ANALISA PERBANDINGAN HYBRID CRYPTOGRAPHY RSA-AES DAN ECDH-AES UNTUK KEAMANAN PESAN

Lilik Widyawati[1)] **Husain**[2)] **Muhamad Azwar**[3)] **Mikha Christian Satria Girsang**[4)]
Program Studi Ilmu Komputer[1) 3) 4)]
Program Studi RekayasaPerangkat Lunak Aplikasi[2)]
Fakultas Teknik, Universitas Bumigora, Mataram, Nusa Tenggara Barat[1) 2) 3) 4)]
lilikwidya@universitasbumigora.ac.id[1)] Husain@universitasbumigora.ac.id[2)]
muhamadazwar@universitasbumigora.ac.id[3)] 1810520086@universitasbumigora.ac.id[4)]

ABSTRACT

Information exchange becomes easier, information exchange can easily be known by unauthorized parties if it is not accompanied by a security system, this can be overcome by combining symmetric algorithms with asymmetric algorithms or commonly called Hybrid cryptography. This research was conducted with the aim of knowing the results of the Comparative Analysis of Hybrid cryptography RSA-AES and ECDH-AES in message security by measuring processing time, memory and security when the algorithm is run on measuring applications. The methodology used in making the measurement application to be built is Rapid Application Development. (RAD) and the stages used are requirements planning, application design, and implementation. The results of the RSA-AES experiment found that the lowest and highest RSA-AES processing times were 0.03 seconds and 7 seconds, the lowest and highest RSA-AES memory were 2.9 KB and 8.9 KB, while in the ECDH -AES experiment, the results shows that the lowest and highest processing time is 0.003 seconds and 0.01 seconds, the lowest and highest ECDH-AES memory is 2.6 KB and 4.2 KB. The safety test shows that it takes 4.6 seconds on the secp192r1 curve with 4 digit numbers in the discrete logarithm of the elliptic curve while on prime factors, it takes 0.003 seconds to solve the problem with 4 digit numbers. Based on the experiments that have been carried out, it is known that ECDH-AES has faster time performance and lower memory requirements and higher security.

Keywords : Hybrid Cryptography, RSA-AES, ECDH-AES, Prime Factorization, Elliptic Curve Discrete Logarithms

ABSTRAK

Pertukaran informasi sudah semakin mudah dilakukan, pertukan informasi tersebut bisa dengan mudah untuk diketahui oleh pihak yang tidak berwenang jika tidak dibarengi dengan system keamanan, hal ini dapat diatasi dengan menggabungkan algoritma simetris dengan algoritma asimetris atau biasa disebut *Hybridcryptography*. Penelitian ini dilakukan dengan tujuan untuk mengetahui hasil Analisa Perbandingan *Hybridcryptography* RSA-AES dan ECDH-AES dalam keamanan pesan dengan mengukur waktu proses, memori dan keamanan saat algoritma dijalankan di aplikasi pengukur .Metodologi yang digunakan dalam pembuatan aplikasi pengukur yang akan dibangun adalah *Rapid Application Development* (RAD) dan tahapan yang digunakan yaitu perencanaan kebutuhan, perancangan aplikasi, dan implementasi. Hasil percobaan RSA-AES, diperoleh bahwa waktu proses terendah dan tertinggi RSA-AES adalah 0.03 detik dan 7 detik, memori terendah dan tertinggi RSA-AES adalah 2.9 KB dan 8.9 KB, sedangkan pada percobaan ECDH-AES, diperoleh hasil bahwa waktu proses terendah dan tertinggi adalah 0.003 detik dan 0.01 detik, memori terendah dan tertinggi ECDH-AES adalah 2.6 KB dan 4.2 KB. Percobaan keamanan menunjukan bahwa dibutuhkan waktu selama 4.6 detik di kurva secp192r1 dengan 4 digit angka di logaritma diskrit kurva eliptik sedangkan pada pemfaktoran prima, dibutuhkan waktu selama 0.003 detik untuk memecahkan masalah dengan 4 digit angka. Berdasarkan percobaan yang telah dilakukan diketahui bahwa ECDH-AES memiliki performa waktu yang lebih cepat dan kebutuhan memori yang lebih rendah serta kemanan yang lebih tinggi.

Kata Kunci : Hybridcryptography, RSA-AES, ECDH-AES, Faktorisasi Prima, Logaritma Diskrit Kurva Eliptik

PENDAHULUAN

Dalam perkembangan teknologi sekarang ini, pertukaran informasi sudah sangat mudah dilakukan. Namun kebutuhan keamanan dan kerahasiaan dari suatu informasi semakin berkembang dari hari ke hari [1] dan dibutuhkan suatu cara untuk menjaga keamanan dan kerahasiaan suatu informasi.

Keamanan data pada lalu-lintas jaringan merupakan suatu hal yang diinginkan semua orang untuk menjaga privasi, supaya data yang dikirim aman dari gangguan orang yang tidak bertanggung jawab maka harus disembunyikan menggunakan suatu sistem keamanan yang sering dikenal dengan istilah kriptografi. Pada kriptografi, terdapat dua proses yaitu enkripsi dan dekripsi. Enkripsi adalah proses mengubah suatu pesan menjadi tidak terbaca, sedangkan dekripsi adalah proses mengubah pesan yang sudah dienkripsi tersebut menjadi pesan yang bisa dibaca Kembali. Untuk melakukan kedua proses tersebut, dibutuhkan yang namanya algoritma kriptografi, salah satu algoritma kriptografi yang cukup dikenal adalah Advanced Encryption Standard (AES). AES merupakan algoritma enkripsi Simetris, algoritma ini menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. Selain algoritma simetris, terdapat juga algoritma asimetris yang merupakan sebuah algoritma yang membutuhkan kunci public dan kunci privat untuk meakukan proses enkripsi dan dekripsi. Dalam pertukaran pesan, algoritma simetris tidak cukup untuk mengamankan percakapan dikarenakan kedua pihak harus memiliki kunci rahasia yang sama, dan agar kedua pihak dapat memiliki kunci yang sama, kunci rahasia tersebut harus dikirim melalui saluran publik yang dimana pada saluran publik terdapat banyak pihak pihak yang tidak bersangkutan yang dapat mengambil kunci tersebut dan mendengarkan percakapan, hal ini dapat diatasi dengan menggunakan Hybridcryptography yang merupakan penggabungan algoritma simetris dengan algoritma asimetris.

Pada penelitian sebelumnya oleh [2], telah dilakukan perbandingan dari beberapa kombinasi Hybridcryptography yaitu RSA (Rivest Shamir Adleman), AES, dan ECC (Eliptic Curve Cryptography). Pada penelitian sebelumnya juga oleh [3], telah dilakukan dua perbandingan algoritma kriptografi yaitu Data Encryption Standard (DES) dan Rivest Shamir Adleman (RSA). Dan Penelitian oleh [4] telah dilakukan perbandingan antara algoritma RSA dan Elgamal pada keamanan sidik jari.

Berdasarkan penelitian terdahulu, belum ada penelitian yang membandingkan performa dan keamanan dari dua gabungan algoritma kriptografi. Pada penelitian ini akan akan dilakukan perbandingan antara algoritma RSA-AES dan ECDH-AES untuk mencari kombinasi yang memerlukan memori dan waktu proses yang rendah serta keamanan yang tinggi. Pada penelitian ini RSA akan digunakan untuk enkripsi-dekripsi kunci, ECDH akan digunakan untuk mendapatkan kunci Bersama dan AES akan digunakan untuk enkripsi pesan, parameter yang diukur pada penelitian ini adalah penggunaan memori saat program aplikasi RSA-AES dan ECDH-AES dijalankan, Waktu proses yang dibutuhkan, dan kemanan, dengan melakukan bruteforce pada permasalahan matematika yang diandalkan oleh algoritma RSA dan ECDH yaitu pemfaktoran prima dan logaritma diskrit kurva eliptik. Dari latar belakang diatas, maka penulis akan membandingkan algoritma RSA-AES dan ECDH-AES untuk mengetahui perbedaan dari dua hybridcryptography tersebut dari segi performa dan keamanan.

TINJAUAN PUSTAKA

Pngertian Kriptografi

Menurut Munir, R. (2006) Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keaman informasi seperti kerahasian, integrasi data, serta keaslian data Kriptografi bertujuan agar informasi yang bersifat rahasia, integrasi data, ontentika dan nirpenyangkalan yang dikirim melalui suatu jaringan internet, tidak dapat diketahui dan dimanfaatkan oleh orang lain atau pihak yang tidak berkepentingan. (dalam [5]

Menurut Amita Pandey (2013), dasar konsep kriptografi terdiri dari:

- Plain text, adalah pesan asli yang ingin dikirim.

- Cipher text, adalah pesan yang tidak dapat dimengerti oleh siapapun yang awalnya merupakan plain text.
- Encryption, mengkonversi plain text menjadi cipher text, membutuhkan 2 proses, algoritma enkripsi dan kunci.
- Decryption, mengkonversi cipher text menjadi plain text, membutuhkan 2 proses, algoritma dekripsi dan kunci.
- Key, merupakan kombinasi dari angka atau huruf atau symbol spesial yang digunakan dalam enkripsi dan dekripsi dan memiliki peran penting dalam kriptografi karena algoritma bergantung kepadanya (dalam [6])

Hybridcryptography

Menurut R. P. S and V. Paul (2011) Kriptografi hybrid merupakan protokol yang memanfaatkan beberapa *key* dari algoritma berbeda secara bersamaan dengan keunggulan tiap algoritma tersebut. Salah satu cara yang sering digunakan adalah membangkitkan kunci simetris dan mengenkripsi kunci ini dengan kunci asimetris dari kunci publik penerima. Data dienkripsi dengan kunci simetris dan kunci rahasia ini dikirim ke penerima kemudian penerima mendekripsi kunci rahasia terlebih dahulu menggunakan kunci privat miliknya, lalu mendekripsi data dengan kunci yang telah didekripsi tersebut (dalam [6])

Algoritma RSA

(Z. Arifin 2009) Ditemukan oleh Ron Rivest, Adi Shamir dan Len Adleman dari MIT (Massachusetts Institute of Technology) pada tahun 1977, algoritma RSA merupakan penemuan besar dalam kriptografi kunci public dan masih populer digunakan sampai saat ini, karena kunci-kunci yang panjang dan penerapannya makin disempurnakan (dalam [6]).

Adapun proses enkripsi dan dekripsi algoritma RSA adalah sebagai berikut :

1. Akan digunakan dua bilangan prima sembarang, sebut a dan b. Jaga kerahasiaan a dan b.
2. Hitung n = a × b. Besaran n tidak dirahasiakan.
3. Hitung m = (a – 1) × (b – 1). jika m telah dihitung, a dan b dapat dihapus agar tidak diketahui pihak lain, serta jaga kerahasiaan m
4. Akan digunakan sebuah bilangan bulat untuk kunci publik, sebut namanya e, yang relatif prima terhadap m.
5. Bangkitkan kunci deskripsi, d, dengan kekongruenan ed ≡ 1 (mod m) atau e.d (mod m) = 1. Lakukan enkripsi terhadap isi pesan dengan persamaan $c_i = p_i^{\ e} \bmod n$, yang dalam hal ini pi adalah blok plainteks, c_i adalah chiperteks yang diperoleh, dan e adalah kunci enkripsi (kunci publik). Harus dipenuhi persyaratan bahwa nilai pi harus terletak dalam himpunan nilai 0, 1, 2, …, n – 1 untuk menjamin hasil perhitungan tidak berada di luar himpunan.
6. Proses deskripsi dilakukan dengan menggunakan persamaan $p_i = c_i^{\ d} \bmod n$, yang dalam hal ini d adalah kunci deskripsi.

Algoritma Elliptic Curve

Elliptic Curve Cryptography (ECC) adalah salah satu pendekatan algoritma kriptografi kunci publik berdasarkan pada struktur aljabar dari kurva elliptik pada daerah finite. Penggunaan kurva elips dalam kriptografi dicetuskan oleh Neal Koblitz dan victor S. Miller pada tahun 1985.

Kurva ellips dapat ditulis dengan perhitungan matematis sebagai berikut:

$$\boldsymbol{y^2 = x^3 + ax + b} \quad (1)$$

Dalam kriptografi kunci asimetris, harus ditentukan terlebih dahulu nilai parameter yang akan digunakan dan telah disepakati oleh pihak yang akan berkomunikasi. Parameter yang digunakan dalam Algoritma ini yaitu nilai a dan b, bilangan prima p jika dalam persamaan kurva eliptik bidang terbatas serta titik generator G yang dipilih dari kurva eliptik. [7]

Kurva eliptik memiliki 3 karakteristik, yaitu :

- Kurva eliptic simetris sepanjang sumbu x
- P(x,y) = P(x,-y)
- Garis lurus akan bertemu dengan sumbu x pada titik 1 atau 3
- Kurva eliptic tidak *singular* (jika kita anggap $4a^3 + 27b^2 \neq 0$) [8]

National Institute of Standards and Technology atau NIST merekomendasikan 5 jenis kurva

yang dapat digunakan yaitu, secp192r1 (P-192), secp224r1 (P-224), secp256r1 (P-256), secp384r1 (P-384), secp521r1 (P-521). (National Institute of Standards and Technology, Federal Information Processing Standards Publications 186-4, 2013, p.88)

Algoritma *Elliptic Curve Diffie Hellman*

Elliptic Curve Diffie Hellman merupakan varian dari Diffie Hellman, algortima ini menggunakan kurva eliptik untuk membangkitkan kunci. Berikut merupakan langkah pertukaran kunci ECDH :

parameter publik :

kurva : $y^2 = x^3 + ax + b$

titik awal R(x, y)

Alice akan memilih bilangan a dalam rentang bilangan [2, |E|-1]

Alice akan menghitung kunci publik : $Ka = a \times R(x, y)$

Bob akan memilih bilangan b dalam rentang bilangan [2, |E|-1]

Bob akan menghitung kunci publik : $Kb = b \times R(x, y)$

Alice menghitung kunci privat : $PK = a \times Kb$

Bob menghitung kunci privat : $PK = b \times Ka$

kunci privat akan sama:

$Ka = a \times R$

$Kb = b \times R$

karena operasi perkalian memiliki sifat komutatif maka dari itu

$a \times Kb = b \times Ka$

$a \times b \times R = b \times a \times R$

Algoritma Advanced Encryption Standard (AES)

Advanced Encryption Standard merupakan algoritma kriptografi yang disetujui oleh *Federal Information Processing Standards Publications* (FIPS PUBS) sebagai algoritma yang dapat digunakan untuk melindungi data elektronik. Algoritma AES adalah cipher blok simetris yang dapat mengenkripsi (*encipher*) dan mendekripsi (*decipher*) informasi. Enkripsi mengubah data menjadi bentuk yang tidak dapat dipahami yang disebut ciphertext; mendekripsi ciphertext mengubah data kembali ke bentuk aslinya, yang disebut plaintext. (National Institute of Standards and Technology, Federal Information Processing Standards Publications 197, 2001)

Spesifikasi Algoritma

Operasi algoritma AES dilakukan pada array byte dua dimensi disebut *State*. State terdiri dari empat baris byte, masing-masing berisi Nb byte, di mana Nb adalah panjang balok dibagi 32. Pada algoritma AES, panjang blok input, blok output dan State adalah 128 bit. Ini diwakili oleh Nb = 4, yang mencerminkan jumlah kata 32-bit (jumlah kolom) di State. panjang Kunci Cipher, K, adalah 128, 192, atau 256 bit. Panjang kunci diwakili oleh Nk = 4, 6, atau 8, yang mencerminkan jumlah kata 32-bit (jumlah kolom) di Kunci Cipher. jumlah ronde yang harus dilakukan selama pelaksanaan algoritma tergantung pada ukuran kunci. Banyaknya putaran dilambangkan dengan Nr, dimana Nr = 10 saat Nk = 4, Nr = 12 saat Nk = 6, dan Nr = 14 saat Nk = 8.

Untuk Cipher dan Inverse Cipher, algoritma AES menggunakan *Round Function* yang terdiri dari empat transformasi berorientasi byte yang berbeda:

1) substitusi byte menggunakan tabel substitusi (S-box) (***SubBytes()***)

2) menggeser baris dari array State dengan offset yang berbeda (***ShiftRows()***)

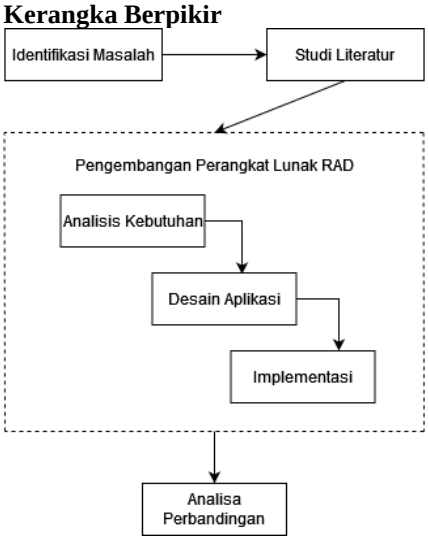
3) mencampur data dalam setiap kolom array State (***MixColumns()***)

4) menambahkan Round Key ke State. (***AddRoundKey()***)

semua putaran Nr identik dengan pengecualian putaran terakhir, yang tidak termasuk transformasi MixColumns().

METODE PENELITIAN

Kerangka Berpikir



Gambar 1. Kerangka Berpikir

Metode Penelitian

Metodologi yang digunakan pada penelitian ini memiliki beberapa tahapan dimulai dari identifikasi masalah, studi literatur, metode pengembangan *rapid application development* (RAD) dan analisa perbandingan.

Identifikasi Masalah

Tahapan identifikasi masalah menjadi dasar penelitian ini. Pada tahapan ini terdiri dari beberapa proses yaitu perumusan masalah, penentuan tujuan dan manfaat, penentuan batasan masalah dan studi pustaka.

Studi Literatur

Tahapan kedua yaitu tahapan studi literatur. Tahapan ini bertujuan untuk melakukan pengumpulan informasi dari buku, artikel, jurnal, dan situs internet yang terkait dengan topik penelitian, informasi yang didapatkan akan digunakan sebagai landasan teori dari penelitian ini.

Metode Pengembangan *Rapid Application Development* (RAD)

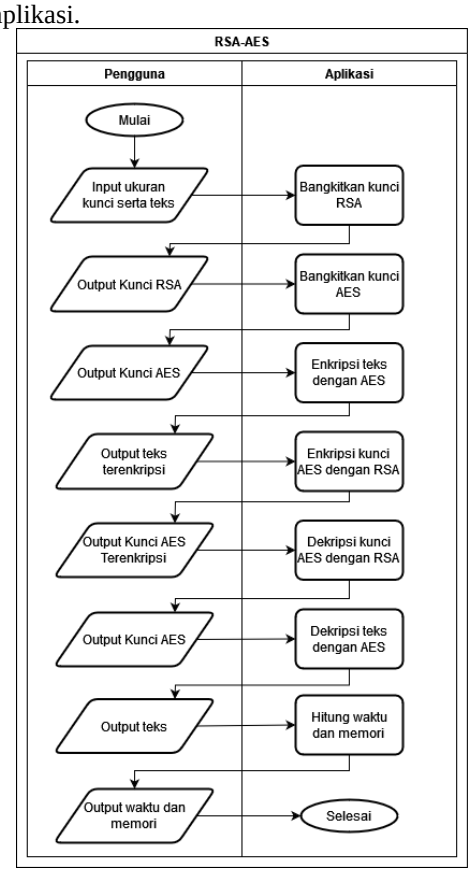
Metode perancangan yang diterapkan pada penelitian ini adalah metode pengembangan Rapid Application Development (RAD). RAD adalah model proses pengembangan perangkat lunak yang bersifat incremental (bertingkat) dan menekankan pada siklus pembangunan yang pendek, singkat dan cepat [9], batasan utama dari metode ini adalah waktu yang singkat, maka dari itu metode ini merupakan metode yang tepat untuk penelitian ini. Adapun tahapan metode yang digunakan adalah sebagai berikut

Analisa Kebutuhan

Tahap ini merupakan tahap dimana peneliti akan menganalisa kebutuhan dari sistem tersebut yaitu, analisis kebutuhan masukan, analisis kebutuhan proses, analisis kebutuhan keluaran, analisis kebutuhan antarmuka, analisis kebutuhan perangat lunak dan analisis kebutuhan perangkat keras. Aplikasi akan terbagi menjadi 4 yaitu RSA-AES, ECDH-AES, Faktorisasi Prima dan Logaritma Diskrit Kurva Eliptik.

Desain Aplikasi

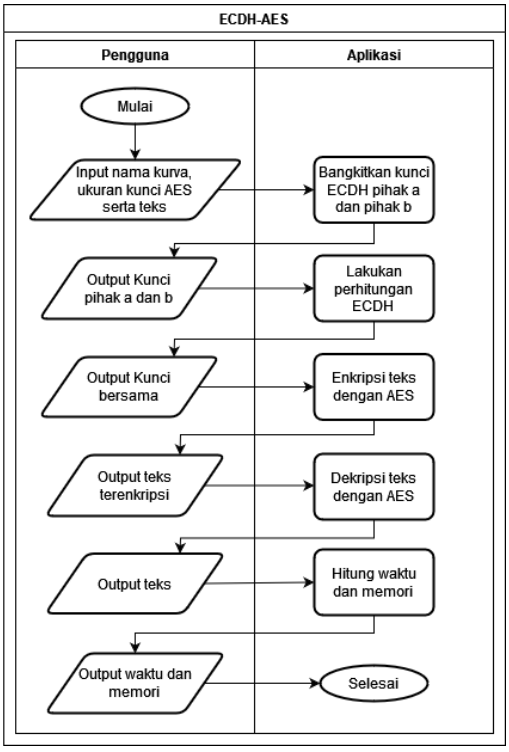
Pada tahap ini akan dilakukan perancangan aplikasi yang akan dibangun, seperti alur kerja aplikasi.



Gambar 1. Alur kerja aplikasi RSA-AES

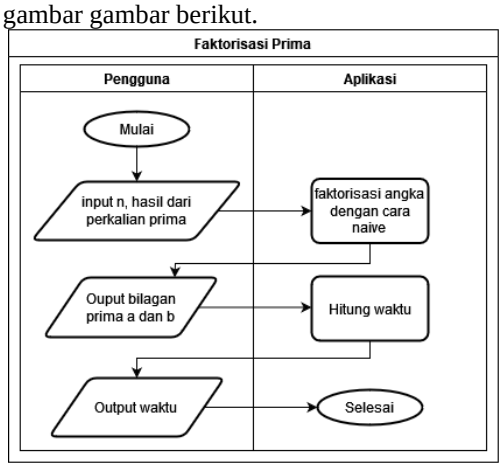
Pada skema RSA-AES, akan dilakukan terlebih dahulu pembangkitan kunci publik dan privat RSA lalu dilakukan pembangkitan kunci AES. Setelah kunci sudah dibangkitkan, plainteks akan dienkripsi menggunakan kunci AES, dan kunci AES tersebut juga akan dienkripsi dengan menggunakan kunci publik RSA, setelah itu pesan berserta kunci AES yang terenkripsi akan didekripsi kembali, dimulai dari mendekripsi kunci AES terlebih dahulu dengan kunci privat RSA lalu setelah kunci AES didapatkan, barulah pesan dapat didekripsi.

Pada skema ECDH-AES, akan dilakukan terlebih dahulu proses perhitungan kunci, setelah perhitungan dilakukan maka akan didapat kan dua buah kunci yang sama, lalu kunci yang sudah didapatkan akan digunakan sebagai kunci AES untuk enkripsi dan dekripsi pesan. Alur kerja sistem atau aplikasi yang akan dikembangkan dapat dilihat pada gambar gambar berikut**:**

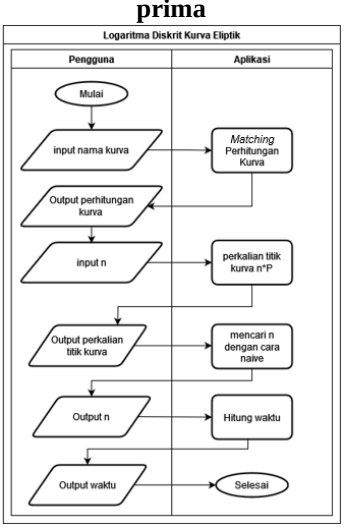


Gambar 2. Alur kerja aplikasi ECDH-AES

Lalu pada penyelesaian permasalahan faktorisasi prima dan logaritma diskrti kurva eliptik akan dilakukan perhitungan dengan cara mencoba semua kemungkinan angka untuk menyelesaikan permasalahan. Aplikasi akan menghitung waktu dan memori yang diperlukan untuk melakukan operasi hybridcryptography dan hanya menghitung waktu pada penyelesaian permasalahan. Alur kerja sistem atau aplikasi yang akan dikembangkan dapat dilihat pada gambar gambar berikut. Alur kerja sistem atau aplikasi yang akan dikembangkan dapat dilihat pada gambar gambar berikut.



Gambar 3. Alur kerja aplikasi faktorisasi prima



Gambar 4. Alur kerja aplikasi logaritma diskrit kurva eliptik

IMPLEMENTASI SISTEM

Tahap ini adalah tahap pengimplementasian dari tahap sebelumnya. Dengan kata lain tahap ini adalah tahap pembangunan atau pembuatan dari aplikasi yang akan dikembangkan dengan

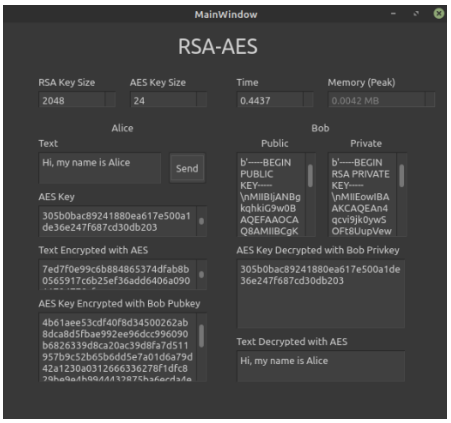
menerjamahkan desain aplikasi kedalam bahasa pemrograman Python. Pada tahap ini akan dibangun 4 aplikasi yang berbeda. Keempat aplikasi ini akan dibangun menggunakan bahasa pemrograman python dengan memanfaatkan library cryptography yang digunanakan untuk proses perhitungan kriptografi dan library pyqt yang digunakan untuk pembuatan tampilan antarmuka atau interface dari aplikasi yang akan dibangun.

Implementasi Aplikasi

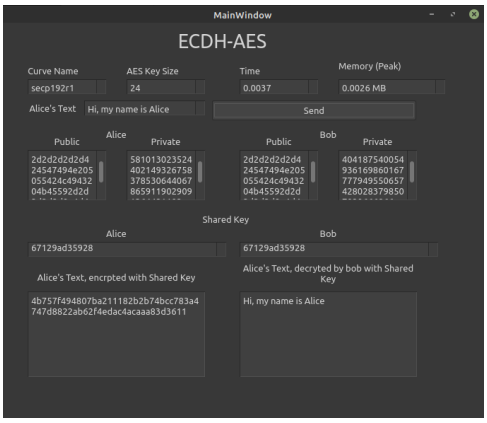
Tahap ini merupakan tahap yang mengimplementasikan semua rancangan aplikasi yang ada di bab sebelumnya, sehingga aplikasi siap digunakan untuk melakukan perbandingan. Berikut merupakan implementasi aplikasi "perbandingan hybridcryptography RSA-AES dan ECDH-AES".

Halaman RSA-AES dan ECDH-AES

Gambar 6 dan 7 merupakan tampilan aplikasi RSA-AES dan ECDH-AES, Pada dua halaman ini dapat dilakukan simulasi skema hybridcryptography RSA-AES dan ECDH-AES dengan cara memasukan ukuran kunci atau jenis kurva dan pesan yang ingin dienkripsi dan dikirim, lalu tekan tombol "send".



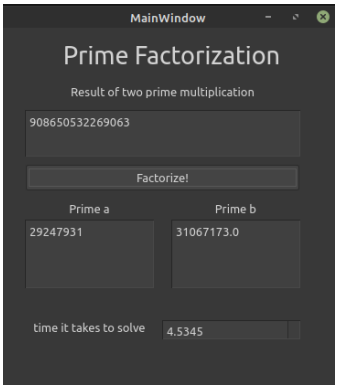
Gambar 6. Tampilan aplikasi RSA-AES



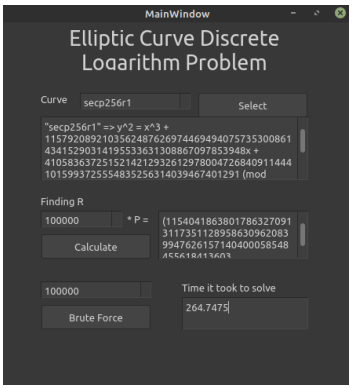
Gambar 7. Tampilan aplikasi ECDH-AES

Halaman Pemfaktoran Prima

Gambar 8 dan 9 merupakan tampilan aplikasi pemfaktoran prima dan logaritma diskrit kurva eliptik, Pada dua halaman ini dapat dilakukan pemfaktoran prima secara naïve dengan cara memasukan hasil perkalian dari dua buah bilangan prima ke tempat yang disediakan, lalu tekan tombol "Factorize!" dan dapat dilakukan penyelesaian permasalahan logaritma diskrit kurva eliptik secara naïve dengan cara masukan nama kurva terlebih dahulu, setelah itu masukan nilai n yang ingin dicari, lalu klik tombol "Brute Force"



Gambar 8. Tampilan aplikasi Faktorisasi Prima



Gambar 9. Tampilan aplikasi Logaritma Diskrit Kurva Eliptik

Analisa Hasil

Untuk mendapatkan hasil, dilakukan pengujian dengan menggunakan teks berukuran 24, 61, 152, dan 317 bytes dengan kunci dan kurva yang berbeda sebanyak 30 kali pada masing masing teks, dari 30 hasil tersebut akan dihitung rata-ratanya dari hasil tersebut. Berikut merupakan Analisa hasil dari pengujian yang telah dilakukan.

Analisis Enkripsi – Dekripsi RSA-AES dan ECDH-AES

Tabel 1 dan 2 merupakan hasil pengujian kombinasi algoritma RSA-AES dan ECDH-AES dengan menggunakan teks dengan ukuran 24, 61, 152, 317 bytes serta kunci dan kurva yang berbeda.

Tabel 1. Hasil Uji RSA-AES

Teks	Kunci RSA	Kunci AES	Waktu	Memory (KB)
24 bytes	1024	128	0.03	2.9
		192	0.03	
		256	0.03	
	2048	128	0.42	4.2
		192	0.46	
		256	0.36	
	3072	128	1.11	5.7
		192	1.48	
		256	1.20	
	4096	128	2.78	7.3
		192	2.77	
		256	2.64	
	5120	128	6.63	8.9
		192	6.63	
		256	6.63	
61 bytes	1024	128	0.03	3.0
		192	0.03	
		256	0.03	
	2048	128	0.43	4.2
		192	0.43	
		256	0.42	
	3072	128	1.30	5.7
		192	1.17	
		256	1.23	
	4096	128	2.80	7.3
		192	3.33	
		256	3.01	
	5120	128	6.94	8.9
		192	7.22	
		256	6.58	
152 bytes	1024	128	0.03	3.4
		192	0.03	
		256	0.03	
	2048	128	0.50	4.2
		192	0.37	
		256	0.44	
	3072	128	1.55	5.7
		192	1.37	
		256	1.27	
	4096	128	3.19	7.3
		192	3.03	
		256	2.48	
	5120	128	8.20	8.9
		192	7.39	
		256	7.78	
317	1024	128	0.03	4.4
		192	0.03	
		256	0.03	
	2048	128	0.45	4.5
		192	0.37	
		256	0.43	
	3072	128	1.22	5.8
		192	1.41	
		256	1.41	
	4096	128	3.20	7.3
		192	2.61	
		256	2.83	
	5120	128	7.18	8.9
		192	5.89	

		256	7.33	

Tabel 2. Hasil Uji ECDH-AES

Teks	Kurva Eliptik	Kunci AES	Waktu	Memory (KB)
24 bytes	Secp192r1	128	0.0037	2.6
		192	0.0037	2.6
		256	0.0037	2.7
	Secp224r1	128	0.0075	2.6
		192	0.0075	2.7
		256	0.0075	2.7
	Secp256r1	128	0.0020	2.6
		192	0.0019	2.7
		256	0.0019	2.7
	Secp384r1	128	0.0100	2.7
		192	0.0100	2.7
		256	0.0100	2.7
	Secp521r1	128	0.0049	2.7
		192	0.0048	2.7
		256	0.0048	2.7
61 bytes	Secp192r1	128	0.0037	2.8
		192	0.0037	2.8
		256	0.0036	2.8
	Secp224r1	128	0.0075	2.8
		192	0.0076	2.8
		256	0.0076	2.8
	Secp256r1	128	0.0019	2.8
		192	0.0019	2.8
		256	0.0019	2.8
	Secp384r1	128	0.0099	2.8
		192	0.0100	2.8
		256	0.0100	2.8
	Secp521r1	128	0.0049	2.8
		192	0.0049	2.9
		256	0.0049	2.9
152 bytes	Secp192r1	128	0.0037	3.2
		192	0.0036	3.2
		256	0.0036	3.2
	Secp224r1	128	0.0073	3.2
		192	0.0073	3.2
		256	0.0073	3.2
	Secp256r1	128	0.0018	3.2
		192	0.0018	3.2
		256	0.0019	3.2
	Secp384r1	128	0.0100	3.2
		192	0.0096	3.2
		256	0.0097	3.2
	Secp521r1	128	0.0048	3.2
		192	0.0048	3.3
		256	0.0048	3.3
317	Secp192r1	128	0.0038	4.1
		192	0.0038	4.1

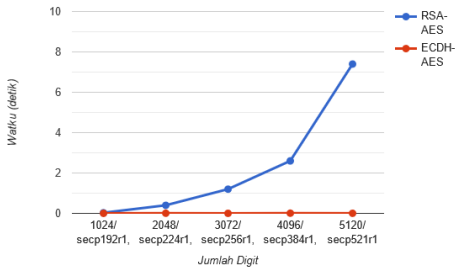
		256	0.0038	4.2
	Secp224r1	128	0.0074	4.1
		192	0.0076	4.2
		256	0.0077	4.2
	Secp256r1	128	0.0019	4.1
		192	0.0020	4.2
		256	0.0019	4.2
	Secp384r1	128	0.0100	4.2
		192	0.0100	4.2
		256	0.0100	4.2
	Secp521r1	128	0.0049	4.2
		192	0.0050	4.2
		256	0.0050	4.2

Dapat dilihat pada tabel 1 bahwa pada hasil waktu proses teks 317 bytes dengan kunci RSA 4096 dan kunci AES 128 bit adalah 3.20 detik, sedangkan pada teks yang sama dengan kunci RSA 5120 dan kunci AES 128 hasil waktu proses adalah 7.18, disini dapat dikatakan bahwa semakin besar kunci algoritma RSA yang digunakan maka waktu yang dibutuhkan dalam proses pertukaran pesan semakin lama, sama halnya dengan kebutuhan memori, kebutuhan memori juga dipengaruhi oleh ukuran kunci RSA, dapat dibuktikan pada hasil uji yang dimana kebutuhan memori pada teks 317 bytes dengan kunci RSA 4096 adalah 7.3 KB sedangkan kebutuhan memori pada teks yang sama dengan kunci RSA 5120 adalah 8.9 KB.

Pada tabel 2 Didapatkan bahwa waktu proses dari skema ini hanya dipengaruhi oleh kurva yang digunakan, pada skema ini semakin besar bit kurva tidak berarti waktu proses semakin lama, dapat dilihat pada kurva secp256r1 yang memiliki waktu proses lebih cepat (0.002) dibandingkan dengan kurva secp224r1 (0.007) dan secp192r1 (0.003), juga dapat dilihat pada kurva secp521r1 yang memiliki waktu proses lebih cepat (0.005) dari secp384r1 (0.010) dan secp224r1. Memori pada skema ini dipengaruhi oleh banyaknya bytes dalam teks, pada table dapat dilihat bahwa pada kebutuhan memori teks 152 bytes dengan kurva secp192r1 dan kunci AES 128 bit adalah 3.2 KB sedangkan kebutuhan memori teks 61 bytes dengan kurva dan kunci AES yang sama adalah 2.8, jadi dapat dikatakan bahwa semakin besar ukuran teks maka kebutuhan memori akan menjadi semakin banyak.

Perbandingan Hasil

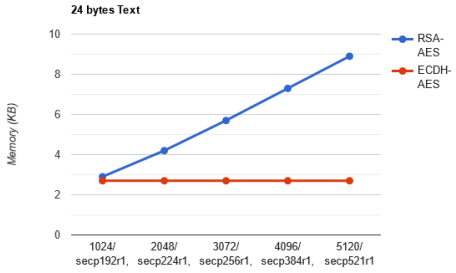
Pada gambar dibawah menunjukan hasil dari uji coba kecepatan enkripsi-dekripsi pada skema RSA-AES dan ECDH-AES dengan menggunakan teks yang berukuran 24, 61, 152, dan 137 *bytes*



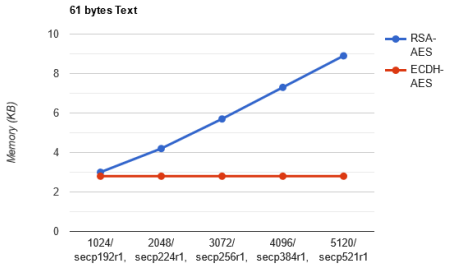
Gambar 10. Perbandingan waktu hasil uji coba

Sesuai dengan grafik diatas, waktu yang dibutuhkan untuk melakukan enkripsi-dekripsi pada kedua skema sangatlah berbeda, pada skema RSA-AES waktu akan semakin lama seiiring bertambahnya bit kunci, pada skema ECDH-AES waktu yang dibutuhkan dibawah angka 1 detik. Dari perbandingan hasil ini menunjukan bahwa kecepatan enkripsi-dekripsi pada skema ECDH-AES jauh lebih cepat dibandingkan dengan skema RSA-AES.

Pada gambar dibawah menunjukan hasil dari uji coba kebutuhan memori enkripsi-dekripsi pada skema RSA-AES dan ECDH-AES dengan menggunakan teks yang berukuran 24 bytes dan 61 bytes.

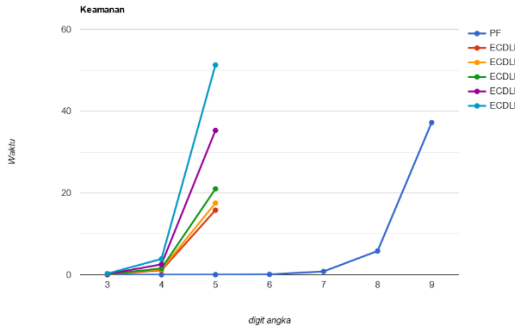


Gambar 11. Perbandingan memori pada teks 24 Bytes



Gambar 12. Perbandingan memori pada teks 61 Bytes

Pada gambar dibawah menunjukan hasil dari uji coba keamanan dengan melakukan bruteforce pada permasalahan faktorisasi prima dan logaritma diskrit pada kurva eliptik.



Gambar 15. Perbandingan waktu pemecahan masalah

SIMPULAN

Dari Hasil penelitian ini, maka dapat diambil beberapa kesimpulan, yaitu pada skenario waktu, ECDH-AES memiliki waktu proses yang lebih cepat dengan hasil uji waktu proses di bawah 1 detik. Pada kebutuhan memori ECDH-AES lebih rendah dibandingkan dengan RSA-AES karena memori terendah dan tertinggi ECDH-AES adalah 2.6 KB dan 4.2 KB sedangkan memori terendah dan tertinggi RSA-AES adalah 2.9 KB dan 8.9 KB. Untuk keamanan, ECDH-AES lebih baik karena permasalahan logaritma diskrit kurva eliptik memerlukan waktu yang lebih lama untuk dipecahkan dibandingkan dengan pemfaktoran prima pada jumlah digit angka yang sama.

DAFTAR PUSTAKA

[1] C. Himawan, T. Wibowo, B. Sulityo, R. Roestam, Y. Wahyu and R. Wahyu, "STUDI Perbandingan Algoritma Rsa Dan Algoritma El-Gamal," 2016.

[2] Z. Subedar and A. Araballi, "Hybrid Cryptography: Performance Analysis of Various Cryptographic Combinations for Secure Communication," *I. J. Mathematical Sciences and Computing*, vol. 4, pp. 35-41, 2020.

[3] A. Hidayat and A. Faizin, "Perbandingan Kriptografi Menggunakan Algoritma Data Encryption Standart (Des) Dan Algoritma Rivest Shamir Adleman (Rsa) Untuk Keamanan Data," *JASIEK*, vol. 1, no. 2, 2019.

[4] A. B. R. P. A. Putra, "Perbandingan Algoritma Rsa Dan Elgamal Pada," *Universitas Sebelas Maret*, 2017.

[5] M. Q. Khairuzzaman, "Implementasi Kriptografi Kunci Publik Dengan Algoritma Rsa," Pontianak, 2019.

[6] S. Suhandinata, R. A. Rizal, D. O. Wijaya, P. Warren and S. Srinjiwi, "Analisis Performa Kriptografi Hybrid Algoritma Blowfish Dan Algoritma Rsa," *JURTEKSI*, vol. 6, no. 1, 2019.

[7] P. Damanik, "Implementasi Algoritma Elliptic Curve Cryptography (ECC) Untuk Penyandian Pesan Pada Aplikasi Chatting Client Server Berbasis Desktop," *JURIKOM*, vol. 6, no. 4, 2019.

[8] Y. E. Housni, "Introduction to the Mathematical Foundations of Elliptic Curve Cryptography.," 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01914807.

[9] W. A. Nazaruddin and E. Hartati, Rekayasa Perangkat Lunak, Bandung: WIDINA, 2022.