

# Aplikasi Algoritma Hybrid RSA dalam Layanan Web dan Surat Elektronik

Eka Novendra Wahyunadi 13517011  
Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia  
ekanovendra726@gmail.com

**Abstrak**—Di era informasi ini pertukaran data berlangsung sangat cepat, namun seiring meningkatnya kecepatan pertukaran data, risiko keamanan data juga semakin meningkat, karena itulah diperkenalkan sistem kriptografi seperti algoritma *DES (Data Encryption Standard)*, *AES (Advanced Encryption Standard)*, dan *RSA* untuk meningkatkan keamanan dalam pertukaran data. Setelah beberapa tahun digunakan, algoritma seperti *RSA* dirasa masih kurang aman, sehingga muncul pengembangannya, yaitu algoritma *ERSA (Enhanced RSA)*, tetapi meski sudah lebih aman, *ERSA* masih dirasa banyak kelemahannya, sehingga muncul pengembangan baru dari *RSA*, yaitu algoritma *Hybrid RSA (HRSA)*. Beberapa bentuk dari *HRSA* adalah algoritma *HRSA* dengan menggabungkan *AES*, dan algoritma *HRSA* dengan menggunakan empat bilangan prima. *HRSA* mempunyai beberapa keunggulan dibandingkan dengan *ERSA* atau *RSA*, karena itulah *HRSA* diaplikasikan ke beberapa proses pengamanan dalam pertukaran data, seperti dalam layanan web dan surat elektronik.

**Kata Kunci**—Hybrid RSA, Kriptografi, RSA, Teori Bilangan

## I. PENDAHULUAN

Pada tahun 1976, tiga peneliti dari *MIT (Massachusetts Institute of Technology)*, yaitu Ron Rivest, Adi Shamir, dan Len Adleman, memperkenalkan Algoritma *RSA*, sebuah algoritma kriptografi yang mendasarkan proses enkripsi dan dekripsinya pada konsep tentang bilangan prima dan aritmetika modulo. Kunci enkripsi dan kunci dekripsi dari algoritma *RSA* berupa bilangan bulat. Karena kunci enkripsi tidak dirahasiakan dan diketahui oleh umum, maka kunci enkripsi dinamakan juga kunci publik, sedangkan untuk kunci dekripsi bersifat rahasia. Kunci dekripsi dibangkitkan dari beberapa buah bilangan prima bersama-sama dengan kunci enkripsi.

Untuk menghasilkan kunci dekripsi, seseorang harus memfaktorkan suatu bilangan non prima menjadi faktor primanya. Memfaktorkan bilangan non prima menjadi faktor primanya bukanlah pekerjaan yang mudah, semakin besar bilangan non prima yang akan difaktorkan, tentu semakin sulit pemfaktoranannya. Algoritma yang ada saat ini salah satunya adalah algoritma *shor*, namun algoritma ini adalah sebuah algoritma kuantum (sebuah algoritma yang berjalan di komputer kuantum). Pada tahun 2012 tercatat bilangan non prima terbesar yang berhasil difaktorkan menjadi faktor primanya oleh algoritma *shor* adalah 21, dengan faktor prima 3 dan 7[1], sehingga bisa dikatakan saat ini belum ada algoritma yang cukup mangkus (efisien) untuk memfaktorkan sebuah bilangan

non prima menjadi faktor primanya.

Seiring perkembangan teknologi, algoritma *RSA* pun ikut berkembang, muncul variasi – variasi baru seperti *ERSA (Enhanced RSA)*, dan *HRSA (Hybrid RSA)*. Ada berbagai macam *Hybrid RSA* yang sudah dikembangkan, seperti *HRSA* yang menggabungkan *AES (Advanced Encryption Standard)* dengan *RSA*, dan *HRSA* yang menggunakan 4 bilangan prima. Pengembangan dari algoritma *RSA* berupa *HRSA* dapat dimanfaatkan dalam beberapa bentuk, contohnya seperti penggunaannya dalam mengamankan layanan web dan surat elektronik.

## II. DASAR TEORI

### A. *DES (Data Encryption Standard)*

Algoritma *DES* memadukan teknik permutasi, ekspansi, kompaksi, dan substitusi, proses - proses tersebut dilakukan dalam 16 kali perulangan. Panjang kunci dari *DES* adalah 8 karakter atau 64 bit. Meski terdapat 64 bit, tetapi untuk proses enkripsinya hanya dipakai 56 bit saja[2].

### B. *AES (Advanced Encryption Standard)*

*Advanced Encryption Standard*, yang juga dikenal sebagai algoritma enkripsi *Rijindael*, adalah sebuah algoritma kriptografi simetri yang merupakan improvisasi dari algoritma *DES*. Panjang kunci *AES* bisa berukuran 128 bit, 192 bit, dan 256 bit. Menurut *FIPS PUB (Federal Information Processing Standards Publication) 197* yang dikeluarkan oleh *NIST (National Institute of Standards and Technology)*[3] untuk algoritma *AES*, panjang kunci direpresentasikan sebagai  $N_k = 4, 6$ , atau 8, yang merefleksikan bilangan dari *32-bit words* (banyak kolom) di *Cipher Key*. Ukuran blok masukan maupun blok keluaran di algoritma *AES* ini sama yaitu 128 bits, yang direpresentasikan sebagai  $N_b = 4$ , yang merefleksikan bilangan dari *32-bit words* di *State*. Banyak perulangan yang dilakukan pada algoritma ini bergantung pada ukuran kuncinya, banyaknya perulangan direpresentasikan sebagai  $N_r$ , sehingga  $N_r = 10$  ketika  $N_k = 4$ ,  $N_r = 12$  ketika  $N_k = 6$ , dan  $N_r = 14$  ketika  $N_k = 8$ [3]. Kombinasi panjang kunci, ukuran blok, dan banyaknya perulangan ditunjukkan pada gambar 1.

	Key Length ( <i>Nk words</i> )	Block Size ( <i>Nb words</i> )	Number of Rounds ( <i>Nr</i> )
<b>AES-128</b>	4	4	10
<b>AES-192</b>	6	4	12
<b>AES-256</b>	8	4	14

Gambar 1. Kombinasi panjang kunci, ukuran blok, dan banyaknya perulangan pada algoritma AES[3]

### C. RSA

RSA adalah algoritma kriptografi nirsimetri, atau biasa disebut juga algoritma kriptografi kunci – publik. Algoritma dari RSA adalah sebagai berikut[2].

Pembangkitan pasangan kunci

1. Pilih dua bilangan prima sembarang, misal a dan b. Jaga kerahasiaan a dan b ini.
2. Hitung  $n = ab$ , besaran n tidak perlu dirahasiakan.
3. Hitung  $m = (a-1)(b-1)$ . Sekali m telah dihitung, a dan b dapat dihapus untuk mencegah diketahuinya oleh pihak lain.
4. Pilih sebuah bilangan bulat untuk kunci publik, misal e, yang relatif prima terhadap m.
5. Hitung kunci dekripsi, misal d, dengan kongruen  $ed \equiv 1 \pmod{m}$ .

Enkripsi

1. Nyatakan pesan menjadi blok – blok plainteks:  $p_1, p_2, p_3, \dots$  (harus dipenuhi persyaratan bahwa nilai  $p_1$  harus terletak dalam himpunan nilai 0, 1, 2, 3, ..., n-1 untuk menjamin hasil perhitungan tidak berada di luar himpunan).
2. Hitung blok cipherteks  $c_1$  untuk blok plainteks  $p_1$  dengan persamaan
$$c_i = p_i^e \pmod{n}$$
yang dalam hal ini, e adalah kunci publik.

Dekripsi

1. Proses dekripsi dilakukan dengan menggunakan persamaan
$$p_i = c_i^d \pmod{n}$$
yang dalam hal ini, d adalah kunci pribadi.

Dalam implementasinya, nilai a dan b disarankan nilai yang sangat besar (200 angka) agar pekerjaan memfaktorkan n menjadi faktor primanya menjadi sangat sulit.

### D. ERSA (Enhanced RSA)

Konsep dari ERSA hampir sama dengan RSA, dengan perubahan kecil di bagian masukan, contohnya penggunaan tiga bilangan prima daripada hanya dua bilangan prima. Untuk proses enkripsi dan dekripsinya sama dengan RSA. Algoritma dari ERSA adalah sebagai berikut.

Pembangkitan pasangan kunci

1. Pilih tiga bilangan prima sembarang, misal x, y dan z. Jaga kerahasiaan x, y dan z ini.
2. Hitung  $n = xyz$ , besaran n tidak perlu dirahasiakan.
3. Hitung  $m = (x-1)(y-1)(z-1)$ . Sekali m telah dihitung, x, y dan z dapat dihapus untuk mencegah diketahuinya oleh pihak lain.
4. Pilih sebuah bilangan bulat untuk kunci publik, misal e, yang relatif prima terhadap m.
5. Hitung kunci dekripsi, misal d, dengan kongruen  $ed \equiv 1 \pmod{m}$ .

Enkripsi

1. Nyatakan pesan menjadi blok – blok plainteks:  $p_1, p_2, p_3, \dots$  (harus dipenuhi persyaratan bahwa nilai  $p_1$  harus terletak dalam himpunan nilai 0, 1, 2, 3, ..., n-1 untuk menjamin hasil perhitungan tidak berada di luar himpunan).
2. Hitung blok cipherteks  $c_1$  untuk blok plainteks  $p_1$  dengan persamaan
$$c_i = p_i^e \pmod{n}$$
yang dalam hal ini, e adalah kunci publik.

Dekripsi

1. Proses dekripsi dilakukan dengan menggunakan persamaan
$$p_i = c_i^d \pmod{n}$$
yang dalam hal ini, d adalah kunci pribadi.

Dengan penambahan bilangan ketiga, tingkat keamanan meningkat seiring dengan meningkatnya kesulitan dalam memfaktorkan bilangan non primanya.

### E. Hybrid RSA dengan menggabungkan AES

Dari penjelasan Algoritma AES dan RSA, sekelompok peneliti[4] membuat sebuah program dengan bahasa pemrograman python untuk mengetes waktu yang dibutuhkan dalam proses untuk enkripsi dan dekripsi dari teks. Ukuran teks dari tiga tes adalah 105k, 1024k, dan 15360k. Hasil eksperimen tersebut ditunjukkan pada Tabel I.

TABEL I. HASIL TES ENKRIPSI DAN DEKRIPSI ALGORITMA AES DAN RSA[4]

Berkas	Algoritma AES		Algoritma RSA	
	Enkripsi	Dekripsi	Enkripsi	Dekripsi
Berkas 1	0.15s	0.21s	1.02s	42.54s
Berkas 2	0.61s	0.69s	4.25s	91.46s
Berkas 3	2.02s	2.57s	53.45s	1069.37s

Analisis hasil eksperimen:

1. Kecepatan enkripsi dari algoritma AES dan algoritma RSA lebih cepat dari kecepatan dekripsi.
2. Semakin besar ukuran data, semakin lambat kecepatan enkripsi dan dekripsi dari kedua algoritma.
3. Kecepatan enkripsi dan dekripsi dari algoritma AES lebih cepat dari algoritma RSA.

Dari eksperimen dan hasil analisis algoritma tersebut, didapat bahwa kecepatan enkripsi dari algoritma AES lebih cepat daripada algoritma RSA. Algoritma AES cocok untuk mengenkripsi data yang berukuran besar, sedangkan algoritma RSA lebih cocok untuk mengenkripsi data yang berukuran kecil, dan mampu melakukan tanda tangan digital dan autentikasi identitas. Algoritma RSA memiliki sedikit kunci, sehingga manajemen kunci mudah, sedangkan kebalikannya berlaku untuk algoritma AES. Jadi *Hybrid RSA* ini bisa mengenkripsi data dengan algoritma AES, dan mengenkripsi kunci AES dengan kunci publik algoritma RSA, sehingga distribusi kunci AES bisa lebih aman.

- F. *Hybrid RSA* dengan menggunakan empat bilangan prima
- Menurut sekelompok peneliti[5] dari *HRSA* ini adalah dengan menggunakan empat bilangan prima, hasil perkalian dari empat bilangan prima ini adalah sebuah variabel  $M$ . Kunci publik ( $P$ ) dan kunci pribadi ( $Q$ ) bergantung ke nilai  $M$ , dengan komputasi secara tidak langsung. Bilangan acak  $P1$  dan  $P2$  dibutuhkan untuk mengevaluasi nilai dari kunci publik ( $P$ ). Jadi, butuh waktu lebih banyak untuk membangkitkan kunci dikarenakan meningkatnya kompleksitas. Nilai dari  $x$  atau  $M$  tidak ditransmisikan sebagai kunci publik, melainkan sebuah nilai  $x$  baru dibangkitkan, misal  $w$ ,  $w$  ditransmisikan sebagai kunci publik. Proses dari enkripsi dan dekripsi bergantung ke  $w$ , sehingga sangat sulit untuk menembus sistem keamanan tersebut. Algoritma *HRSA* ini ditunjukkan pada gambar 2a dan 2b.

```

HRSA_Encryption ()
Input: Select Plain text (T1), Public key(U) and
Random
num ber (r).
Output: Find Cipher text (C1).
Begin
Procedure (T1, U, r and C1)
 $C1 \leftarrow TU \text{ mod } r$ 
End Procedure
End

HRSA_Decryption ()
Input: Select Cipher text (C1), Private key (U) and
Random
num ber (r).
Output: Find Plain text (T1).
Begin
Procedure (C1, R, r and T1)
 $T1 \leftarrow C1R \text{ mod } r$ 
End Procedure

```

Gambar 2a. Algoritma *HRSA* dengan 4 bilangan prima[5]

```

HRSA_Key_Generation ()
Input: Select four random distinct prime numbers w,
x, y and z.
Output: Find Public Key (U), Private Key (R) and
Random number (r).
Begin
Procedure (w, x, y, z, U, R and r)
1.  $j \leftarrow w * x$ 
2.  $k \leftarrow y * z$ 
3.  $M \leftarrow j * k$ 
4. Calculate Euler  $O()$  of j, k and M
a.  $O(j) \leftarrow (w-1) * (x-1)$ 
b.  $O(k) \leftarrow (y-1) * (z-1)$ 
c.  $O(M) \leftarrow O(j) * O(k)$ 
5. Generate a random number r1, such that,
 $\text{gcd}(r1, O(x)) = 1, 1 < r1 < O(j)$ 
6. Generate a random number r2, such that,
 $\text{gcd}(r2, O(k)) = 1, 1 < r2 < O(k)$ 
7. Calculate  $R1 \leftarrow r1$ 
 $r2 \text{ mod } M$ 
8. Generate a public key U, such that,
 $\text{gcd}(U, O(M) * R1) = 1, 1 < U < O(M) * R1$ 
9. Calculate the private key Q, such that,
 $R \leftarrow U-1 \text{ mod } (O(M) * R1)$ 
10. Compute a random number r, such that,
If  $w > x$ 
Satisfy  $j - w < r < j$  and  $\text{gcd}(r, j) = 1$ 
Else if  $w < x$ 
Satisfy  $j - x < r < j$  and  $\text{gcd}(r, j) = 1$ 
End Procedure
End

```

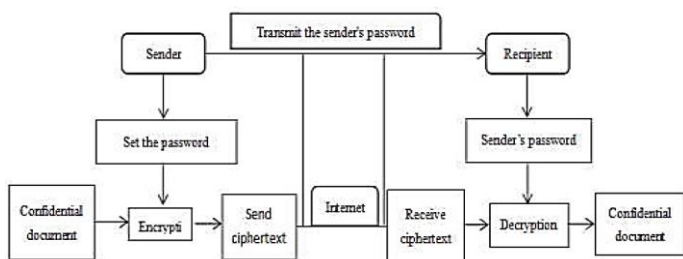
Gambar 2b. Algoritma *HRSA* dengan 4 bilangan prima[5]

### III. METODE APLIKASI HYBRID RSA

- A. Aplikasi Algoritma *Hybrid RSA* dengan AES dalam Pengamanan Surat Elektronik

Dalam kegiatan berkomunikasi lewat surat elektronik sehari – hari, surat elektronik biasa tidak perlu untuk dienkripsi, kedua belah pihak bisa saling bertukar surat elektronik secara langsung, tetapi hal ini tidak berlaku jika surat elektronik yang dikirimkan berisi berkas atau dokumen yang rahasia, untuk kasus tersebut seseorang bisa menggunakan fungsi enkripsi yang sudah disediakan oleh surat elektroniknya. Pengirim surat elektronik bisa mengatur kata sandi untuk enkripsi, lalu mengenkripsi berkas atau dokumen tersebut, dan kemudian surat elektronik dikirimkan ke alamat surat elektronik tujuan. Setelah menerima surat elektroniknya, penerima bisa mendekripsi berkas atau dokumen dengan menggunakan kata sandi yang dibangkitkan oleh pengirim untuk memastikan transmisi dari surat elektronik tersebut aman

Sistem seperti ini biasa disebut sebagai sistem kriptografi simetri, yaitu ketika kunci yang digunakan untuk enkripsi dan dekripsi adalah kunci yang sama, dan karena itu sistem ini mempunyai masalah di bagian distribusi kunci. Sistem kriptografi biasa ditunjukkan pada gambar 3.



Gambar 3. Sistem kriptografi biasa[4]

Jika terdapat seorang peretas yang sudah mengawasi pengguna untuk waktu yang lama, maka sudah tidak aman lagi untuk mentransmisikan kunci lewat jaringan internet yang terbuka, karena sangat besar kemungkinannya peretas tersebut bisa mendapatkan kunci lalu mendeskripsi berkas atau dokumen yang dikirim, sehingga peretas tersebut juga bisa mengetahui isi berkas atau dokumen tersebut, lebih parah lagi jika peretas tersebut mengubah isi dari berkas atau dokumen tersebut lalu mengirimkannya kembali ke alamat tujuan awal sehingga berkas atau dokumen yang diterima oleh penerima tidak sesuai seperti yang dikirim.

Selain itu sistem kriptografi simetri tidak bisa ditanda tangan digital sehingga identitas dari pihak – pihak yang saling berkomunikasi menjadi tidak bisa dikonfirmasi, contohnya seperti berkas – berkas yang berisi tentang perjanjian atau kontrak, karena tidak bisa dikonfirmasi identitas dari pengirimnya, maka mungkin saja terjadi penyangkalan kontrak, penipuan maupun perusakan berkas.

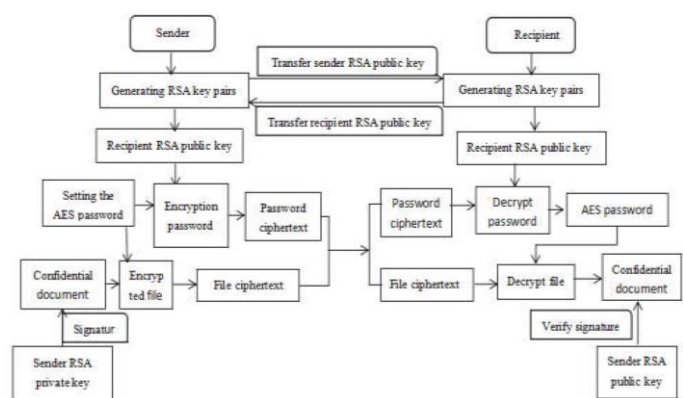
Aplikasi algoritma *Hybrid RSA* dengan *AES* dalam pengamanan surat elektronik adalah berupa fitur surat elektronik yang dapat menunjukkan bahwa transmisi berkas dan dokumen telah dienkripsi. Mode enkripsi *hybrid* menambahkan tanda tangan digital dan enkripsi kunci simetri ke enkripsi kunci simetri yang sudah ada. Kerangka kerja dari aplikasi ini ditunjukkan dalam gambar 4.

B. Aplikasi Algoritma *Hybrid RSA* dengan 4 Bilangan Prima dalam Layanan Web

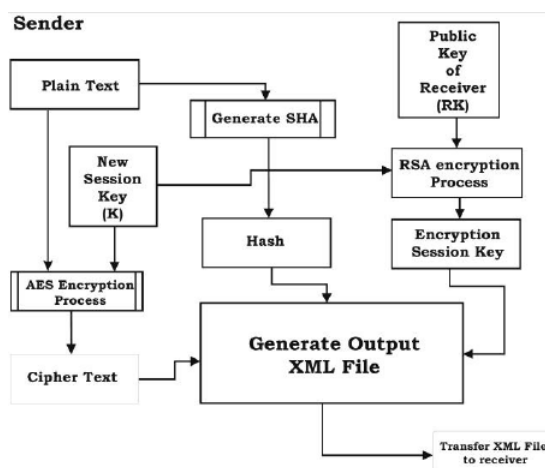
Algoritma *Hybrid RSA* ini bisa diaplikasikan dalam bentuk pertukaran informasi antara aplikasi pengguna dan administrasi web. Algoritma *Hybrid RSA* ini meningkatkan keamanan pertukaran informasi antara dua pengguna dengan memanfaatkan web sebagai perantara. Algoritma ini mengkodekan bagian – bagian dari informasi yang bisa diuraikan oleh sisi pengguna dan tanpa menggunakan sisi *server*.

Dua modul terpisah dibuat untuk mengaplikasikan algoritma ini, yaitu sebuah aplikasi Windows (sisi pengguna) dan sebuah web (sisi server). Aplikasi Windows mengkodekan konten informasi polos yang dimasukkan oleh pengguna dengan cara memanfaatkan *half and half encryption*. Web mendapatkan informasi yang dikirim oleh aplikasi Windows, dan mencoba untuk menguraikan informasi yang diterima dengan memanfaatkan *half and half decoding*[5]. Setelah penerimaan informasi, web memberikan umpan balik kepada program pengguna, contohnya seperti kembali ke aplikasi Windows.

Grafik alur untuk perjalanan informasi pada sisi pengguna ditunjukkan pada gambar 5, sedangkan grafik alur untuk perjalanan informasi pada web (sisi server) ditunjukkan pada gambar 6.

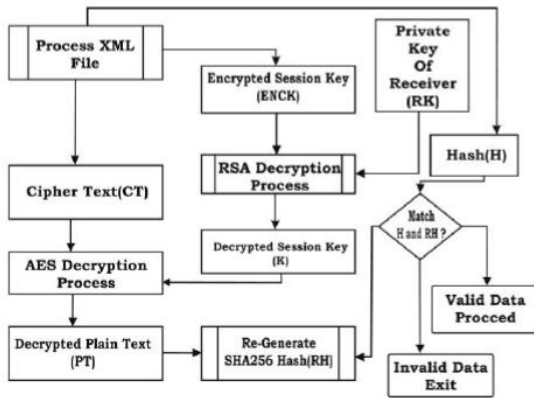


Gambar 4. Kerangka kerja aplikasi algoritma *Hybrid RSA* dengan *AES* dalam pengamanan surat elektronik[4]



Gambar 5. Grafik alur perjalanan informasi di sisi pengguna[5]

Dibandingkan dengan enkripsi kunci simetri tunggal, terlihat jelas keuntungan dari mode *hybrid* ini. Dari keseluruhan proses transmisi surat elektronik, bahkan jika seorang peretas mendapatkan kunci publik RSA dan berkas yang dienkripsi, dia tetap tidak bisa mengetahui isi berkas maupun mencoba untuk mengubah isinya.



Gambar 6. Grafik alur perjalanan informasi di sisi server[5]

#### IV. SIMULASI APLIKASI HYBRID RSA

##### A. Aplikasi Algoritma Hybrid RSA dengan AES dalam Pengamanan Surat Elektronik

Dalam simulasi yang dilakukan oleh sekelompok peneliti[4] untuk mengimplementasikan algoritma Hybrid RSA dengan AES dalam pengamanan surat elektronik, digunakan bahasa pemrograman Java. Algoritma hybrid ini dibagi dalam tiga modul, yaitu modul untuk tanda tangan digital dan verifikasi tanda tangan digital, modul untuk enkripsi dan dekripsi kunci AES, serta modul untuk enkripsi dan dekripsi data. Modul untuk tanda tangan digital ditunjukkan pada gambar 7. Pada awalnya kunci pribadi untuk RSA didapatkan, setelah itu berkas ditanda tangan secara digital.

```
public static String sign(byte[] data, String privateKey) throws Exception {
    // 解密base64编码的私钥
    byte[] keyBytes = decryptBASE64(privateKey);

    // 构造PKCS8EncodedKeySpec对象
    PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(keyBytes);

    // KEY_ALGORITHM 指定的加密算法
    KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);

    // 获取私钥对象
    PrivateKey priKey = keyFactory.generatePrivate(pkcs8KeySpec);

    // 用私钥对信息生成数字签名
    Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM);
    signature.initSign(priKey);
    signature.update(data);

    return encryptBASE64(signature.sign());
}
```

Gambar 7. Tanda tangan digital[4]

Modul yang dipergunakan untuk bagian enkripsi dan dekripsi data pada simulasi ini diimplementasikan menggunakan Cipher class yang sudah disediakan oleh bahasa pemrograman Java. Implementasi dari modul enkripsi dan dekripsi data tersebut ditunjukkan pada gambar 8.

```
public static byte[] decrypt(byte[] data, String key) throws Exception {
    Key k = toKey(decryptBASE64(key));

    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.DECRYPT_MODE, k);

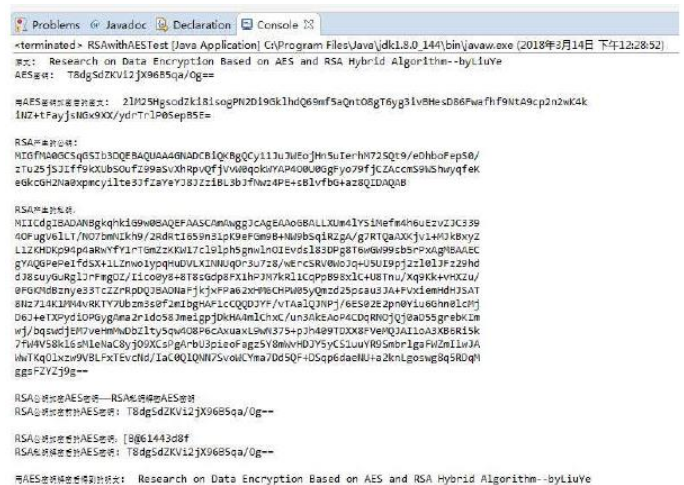
    return cipher.doFinal(data);
}

public static byte[] encrypt(byte[] data, String key) throws Exception {
    Key k = toKey(decryptBASE64(key));
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.ENCRYPT_MODE, k);

    return cipher.doFinal(data);
}
```

Gambar 8. Modul enkripsi dan dekripsi data menggunakan Cipher class yang disediakan Java[4]

String yang digunakan dalam simulasi ini adalah “Research on Data Encryption Based on AES and RSA Hybrid Algorithm—byLiuYe”, hasil dari simulasi ini ditunjukkan dalam gambar 9.

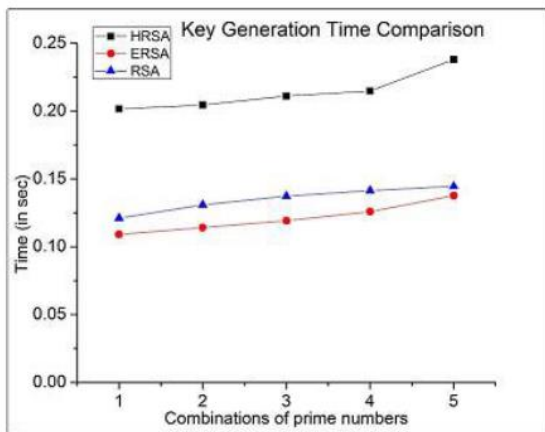


Gambar 9. Hasil simulasi[4]

##### B. Aplikasi Algoritma Hybrid RSA dengan 4 Bilangan Prima dalam Layanan Web

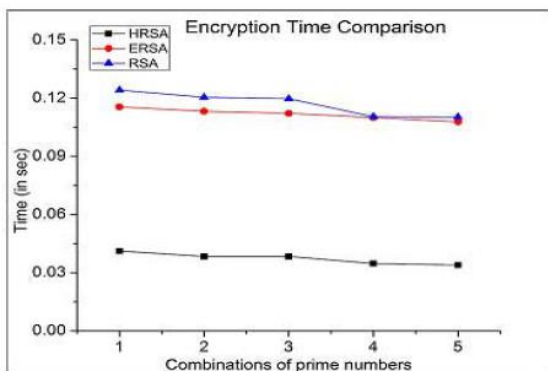
Sekelompok peneliti[5] mencoba mensimulasikan aplikasi algoritma ini dengan menggunakan MATLAB, mereka menentukan dua bilangan prima untuk RSA, tiga bilangan prima untuk ERSA dan empat bilangan prima untuk HRSA. Keunggulan dari algoritma ini sangat jelas terlihat dengan mempertimbangkan beberapa parameter, seperti waktu pembangkitan kunci, waktu enkripsi, dan waktu dekripsi.

Karena HRSA menggunakan empat bilangan prima, maka tentu saja masih bisa dimaklumi jika waktu pembangkitan kunci dari HRSA lebih tinggi dibanding ERSA dan RSA. Dari simulasi dibuktikan bahwa waktu pembangkitan kunci algoritma HRSA lebih lama 84.66% daripada ERSA dan 64.43% lebih lama daripada RSA[5]. Ilustrasi perbandingan waktu pembangkitan kunci dari ketiga algoritma tersebut ditunjukkan pada gambar 10.



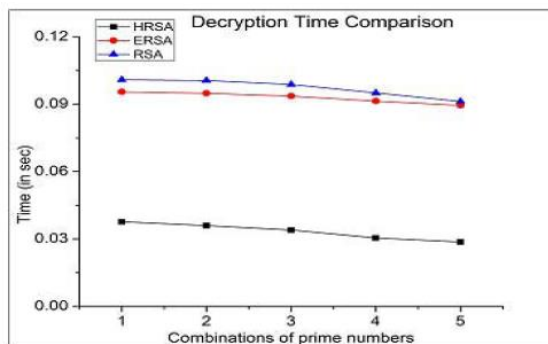
Gambar 10. Ilustrasi perbandingan waktu pembangkitan kunci antara HRSA, ERSA, dan RSA[5]

Untuk waktu yang dibutuhkan pada proses enkripsi, dari simulasi didapatkan bahwa terdapat peningkatan, yaitu algoritma HRSA 64.37% lebih cepat daripada ERSA dan 66.85% lebih cepat daripada RSA[5]. Ilustrasi perbandingan waktu enkripsi ditunjukkan pada gambar 11.



Gambar 11. Ilustrasi perbandingan waktu enkripsi antara HRSA, ERSA dan RSA[5]

Untuk waktu yang dibutuhkan pada proses dekripsi, dari simulasi didapatkan bahwa terdapat peningkatan, yaitu algoritma HRSA 60.55% lebih cepat daripada ERSA dan 62.65% lebih cepat daripada RSA[5]. Ilustrasi perbandingan waktu dekripsi ditunjukkan pada gambar 12.



Gambar 12. Ilustrasi perbandingan waktu dekripsi antara HRSA, ERSA, dan RSA[5]

## V. KESIMPULAN

Pengembangan algoritma kriptografi RSA berupa *Hybrid RSA* memberikan pilihan baru dalam meningkatkan keamanan dalam bertukar informasi, dari penjelasan metode aplikasi dan dari simulasi aplikasi HRSA, dapat disimpulkan bahwa pengembangan algoritma kriptografi RSA berupa *Hybrid RSA* sangat berguna, seperti *Hybrid RSA* yang menggunakan AES untuk meningkatkan keamanannya, dari simulasi bisa dikatakan bahwa *Hybrid RSA* tersebut sangat berguna dalam meningkatkan keamanan dari bagian distribusi kunci, selain itu banyak keuntungan lain dari *Hybrid RSA* tersebut, seperti kemudahan dalam verifikasi identitas dari pengirim, sehingga bisa meminimalkan risiko penipuan dan pemalsuan data, selain itu *Hybrid RSA* tersebut juga bisa meminimalkan risiko peretasan karena meski peretas mendapatkan kunci publik RSA dan berkas yang dienkripsi, dia tetap akan kesulitan jika ingin mengubah keseluruhan atau sebagian data, bahkan dia akan kesulitan hanya untuk mengetahui isi data tersebut. Selain terdapat *Hybrid RSA* yang memanfaatkan AES, terdapat juga *Hybrid RSA* yang memanfaatkan empat buah bilangan prima, dari hasil simulasi, telah terbukti bahwa *Hybrid RSA* ini mempunyai keunggulan yang jauh terhadap RSA dan Enhanced RSA di bagian waktu enkripsi dan waktu dekripsi, bahkan sampai lebih dari 60%. Namun meskipun *Hybrid RSA* seolah – olah sudah sangat baik, tetapi tetap saja *Hybrid RSA* mempunyai kelemahan, entah yang sudah ditemukan atau belum, contohnya seperti *Hybrid RSA* yang menggunakan empat bilangan prima, waktu pembangkitan kuncinya terhitung lama jika dibandingkan dengan RSA dan Enhanced RSA, walaupun bisa dimaklumi dikarenakan *Hybrid RSA* tersebut menggunakan empat bilangan prima untuk membangkitkan kunci – kuncinya, selain itu *Hybrid RSA* tersebut juga tidak bisa dipergunakan di perangkat dengan tenaga yang kecil (*low powered device*), masalah – masalah seperti ini tentu saja akan secara terus - menerus muncul dan menunggu untuk dipecahkan, mungkin saja dengan munculnya masalah – masalah tersebut bisa mengeluarkan sebuah solusi yang membuat aplikasi *Hybrid RSA* tidak sebatas untuk pengamanan layanan web dan surat elektronik saja, tetapi juga bisa diaplikasikan ke bidang yang lain.

## VI. UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa, karena atas limpahan rahmat dan karunia-Nya penulis bisa menulis dan menyelesaikan makalah ini dalam kondisi yang sehat. Penulis juga ingin berterima kasih kepada kedua orang tua penulis, karena berkat bantuan dan dukungan mereka penulis bisa mendapatkan ilmu dan pengetahuan hingga saat ini. Selain itu penulis juga ingin berterima kasih kepada Ibu Harlili, karena melalui pengajarannya penulis dapat mengerti konsep – konsep yang ada dalam Matematika Diskrit, terutama konsep tentang teori bilangan serta penggunaannya dalam bidang kriptografi yang menjadi dasar dari makalah ini. Tak lupa penulis juga mengucapkan terima kasih kepada para penulis sumber referensi yang telah memberikan saya ilmu yang dibutuhkan untuk menyelesaikan makalah ini.

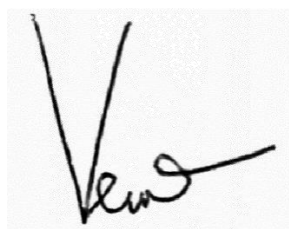
## REFERENSI

- [1] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nat. Photonics*, vol. 6, p. 773, Oct. 2012.
- [2] R. Munir, *Matematika Diskrit*, 4th ed. Penerbit INFORMATIKA Bandung, 2010.
- [3] NIST, "FIPS PUB 197: Specification for the Advanced Encryption Standard (AES)," 2001.
- [4] Y. Liu, W. Gong, and W. Fan, "Application of AES and RSA Hybrid Algorithm in E-mail," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, 2018, pp. 701–703.
- [5] J. Gondaliya, S. Savani, V. S. Dhadvai, and G. Hossain, "Hybrid Security RSA Algorithm in Application of Web Service," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 2018, pp. 149–152.

## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Desember 2018



Eka Novendra Wahyunadi  
13517011