ≠φ) {
13: traverseNode(leftChild(n), code+'0')
14: traverseNode(rightChild(n), code+'1')
} **else**
15: **return** character and data of current node
}
End

Algoritme 7. Proses pendekodean Huffman

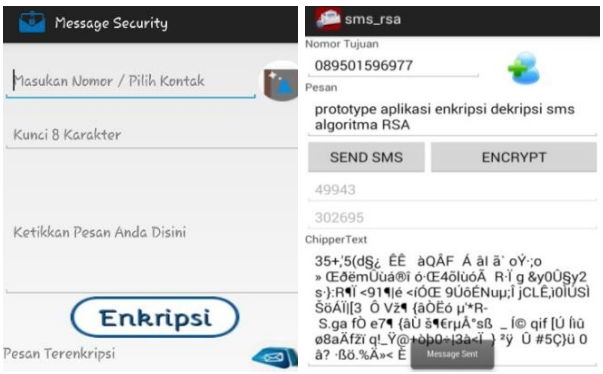
Input:
root ← root dari pohon huffman
S ← bit-stream untuk didekompresi
n ← S.length
Begin
1: **for** (i := 1 to n) {
2: current ← root
3: **while** (current.left != NULL dan current.right != NULL) {
4: **if** (S[i] === 0)
5: current := current.left
6: **else**
7: current := current.right
8: i = i+1
}
}
9: **return** current.symbol
End

publik 5 byte dan modulus 6 byte, sedangkan algoritme AES menggunakan kunci 8 byte. Pengujian dilakukan dengan cara mengenkripsi masing-masing algoritme sehingga didapatkan panjang cipherteks dan kemudian dikompres dengan algoritme Huffman untuk didapatkan persentase efisiensi kompresi data tiap algoritme. Contoh hasil enkripsi dengan dan tanpa kompresi Huffman diperlihatkan dalam Tabel 1. Hasil enkripsi dengan dan tanpa kompresi Huffman diperlihatkan dalam Tabel 2.

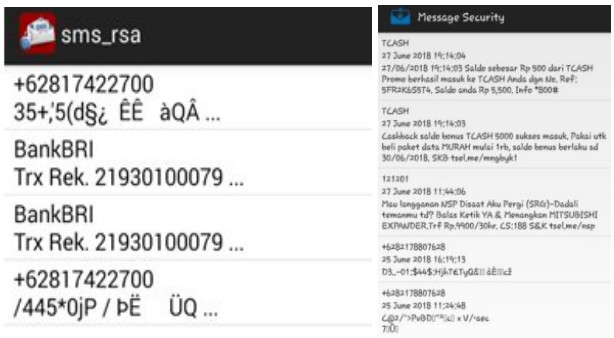
Hasil kompresi Huffman terhadap algoritme RSA dan AES dari empat variasi jumlah karakter plainteks



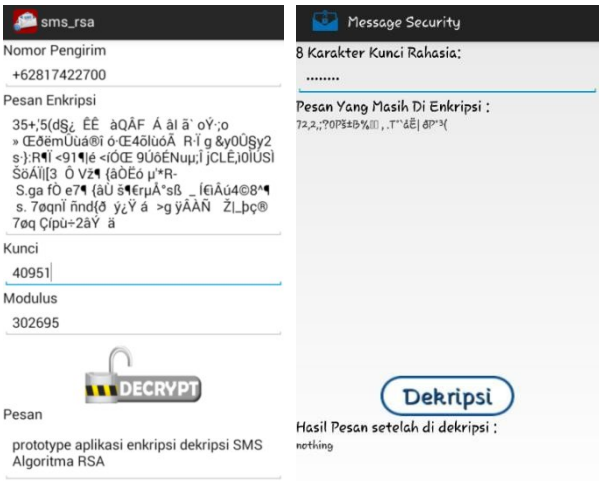
Gambar 3. Halaman pembangkitan kunci RSA



Gambar 4. Halaman pesan SMS RSA dan AES



Gambar 5. Halaman inbox RSA dan AES



Gambar 6. Halaman dekripsi dan dekompresi SMS RSA dan AES

yang sama, yaitu 1, 16, 45, dan 88 karakter, masing-masing dinyatakan dalam Tabel 3 dan Tabel 4. Perbandingan efisiensi kompresi Huffman terhadap algoritme RSA diperoleh rasio efisiensi rata-rata 24,8 % data cipherteks. Hasil kompresi Huffman terhadap algoritme AES memperoleh rasio efisiensi sebesar 17,35 % terhadap cipherteks. Kompresi Huffman ini melakukan kompresi dengan cara mengkodekan masukan ke dalam bentuk bit untuk mewakili data karakter. Semakin banyak karakter yang sama dalam teks, maka semakin besar efisiensinya. RSA merupakan algoritme dengan keluaran angka yang terdiri dari 0-9, sedangkan AES memiliki keluaran angka dan huruf (heksa desimal). Algoritme yang paling terpengaruh oleh algoritme Huffman adalah RSA dengan jumlah karakter yang terkompresi lebih efisien daripada AES.

Kompresi Huffman mampu melakukan kompresi terhadap cipherteks baik dari RSA maupun AES. Namun, rasio efisiensi kompresi Huffman terhadap cipherteks ini jauh lebih kecil daripada terhadap teks dalam [16] yang memperoleh rasio mencapai 63,63 % untuk teks dengan banyak huruf sama. Hal ini disebabkan algoritme Huffman bekerja berdasarkan frekuensi karakter sehingga semakin banyak karakter yang sama dalam plainteks, maka rasio efisiensi kompresi akan semakin tinggi seperti dinyatakan dalam [17].

Pengujian kinerja sistem dilakukan dengan cara menghitung waktu proses enkripsi dan dekripsi algoritme RSA dan AES untuk pesan sebesar 15, 30, 60 dan 90 karakter. Kinerja waktu pemrosesan enkripsi ditunjukkan dalam Gambar 7, sedangkan waktu dekripsi dalam Gambar 8. Waktu enkripsi algoritme RSA rata-rata adalah 0,0087 detik/karakter, sedangkan algoritme AES membutuhkan waktu 0,0058 detik/karakter.

Waktu enkripsi dipengaruhi oleh panjang pesan, panjang kunci, dan tipe kriptografi. Semakin panjang pesan yang dienkripsi, maka semakin lama proses enkripsi seperti ditunjukkan dalam Gambar 7, yaitu waktu enkripsi yang dibutuhkan sebanding dengan banyak jumlah karakter. Semakin panjang kunci yang digunakan, maka data semakin aman, namun hal ini mempengaruhi waktu enkripsi dan dekripsi. Tipe kriptografi RSA menggunakan 2 kunci, yaitu publik dan privat, sehingga membutuhkan waktu enkripsi yang lebih lama dibandingkan dengan algoritme AES yang menggunakan 1 kunci.

Algoritme RSA dengan Huffman memerlukan waktu enkripsi rata-rata 0,0458 detik/karakter dan algoritme AES dengan Huffman membutuhkan waktu 0,0247 detik/karakter. Waktu pemrosesan enkripsi terhadap jumlah karakter setelah penambahan kompresi Huffman dipengaruhi oleh hasil enkripsi, yaitu semakin panjang suatu cipherteks, proses kompresi semakin lama (Gambar 7). Hal ini dapat ditunjukkan dari perbandingan RSA dan AES yang semakin jauh seiring besarnya data plainteks.

Waktu proses dekripsi RSA dengan AES yang telah dikompres ditunjukkan pada Gambar 8. Waktu proses dekripsi ini dipengaruhi oleh jumlah karakter. Semakin banyak jumlah karakter yang didekripsi, maka perbedaan waktu kedua algoritme semakin jauh, sama seperti proses enkripsi. Waktu dekripsi rata-rata per karakter dalam algoritme RSA dengan Huffman adalah 0,0476 detik/karakter, sedangkan algoritme AES dengan Huffman membutuhkan waktu rata-rata 0,0272 detik/karakter.

Hasil kinerja waktu proses yang diperoleh dalam kajian menunjukkan bahwa waktu enkripsi algoritme

Tabel 1. Pengujian enkripsi RSA dan kompresi

Karakter	Hasil Enkripsi tanpa huffman	Hasil Enkripsi dengan Huffman
h	139484;	/61*+,5ú
halo, apa kabar?	139484;200793;279297;76086;167849;159508;200793;58123;200793;159508;64458;200793;299667;200793;18518914727	/61*+,5-&F@ ”E{!ä ¹ o> ßîàì ^| ñ¿ð[kÀ Ó^7 ` ñŠÙã ·Æ Ùõ[ö`Ç } ÆÙõ[ùo0á¿gXõ^ËÓ

Tabel 2. Pengujian enkripsi AES dan kompresi

Karakter	Hasil Enkripsi tanpa huffman	Hasil Enkripsi dengan Huffman
h	B114E95845481ED3BF7C187D2D02D93B	@-,^|bV’H”Ÿ?!. ¦SŠ -€
halo, apa kabar?	608A8BB7C4A5548CCD0F6271AF32586108C81B39D5F2366FA2C3AFDF047A5E2A	4,4=0xxZjN1ÙŇ Ãæè2/ iHñ ¡ç¿ű Yœ _ñ9ç·Þ%p

Tabel 3. Perbandingan hasil enkripsi dan kompresi algoritme RSA

Jumlah Karakter Plainteks	Cipherteks Enkripsi RSA	Cipherteks RSA dan Kompresi	Efisiensi RSA (%)
1	7	8	0
16	106	73	31,13
45	263	159	39,54
88	438	313	28,53
Rata-rata efisiensi			**24,8**

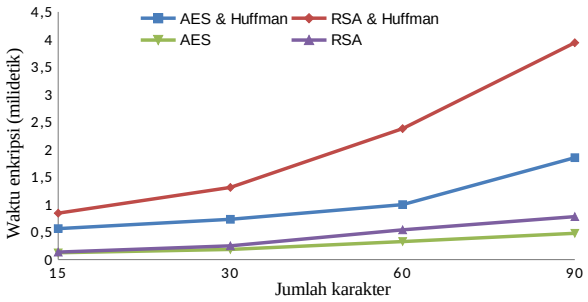
Tabel 4. Perbandingan hasil enkripsi dan kompresi algoritme AES

Jumlah Karakter Plainteks	Cipherteks Enkripsi AES	AES Enkripsi dan Kompresi	Efisiensi AES (%)
1	32	28	12,5
16	64	52	18,8
45	96	80	16,7
88	192	151	21,4
Rata-rata efisiensi			**17,35**

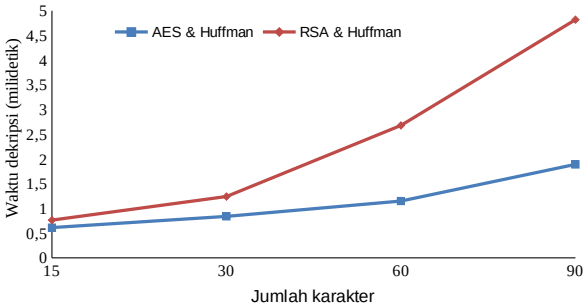
AES lebih cepat daripada algoritme RSA seperti halnya dinyatakan dalam [4], [6]-[8], [14]. Waktu proses dekripsi AES juga lebih cepat daripada RSA. Selain waktu pemrosesan yang lebih cepat, AES juga mempunyai efek Avalance lebih tinggi, sehingga dapat digunakan untuk aplikasi yang membutuhkan derajat kerahasiaan dan integritas yang tinggi, namun lebar pita jaringan yang dibutuhkan lebih tinggi daripada RSA [8]. Dalam penerapannya, proses pertukaran kunci tunggal dalam AES yang aman juga perlu diperhatikan karena digunakan untuk enkripsi dan dekripsi.

IV. KESIMPULAN

Algoritme kompresi Huffman dapat diterapkan pada cipherteks AES dan RSA yang memiliki tipe enkripsi yang berbeda, yaitu simetris dan asimetris. Algoritme AES mempunyai kinerja waktu enkripsi dan dekripsi yang lebih baik daripada RSA seiring dengan besarnya ukuran karakter yang dienkripsi. Namun, kompresi algoritme Huffman terhadap cipherteks RSA lebih efisien daripada AES.



Gambar 7. Waktu enkripsi RSA dan AES terhadap jumlah karakter



Gambar 8. Waktu dekripsi RSA dan AES terhadap jumlah karakter

DAFTAR PUSTAKA

[1] T. Mantoro, L. Laurentinus, N. Agani, and M. A. Ayu, "Improving the security guarantees, authenticity and confidentiality in short message service of mobile applications," in *4th International Conference on Cyber and IT Service Management*, Bandung, Indnesia, Apr. 2016, pp. 1-6. doi: 10.1109/CITSM.2016.7577592

[2] A. Barenghi, G. M. Bertoni, L. Breveglieri, and G. Pelosi, "A fault induction technique based on voltage underfeeding with application to attacks against AES and RSA," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1864-1878, 2013. doi: 10.1016/j.jss.2013.02.021

[3] A. Al-Hasib and A. A. M. M. Haque, "A comparative study of the performance and security issues of AES and RSA cryptography," in *Third International Conference on Convergence and Hybrid Information Technology*, Busan, South Korea, Nov. 2008, pp. 505-510. doi: 10.1109/ICCIT.2008.179

[4] N. Anwar, M. Munawwar, M. Abduh, and N. B. Santosa, "Komparatif performance model keamanan menggunakan metode algoritma AES 256 bit dan RSA," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 2, no. 3, pp. 783-791, 2018. doi: 10.29207/resti.v2i3.606

[5] V. S. Mahalle and A. K. Shahade, "Enhancing the data security in cloud by implementing hybrid (RSA & AES) encryption algorithm," in *2014 International Conference on Power, Automation and Communication*, Amravati, India, Oct. 2014, pp. 146-149. doi: 10.1109/INPAC.2014.6981152

[6] G. Singh and S. Supriya, "A study of encryption algorithms (RSA, DES, 3DES and AES) for information security," *International Journal of Computer Applications*, vol. 67, no. 19, pp. 33-38, 2013. doi: 10.5120/11507-7224

[7] P. Mahajan and A. Sachdeva, "A study of encrytion algorithms AES, DES and RSA for security," *Global Journal of Computer Science and Technology Network, Web & Security*, vol. 13, no. 15, pp. 14-20, 2013.

[8] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, "A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish," in *Procedia Computer Science*, vol. 78, pp. 617-624, 2016. doi: 10.1016/j.procs.2016.02.108.

[9] A. Thomas and E. M. Manuel, "Embedment of montgomery algorithm on elliptic curve cryptography over rsa public key cryptography," *Procedia Technology*, vol. 24, pp. 911-917, 2016. doi: 10.1016/j.protcy.2016.05.179

[10] N. Silva, D. Pigatto, P. Martins, and K. Branco, "Case studies of performance evaluation of asymmetric and symmetric cryptographic algorithms for embedded systems and a general purpose computer," *Journal of Network and Computer Applicaitons*, vol. 60, no. C, pp. 130-143. 2015. doi: 10.1016/j.jnca.2015.10.007

[11] N. Khanezaei and Z. M. Hanapi, "A framework based on RSA and AES encryption algorithms for cloud computing services," in *IEEE Conference on System, Process and Control*, Kuala Lumpur, Malaysia, Dec. 2014, pp. 56-62. doi: 10.1109/SPC.2014.7086230

[12] L. Gong, K. Qiu, C. Deng, and N. Zhou, "An optical image compression and encryption scheme

based on compressive sensing and RSA algorithm," *Optics and Lasers in Engineering*, vol. 121, pp. 169-180, 2019. doi: 10.1016/j.optlaseng.2019.03.006

[13] J. Thakur and N. Kumar, "DES, AES and Blowfish: symmetric key cryptography algorithms simulation based performance analysis," *International Journal of Emerging Technology and Advanced Engineering*, vol. 1, no. 2, pp. 6-12, 2011.

[14] R. Bhanot and R. Hans, "A review and comparative analysis of various encryption algorithms," *International Journal of Security and Its Applications*, vol. 9, no. 4, pp. 289-306, 2015. doi: 10.14257/ijsia.2015.9.4.2

[15] L. Laurentinus, "Implementasi kriptografi dan kompresi SMS menggunakan algoritma RC6 dan algoritma Huffman berbasis Android," *Jurnals of Indo Global Mandiri University*, vol. 8, no. 1, pp. 36-42, 2017.

[16] A. P. U. Siahaan, "Implementasi teknik kompresi teks Huffman," *Jurnal Informatika*, vol. 10, no. 2, pp. 1251-1261, 2016. doi: 10.26555/jifo.v10i2.a5070

[17] A. Pahdi, "Algoritme Huffman dalam pemampatan dan enkripsi data," *IJNS: Indonesian Journal of Network & Security*, vol. 6, no. 3, pp. 1-7, 2017.