BAB III
METODE PENELITIAN

3.1 Objek Penelitian

Objek dalam penelitian ini adalah sebuah aplikasi enkripsi file berbasis *desktop* yang dikembangkan menggunakan bahasa pemrograman Java. Aplikasi ini dirancang untuk memberikan solusi keamanan data pada lingkungan *cloud computing* dengan memanfaatkan algoritma *Advanced Encryption Standard* (AES) sebagai metode enkripsi utama. Aplikasi memiliki tiga fitur utama, yaitu enkripsi file yang berfungsi untuk mengubah file PDF menjadi *ciphertext*, dekripsi file untuk mengembalikan *ciphertext* ke bentuk file asli dengan kunci yang dihasilkan, pengelolaan file di *cloud* termasuk proses unggah (*upload*), unduh (*download*), serta hapus (*delete*) file yang telah terenkripsi melalui integrasi dengan layanan penyimpanan *cloud*.

Penelitian ini berfokus pada penerapan algoritma AES untuk memastikan kerahasiaan dan integritas data yang disimpan di *cloud*, serta mengevaluasi performa aplikasi dari segi efisiensi dan keamanan. AES dipilih karena memiliki keunggulan dalam kecepatan, stabilitas, serta efisiensi memori, sehingga cocok diterapkan pada aplikasi *desktop* ringan yang ditujukan bagi pengguna individu maupun organisasi kecil.

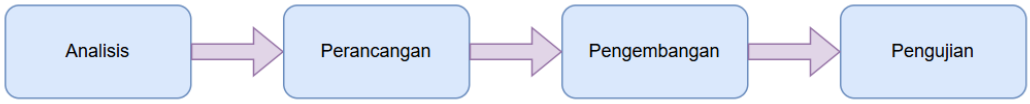
3.2 Metode Penelitian

Penelitian ini berfokus pada pengembangan aplikasi *desktop* berbasis Java yang dirancang untuk melakukan enkripsi dan dekripsi file menggunakan algoritma *Advanced Encryption Standard* (AES), serta mengelola file melalui integrasi dengan layanan *cloud*. Metode yang digunakan dalam penelitian ini adalah *Design and Development* (D&D), yaitu pendekatan sistematis dalam merancang dan

membangun produk teknologi yang bertujuan untuk menghasilkan solusi perangkat lunak yang inovatif, fungsional, dan aplikatif (Richey & Klein, 2007).

Menurut Richey dan Klein (2014), metode D&D terdiri dari dua tipe utama, yaitu *product and tool research* serta *model research*. Keduanya melibatkan proses pengembangan, validasi, dan evaluasi guna menghasilkan solusi yang dapat diterapkan secara praktis. Sementara itu, Sugiyono menjelaskan bahwa metode D&D mencakup langkah-langkah penting untuk menjamin efektivitas produk yang dikembangkan, meliputi tahap analisis kebutuhan, perancangan (*design*), pengembangan (*development*), uji coba (*testing*), dan revisi (*revision*) (Sulindawati dkk., 2023). Pendekatan D&D dipilih karena memberikan kerangka kerja yang terarah dan iteratif, memungkinkan peneliti untuk menghasilkan produk yang tidak hanya bekerja secara teknis, tetapi juga relevan dan berguna dalam konteks keamanan data di lingkungan komputasi awan.

Adapun metode *Design and Development* (D&D) yang digunakan pada penelitian ini melibatkan beberapa tahapan, seperti analisis kebutuhan, perancangan aplikasi, pengembangan aplikasi, dan pengujian aplikasi yang diilustrasikan pada diagram berikut:



Gambar 3.1 Alur Metode Penelitian D&D

3.2.1 Analisis

Tahap analisis kebutuhan merupakan tahap awal yang bertujuan untuk mengidentifikasi dan memahami kebutuhan pengguna serta spesifikasi teknis yang dibutuhkan dalam pengembangan aplikasi. Pada tahap ini dilakukan studi literatur terhadap algoritma kriptografi yang sesuai, yaitu *Advanced Encryption Standard* (AES), serta tinjauan terhadap teknologi *cloud*, khususnya layanan Google Drive

yang akan digunakan untuk penyimpanan file terenkripsi. Analisis ini juga meliputi penentuan *platform* pengembangan (Java), kebutuhan basis data (SQLite), serta integrasi dengan API Google Drive.

Selain dari studi literatur, identifikasi kebutuhan juga dilakukan berdasarkan karakteristik pengguna sistem, yaitu pengguna aplikasi *desktop* yang membutuhkan solusi keamanan file pribadi atau sensitif sebelum disimpan di *cloud*. Oleh karena itu, kebutuhan utama yang ditemukan meliputi fitur enkripsi dan dekripsi file dengan keamanan tinggi, kemudahan penggunaan antarmuka, dan kemampuan mengunggah dan mengunduh file terenkripsi langsung ke Google Drive. Dengan pendekatan ini, aplikasi yang dikembangkan diharapkan dapat memenuhi kebutuhan fungsional dan non-fungsional secara optimal.

3.2.2 Perancangan

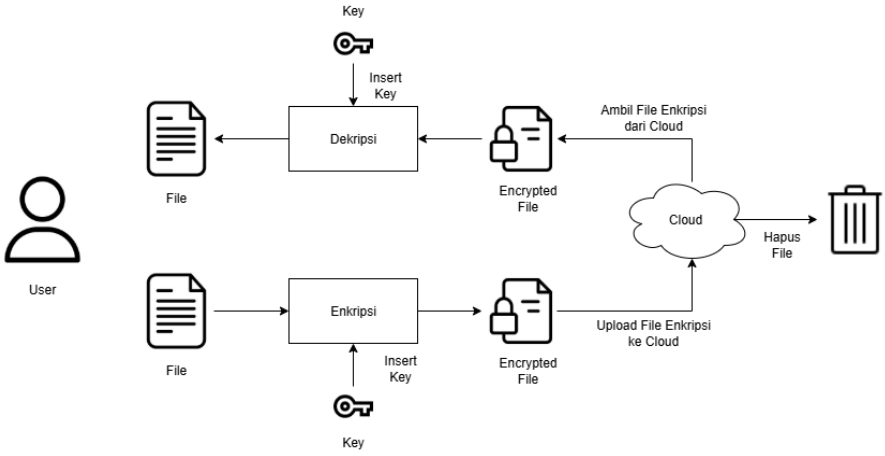
Tahap perancangan merupakan bagian penting dalam pengembangan sistem, di mana semua kebutuhan yang telah diidentifikasi pada tahap analisis diterjemahkan ke dalam bentuk rancangan teknis dan fungsional dari aplikasi. Perancangan sistem ini difokuskan pada pembangunan aplikasi *desktop* berbasis Java yang mampu melakukan proses enkripsi dan dekripsi file menggunakan algoritma *Advanced Encryption Standard* (AES) serta terintegrasi dengan layanan *cloud* Google Drive untuk mendukung pengelolaan file terenkripsi secara *online*. Desain sistem ini mencakup perancangan alur kerja aplikasi secara menyeluruh, mulai dari proses autentikasi pengguna, pemilihan file, proses enkripsi dan dekripsi, hingga fitur pengunggahan, pengunduhan, dan penghapusan file di *cloud*. Perancangan dilakukan dengan pendekatan modular agar setiap fitur dapat dikembangkan dan diuji secara terpisah, namun tetap saling terintegrasi secara keseluruhan dalam sistem.

Beberapa komponen utama yang dirancang pada tahap ini meliputi alur sistem secara umum, *flowchart* dari setiap fitur utama (enkripsi, dekripsi, *upload*, *download*, dan *delete*), serta *use case* diagram yang menggambarkan hubungan antara pengguna dan sistem. Selain itu, dirancang pula tampilan antarmuka pengguna (*user interface*) sementara (*mockup* aplikasi) untuk memberikan

gambaran bagaimana tampilan aplikasi yang nantinya akan dikembangkan. Semua rancangan ini menjadi acuan dalam tahap pengembangan dan implementasi sistem yang akan dilakukan pada tahapan berikutnya.

1. Diagram Arsitektur

Diagram arsitektur sistem (*system architecture diagram*) digambarkan sebagai representasi visual yang memperlihatkan organisasi fundamental dari suatu sistem, termasuk komponen utama, hubungan antarkomponen, dan prinsip–prinsip desain yang mengatur struktur dan evolusi sistem tersebut. Diagram ini mencakup pemetaan fungsi ke dalam komponen perangkat keras dan perangkat lunak, serta interaksi manusia dengan sistem (Wijaksono, 2025). Adapun diagram arsitektur aplikasi secara keseluruhan digambarkan pada gambar 3.2.



Gambar 3.2 Diagram Alur Aplikasi

Gambar 3.2 menggambarkan alur kerja utama dari aplikasi enkripsi file yang dikembangkan, dimulai dari interaksi pengguna dalam melakukan proses enkripsi dan dekripsi file hingga pengelolaan file terenkripsi pada layanan penyimpanan *cloud*. Proses dimulai saat pengguna memilih file (PDF, DOCX, atau XLSX) yang ingin diamankan. File tersebut kemudian diproses oleh modul Enkripsi, yang akan mengubah isi file menjadi bentuk terenkripsi (*ciphertext*) menggunakan algoritma AES-128. Pada tahap ini, pengguna harus memasukkan kunci enkripsi (key) secara manual. Kunci tersebut nantinya juga diperlukan

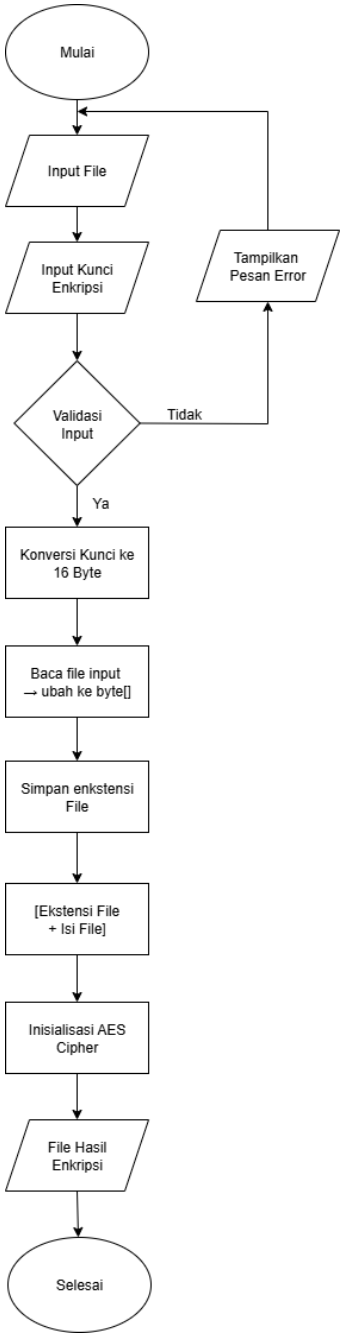
kembali oleh pengguna untuk melakukan proses dekripsi agar file dapat dikembalikan ke bentuk aslinya. File hasil enkripsi kemudian dapat disimpan secara lokal maupun langsung diunggah (*upload*) ke *cloud*, yaitu layanan Google Drive, melalui integrasi API yang telah dikonfigurasi pada aplikasi. Hal ini memungkinkan pengguna menyimpan file dengan aman tanpa khawatir kehilangan atau dibaca oleh pihak tidak berwenang.

Selanjutnya, apabila pengguna ingin mengakses kembali file terenkripsi yang telah tersimpan di *cloud*, mereka dapat menggunakan fitur unduh (*download*) untuk mengambil file tersebut dari Google Drive ke penyimpanan lokal. Setelah file terenkripsi berhasil diunduh, pengguna dapat menjalankan proses dekripsi dengan terlebih dahulu memasukkan kunci enkripsi yang sesuai. Proses dekripsi ini dilakukan oleh modul Dekripsi, yang akan mengubah file terenkripsi kembali menjadi file asli dengan isi yang utuh dan dapat dibuka. Proses dekripsi hanya akan berhasil jika kunci yang dimasukkan sesuai dengan kunci saat proses enkripsi dilakukan, sehingga menjaga integritas dan kerahasiaan data. Setelah file berhasil didekripsi, pengguna dapat kembali mengakses dan menggunakan file tersebut seperti biasa. Selain itu, pengguna juga diberikan beberapa kontrol atas file yang mereka simpan di *cloud*, termasuk kemampuan untuk menghapus file terenkripsi dari Google Drive secara langsung melalui aplikasi. Fitur ini memberikan fleksibilitas dalam pengelolaan file pribadi, terutama ketika pengguna tidak lagi memerlukan file tersebut.

2. *Flowchart* Sistem

Flowchart merupakan representasi grafis yang digunakan untuk menunjukkan langkah-langkah dan alur proses dalam menyelesaikan suatu permasalahan secara terstruktur dan mudah dipahami. Dalam bidang pemrograman dan pengembangan sistem, *flowchart* berperan penting dalam memperjelas tahapan-tahapan yang harus dilalui untuk mencapai tujuan tertentu, sehingga mempermudah pemahaman baik bagi pengembang maupun pengguna (Smrti dkk., 2023).

Flowchart sistem pada Gambar 3.3 menunjukkan alur kerja dari salah satu proses utama dalam sistem, yaitu proses enkripsi file. *Flowchart* ini merepresentasikan logika dasar dari bagaimana sistem beroperasi dalam mengamankan dan mengembalikan data ke bentuk semula dengan memanfaatkan algoritma kriptografi simetris AES.

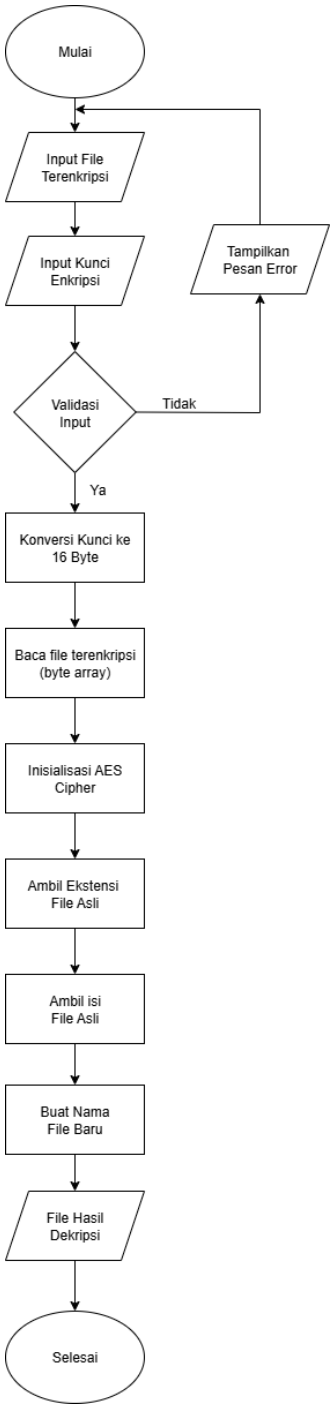


Gambar 3.3 *Flowchart* Proses Enkripsi

Proses enkripsi dimulai dari tahap Mulai, di mana pengguna terlebih dahulu melakukan *Input File* yang akan dienkripsi. Setelah itu, pengguna diminta untuk *Input* Kunci Enkripsi secara manual. Sebelum proses enkripsi dijalankan, sistem melakukan Validasi *Input*. Jika terdapat kesalahan, misalnya file atau kunci tidak sesuai, aplikasi akan menampilkan Pesan *Error* dan pengguna akan diarahkan kembali untuk mengulangi proses *Input*. Jika *input* valid, maka kunci enkripsi akan melalui tahap Konversi Kunci ke 16 *Byte* agar sesuai dengan standar panjang kunci AES-128. Selanjutnya, aplikasi akan membaca file *input* dan mengubahnya ke dalam bentuk *array byte*. Setelah isi file dibaca, sistem juga akan menyimpan ekstensi file agar file yang sudah terenkripsi dapat dikembalikan ke bentuk semula pada proses dekripsi. Kemudian, ekstensi file digabungkan dengan isi file menjadi satu kesatuan data yang siap diproses. Tahap berikutnya adalah Inisialisasi *AES Cipher*, di mana sistem mempersiapkan algoritma AES dengan kunci yang sudah dikonversi. Data gabungan (ekstensi + isi file) kemudian diproses menggunakan algoritma AES untuk menghasilkan File Hasil Enkripsi. Proses diakhiri dengan status Selesai, yang menandakan bahwa file sudah berhasil diamankan dalam bentuk terenkripsi.

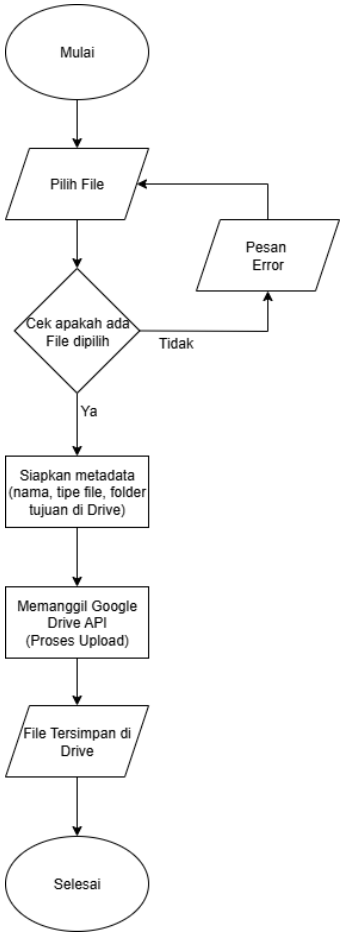
Sementara itu, *flowchart* pada Gambar 3.4 menggambarkan proses dekripsi yang dimulai dari tahap Mulai, di mana pengguna diminta untuk melakukan *Input* File Terenkripsi yang sebelumnya sudah diamankan dengan algoritma AES. Setelah itu, pengguna memasukkan Kunci Enkripsi yang harus identik dengan kunci yang digunakan pada saat proses enkripsi. Sistem kemudian melakukan Validasi *Input*. Jika file atau kunci yang dimasukkan tidak sesuai, aplikasi akan menampilkan Pesan *Error* dan pengguna diarahkan kembali untuk mengulangi proses input. Jika *input valid*, maka kunci enkripsi akan melalui tahap Konversi Kunci ke 16 *Byte* agar sesuai dengan standar panjang kunci AES-128. Selanjutnya, aplikasi akan membaca file terenkripsi dalam bentuk *array byte*, kemudian menjalankan Inisialisasi *AES Cipher* menggunakan kunci yang sudah dikonversi. Pada tahap ini, sistem memproses ciphertext untuk dikembalikan ke bentuk aslinya. Setelah proses dekripsi dilakukan, sistem akan mengambil ekstensi file asli yang sebelumnya disimpan saat enkripsi, kemudian mengambil isi file asli yang sudah berhasil

didekripsi. Agar tidak menimpa file lama, aplikasi juga melakukan tahap Pembuatan Nama File Baru untuk hasil dekripsi. Tahap akhir menghasilkan File Hasil Dekripsi yang identik dengan file sebelum dienkripsi. Proses ditutup dengan status Selesai, menandakan bahwa file berhasil dikembalikan ke bentuk aslinya.



Gambar 3.4 *Flowchart* Proses Dekripsi

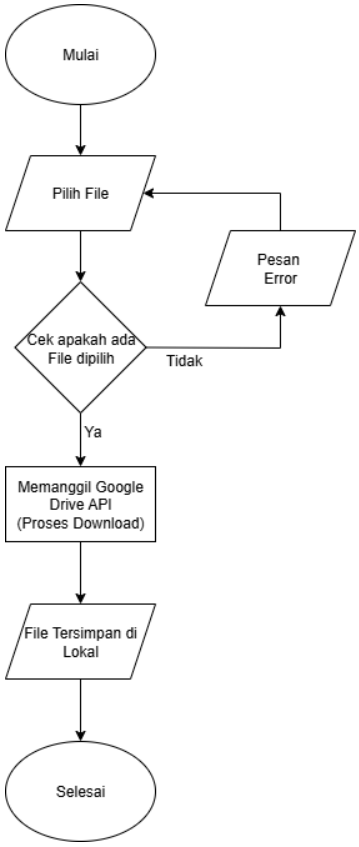
Selain proses enkripsi dan dekripsi, beberapa *flowchart* berikut menggambarkan alur kerja dari tiga fitur penting dalam aplikasi, yaitu *upload*, *download*, dan *delete* file pada layanan *cloud* Google Drive yang diintegrasikan dalam aplikasi. Ketiga fitur ini dirancang untuk memudahkan pengguna dalam mengelola file yang telah dienkripsi secara langsung dari aplikasi *desktop*, tanpa harus membuka *browser* atau mengakses Google Drive secara manual.



Gambar 3.5 *Flowchart* Proses *Upload File* ke *Cloud*

Gambar 3.5 menunjukkan proses *upload file*. Proses ini dimulai dari tahap Mulai, di mana pengguna diminta untuk memilih file yang akan diunggah ke Google Drive, biasanya berupa file hasil enkripsi, atau file lainnya yang ingin pengguna *upload*. Setelah itu, sistem akan melakukan pengecekan apakah ada file yang dipilih. Jika tidak ada file yang dipilih, aplikasi akan menampilkan Pesan *Error* dan pengguna diarahkan kembali untuk memilih file. Jika file sudah dipilih,

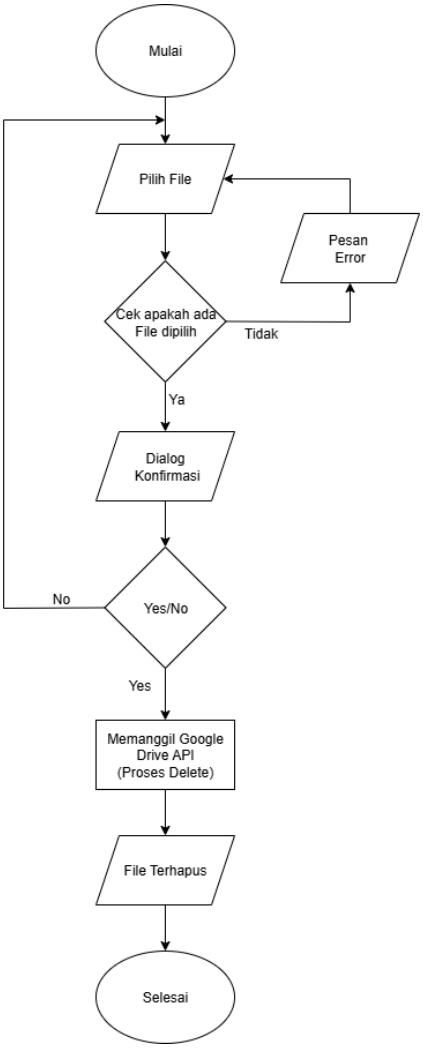
sistem menyiapkan *metadata* yang berisi informasi seperti nama file, tipe file, serta folder tujuan di Google Drive. *Metadata* ini diperlukan agar file dapat tersimpan dengan benar di lokasi yang sesuai pada akun pengguna. Tahap berikutnya adalah memanggil Google Drive API, yaitu proses *upload file* ke *cloud* dengan menggunakan kredensial autentikasi yang telah disediakan. Setelah proses pengiriman selesai, sistem memastikan bahwa file berhasil tersimpan di Google Drive. Proses diakhiri dengan status Selesai, yang menandakan bahwa file sudah berhasil diunggah ke *cloud* dan dapat diakses kembali kapan saja oleh pengguna.



Gambar 3.6 *Flowchart* Proses *Download File* dari *Cloud*

Selanjutnya, *flowchart* pada Gambar 3.6 menggambarkan proses *download file*, yang dimulai dari tahap Mulai, di mana pengguna diminta untuk memilih file yang tersedia di akun Google Drive mereka. Setelah itu, sistem akan melakukan pengecekan apakah file sudah dipilih. Jika tidak ada file yang dipilih, aplikasi akan menampilkan Pesan *Error* dan pengguna diarahkan kembali untuk melakukan pemilihan file. Jika file berhasil dipilih, sistem akan memanggil Google Drive API

untuk melakukan proses *download* file dari *cloud*. Setelah proses unduh selesai, sistem memastikan bahwa file tersimpan di penyimpanan lokal pengguna. Proses diakhiri dengan status Selesai, yang menandakan bahwa file berhasil diambil dari Google Drive dan siap untuk digunakan kembali, termasuk untuk proses dekripsi jika file tersebut masih dalam bentuk terenkripsi.



Gambar 3.7 *Flowchart* Proses *Delete File*

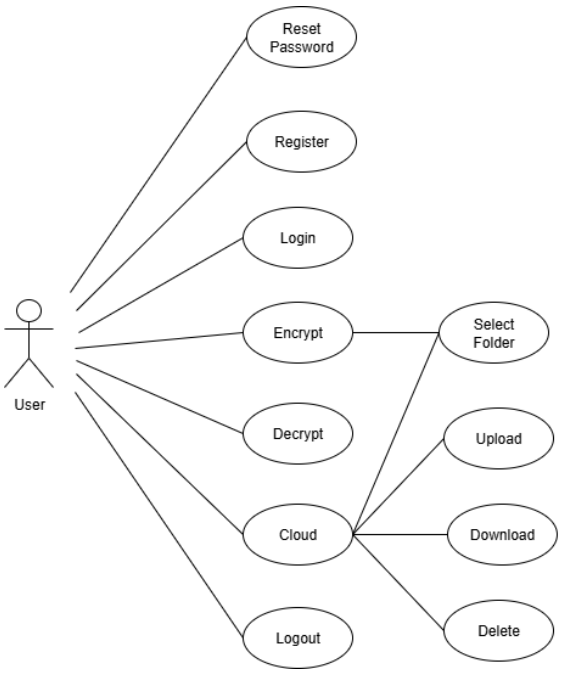
Sementara itu, flowchart pada Gambar 3.7 menunjukkan proses delete file pada cloud. Proses ini dimulai dari tahap Mulai, di mana pengguna diminta untuk memilih file yang tersedia di akun Google Drive. Setelah itu, sistem akan melakukan pengecekan apakah ada file yang dipilih. Jika tidak ada file yang dipilih, aplikasi akan menampilkan Pesan *Error* dan pengguna diarahkan kembali untuk

melakukan pemilihan file. Jika file berhasil dipilih, aplikasi akan menampilkan Dialog Konfirmasi sebagai langkah verifikasi, untuk memastikan apakah pengguna benar-benar ingin menghapus file tersebut. Pada tahap Yes/No, jika pengguna memilih No, maka proses dibatalkan dan sistem kembali ke tahap pemilihan file. Jika pengguna memilih Yes, sistem akan melanjutkan proses penghapusan. Tahap selanjutnya adalah memanggil Google Drive API untuk menjalankan proses delete dengan menggunakan identifikasi file yang sudah dipilih. Jika perintah berhasil dijalankan, sistem akan memberikan status bahwa file telah terhapus dari Google Drive. Proses diakhiri dengan status Selesai, yang menandakan bahwa file benar-benar sudah dihapus dari *cloud storage*.

3. *Use Case Diagram*

Diagram *use case* merupakan representasi visual yang digunakan dalam pemodelan sistem untuk menunjukkan hubungan antara aktor (pengguna) dan sistem yang sedang dirancang. Dalam proses pengembangan perangkat lunak, diagram ini memiliki peranan penting karena membantu pengembang serta pihak-pihak terkait memahami fitur-fitur utama sistem dan bagaimana interaksi pengguna berlangsung (Andriyani dkk., 2022). Oleh karena itu, diagram *use case* tidak hanya berfungsi sebagai ringkasan spesifikasi fungsional, tetapi juga sebagai sarana komunikasi yang efektif antara tim pengembang dengan klien atau pengguna akhir (Sabharwal dkk., 2014).

Use case diagram pada Gambar 3.8 menggambarkan interaksi antara aktor utama, yaitu pengguna (*User*), dengan sistem aplikasi enkripsi file yang dikembangkan. Diagram ini digunakan untuk mendeskripsikan fungsi-fungsi utama yang tersedia dalam sistem dan bagaimana pengguna dapat berinteraksi dengan masing-masing fungsi tersebut. Aktor *User* merepresentasikan semua pengguna sistem yang memiliki hak akses penuh terhadap fitur-fitur aplikasi setelah melakukan proses registrasi dan *login*.



Gambar 3.8 *Use Case Diagram*

Proses awal interaksi dimulai dengan fitur *Login*, yang mana apabila sebelumnya pengguna sudah memiliki akun, maka dapat langsung memasukkan *username* maupun *password* untuk masuk ke menu utama aplikasi. Apabila pengguna belum memiliki akun dapat langsung melakukan registrasi pada halaman *register*, yang memungkinkan pengguna baru untuk membuat akun di dalam aplikasi. Setelah registrasi berhasil, pengguna dapat kembali masuk ke sistem melalui fitur *Login*, yang akan memverifikasi kredensial terhadap data yang tersimpan dalam basis data lokal SQLite. Proses login maupun register ini menjadi prasyarat agar pengguna dapat mengakses seluruh fitur lainnya dalam aplikasi. Apabila pengguna lupa atau ingin mengganti *password* akun mereka, dapat langsung masuk ke menu *reset password*, yang dapat diakses lewat menu login. Setelah berhasil masuk, pengguna dapat menggunakan fitur utama yaitu *Encrypt* dan *Decrypt*. Fitur *Encrypt* memungkinkan pengguna memilih file berekstensi PDF, DOCX, atau XLSX untuk dienkripsi menggunakan algoritma AES 128-bit. File yang telah dienkripsi kemudian disimpan di direktori lokal sebagai file *ciphertext,* maupun *upload* langsung ke *drive* akun pengguna, dengan mencentang *check box Upload to Drive* yang terdapat pada menu *Encrypt*. Di sisni pengguna juga dapat

mengganti folder tujuan pada drive yang ingin digunakan sebagai tempat penyimpanan di *cloud*. Selanjutnya, fitur *Decrypt* digunakan untuk mengembalikan file terenkripsi ke bentuk aslinya menggunakan kunci yang sama, dengan memastikan bahwa proses dekripsi hanya berhasil apabila data dan kunci valid.

Selain itu, aplikasi juga menyediakan fitur *Cloud* untuk integrasi dengan layanan penyimpanan *cloud* Google Drive. Fitur ini menjadi pusat dari tiga aktivitas turunan lainnya, yaitu *Upload*, *Download*, *Delete,* dan *Change Folder*. Pada fitur *Upload*, pengguna dapat mengunggah file terenkripsi ke akun Google Drive mereka menggunakan API dan sistem autentikasi berbasis *Email User*. Fitur *Download* memungkinkan pengguna mengambil kembali file dari Google Drive ke penyimpanan lokal, untuk kemudian bisa didekripsi. Fitur *Delete* berfungsi untuk menghapus file dari akun Google Drive secara langsung melalui aplikasi, tanpa perlu membuka platform Google Drive secara manual. Sementara itu, apabila pengguna ingin mengganti folder *default* yang digunakan untuk penyimpanan *cloud*, pengguna dapat menggunakan fitur *Change Folder* yang nantinya akan diarahkan ke menu *Select Folder*. Terakhir, aplikasi juga menyediakan fitur *Logout*, yang memungkinkan pengguna keluar dari sesi penggunaan aplikasi dengan aman. Proses ini memastikan bahwa akses ke file dan layanan cloud tidak dapat dilakukan oleh pihak lain setelah pengguna selesai menggunakan aplikasi.

Secara keseluruhan, *use case diagram* ini merepresentasikan seluruh fitur kunci dari aplikasi yang dikembangkan, sekaligus menunjukkan alur kerja sistem yang fokus pada keamanan, enkripsi, dan manajemen file *cloud* secara praktis melalui platform *desktop*.

3.2.2.3 Perancangan Tampilan Aplikasi

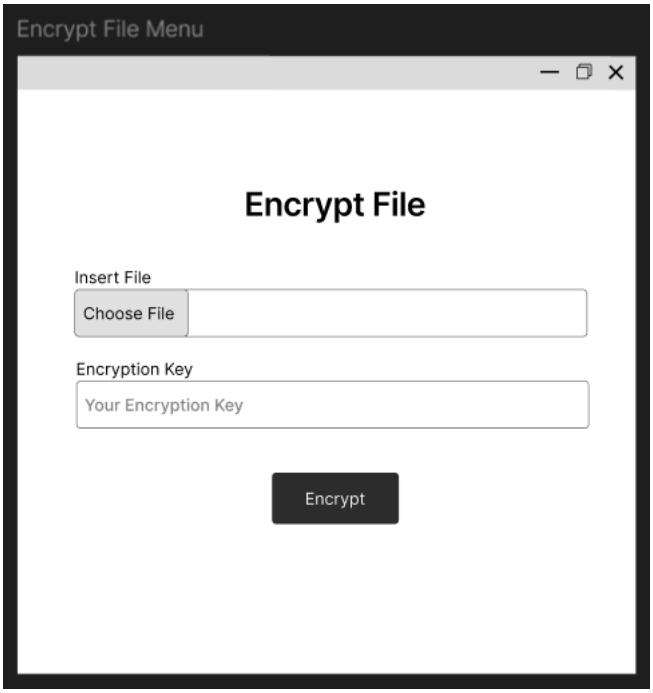
Perancangan tampilan aplikasi merupakan tahap penting dalam proses pengembangan, karena menjadi antarmuka utama antara pengguna dan sistem. Tampilan yang dirancang harus mampu merepresentasikan seluruh fungsi utama aplikasi secara jelas, sederhana, dan mudah digunakan oleh pengguna akhir. Pada tahap ini, dilakukan pembuatan rancangan awal (*mockup*) dari antarmuka pengguna

untuk memvisualisasikan bagaimana alur interaksi pengguna dengan fitur-fitur yang disediakan dalam aplikasi, seperti tampilan menu utama, enkripsi, dekripsi, serta pengelolaan file di layanan *cloud*. Setiap tampilan disesuaikan dengan alur sistem yang telah dirancang sebelumnya, serta mempertimbangkan prinsip *user experience* (UX) agar pengguna dapat menjalankan proses secara intuitif. Pada subbab ini akan ditampilkan dan dijelaskan masing-masing *mockup* antarmuka dari fitur utama dalam aplikasi, sebagai acuan dalam tahap implementasi program pada fase pengembangan berikutnya.

Mockup pada Gambar 3.9 merupakan tampilan menu utama (*home*) dari aplikasi yang diberi nama EncSafe. Halaman ini menjadi tampilan awal yang muncul setelah pengguna berhasil melakukan proses login ke dalam sistem. Tujuan dari halaman ini adalah memberikan navigasi utama yang sederhana dan jelas, sehingga pengguna dapat langsung memilih fungsi inti dari aplikasi sesuai kebutuhan. Di bawahnya, terdapat tiga tombol utama yang masing-masing mengarahkan pengguna ke fitur inti aplikasi, yaitu *Encrypt File* untuk menuju ke halaman proses enkripsi file, *Decrypt File* untuk menuju ke halaman dekripsi file, dan *Cloud Storage*, yang mengarahkan pengguna ke halaman pengelolaan file di layanan Google Drive, termasuk unggah, unduh, dan hapus file.

Gambar 3.9 *Mockup* Menu Utama

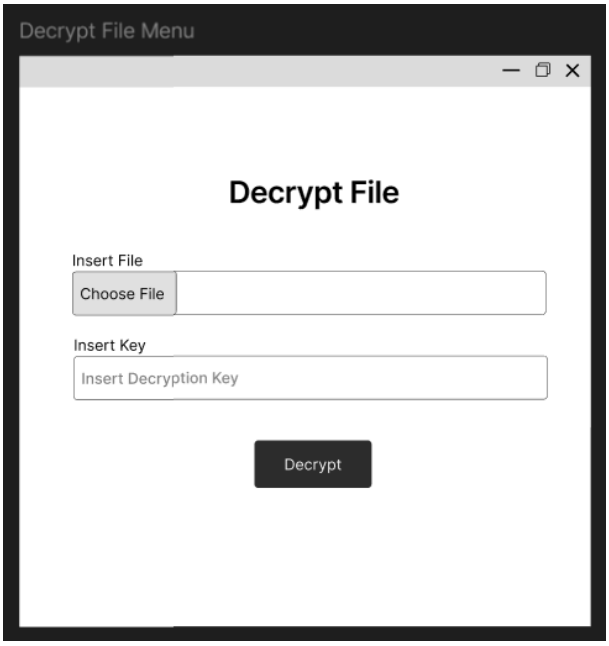
Gambar 3.10 merupakan *mockup* dari halaman *Encrypt File*, yaitu salah satu fitur utama dalam aplikasi EncSafe yang digunakan untuk melakukan proses enkripsi file. Tampilan ini dirancang dengan tujuan memberikan alur kerja yang sederhana dan mudah dipahami oleh pengguna, khususnya dalam mengamankan file pribadi atau sensitif sebelum disimpan secara lokal maupun di *cloud*. Terdapat dua komponen input utama. Komponen pertama adalah field "*Insert File*", dilengkapi dengan tombol "*Choose File*" yang memungkinkan pengguna untuk memilih file dari direktori lokal mereka. File yang dipilih akan ditampilkan di kolom input sebagai referensi visual. Komponen kedua adalah field "*Encryption Key*", yaitu tempat bagi pengguna untuk menerima kunci enkripsi yang akan dihasilkan dalam proses pengubahan isi file menjadi bentuk terenkripsi (*ciphertext*). Penggunaan kunci ini sangat penting, karena file hanya dapat didekripsi kembali jika kunci yang dimasukkan pada proses dekripsi sesuai dengan kunci enkripsi yang digunakan di sini. Di bagian bawah, terdapat tombol "*Encrypt*" yang ketika ditekan akan mengeksekusi proses enkripsi berdasarkan input yang telah dimasukkan. Desain tombol dan komponen lainnya dibuat dengan tampilan bersih dan fungsional, menyesuaikan konsep antarmuka yang sederhana namun efisien.



Gambar 3.10 *Mockup* Menu Encrypt

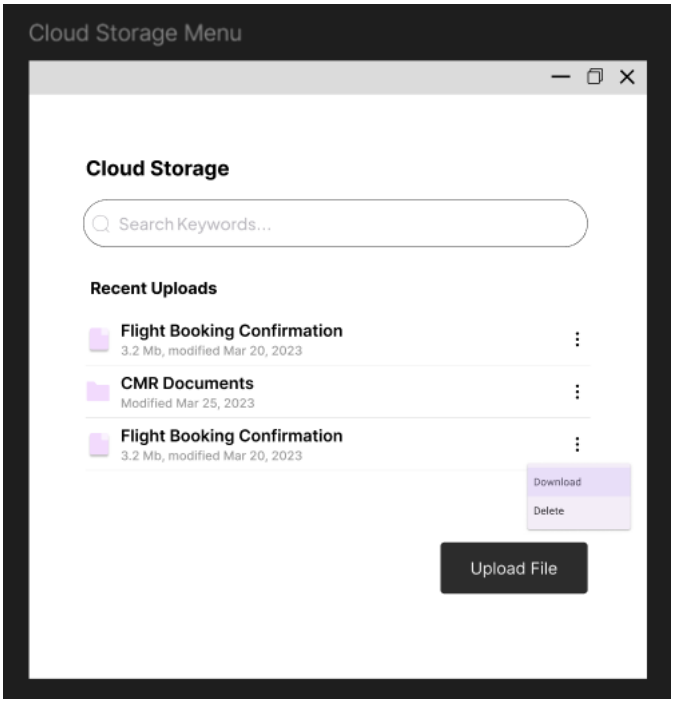
Mockup pada Gambar 3.11 merupakan tampilan halaman *Decrypt File*, yaitu fitur yang digunakan untuk mengembalikan file terenkripsi ke bentuk aslinya (*plaintext*) menggunakan kunci dekripsi yang sesuai. Halaman ini menjadi salah satu komponen penting dalam aplikasi EncSafe, karena memungkinkan pengguna untuk mengakses kembali isi file yang telah diamankan melalui proses enkripsi sebelumnya. Terdapat dua input utama yang diperlukan untuk menjalankan proses dekripsi. Komponen pertama adalah "*Insert File*", yang dilengkapi dengan tombol "*Choose File*". Tombol ini memungkinkan pengguna memilih file terenkripsi (*ciphertext*) yang sebelumnya telah mereka hasilkan melalui proses enkripsi di dalam aplikasi. File yang dipilih akan ditampilkan pada kolom input sebagai referensi, membantu pengguna memastikan file yang akan didekripsi sudah benar. Komponen kedua adalah "*Insert Key*", yaitu kolom untuk memasukkan kunci dekripsi yang harus sama dengan kunci yang dihasilkan saat proses enkripsi sebelumnya. Kunci ini menjadi elemen krusial, karena apabila tidak sesuai, proses dekripsi akan gagal dan file tidak dapat dikembalikan ke bentuk semula. Di bagian bawah, terdapat tombol "*Decrypt*" yang akan memulai proses dekripsi begitu file

dan kunci dimasukkan. Setelah proses selesai, sistem akan menghasilkan file hasil dekripsi dan menyimpannya di direktori lokal.



Gambar 3.11 *Mockup* Menu Decrypt

Terakhir pada Gambar 3.12 merupakan *mockup* dari fitur *Cloud Storage* pada aplikasi enkripsi desktop yang dirancang untuk memberikan kemudahan kepada pengguna dalam mengelola file yang telah dienkripsi. Fitur ini merupakan salah satu bagian penting dalam sistem, karena selain menyimpan file secara lokal, pengguna juga memiliki opsi untuk mengunggah file terenkripsi ke penyimpanan awan (*cloud storage*). Tersedia sebuah kolom pencarian dengan *placeholder* "*Search Keywords...*" yang berfungsi untuk memudahkan pengguna mencari file tertentu dari daftar file yang telah diunggah. Selain itu, terdapat daftar file yang baru saja diunggah oleh pengguna ke cloud. Informasi yang ditampilkan meliputi nama file, ukuran file, dan tanggal modifikasi file. Di samping setiap file terdapat ikon tiga titik vertikal yang berfungsi sebagai tombol menu aksi. Ketika diklik, menu ini akan menampilkan dua opsi penting, yaitu *Download* dan *Delete* yang berguna untuk keperluan unduh ulang maupun penghapusan file di *cloud*. Terakhir terdapat tombol berlabel "*Upload File*" yang berfungsi untuk mengunggah file baru ke penyimpanan *cloud*.



Gambar 3.12 *Mockup* Menu Cloud

3.2.3 Pengembangan

Tahap pengembangan pada penelitian ini menggunakan pendekatan *Agile Development*, yaitu metode pengembangan perangkat lunak secara iteratif dan bertahap yang terbagi ke dalam beberapa *sprint*. Metode ini dipilih karena memberikan fleksibilitas tinggi terhadap perubahan kebutuhan yang mungkin terjadi selama proses pengembangan, karena metode ini memungkinkan pengembang untuk melakukan iterasi, menerima umpan balik dengan cepat, dan menyesuaikan proyek sesuai dengan kebutuhan yang baru muncul (Prasetyo & Nugraha, 2023). Setiap *sprint* difokuskan pada pembangunan satu atau beberapa fitur inti dalam aplikasi. Misalnya, *sprint* awal diarahkan pada pengembangan modul login dan enkripsi file lokal, sementara *sprint* berikutnya mencakup integrasi layanan *cloud*, pengujian fungsionalitas, serta perbaikan dari umpan balik tahap sebelumnya.

Implementasi awal dimulai dengan membangun fitur *register* dan *login* yang terhubung ke basis data lokal menggunakan SQLite. Sistem ini memungkinkan pengguna untuk menyimpan informasi akun dan mengakses fitur-

fitur aplikasi secara personal. Setelah autentikasi berhasil, pengguna dapat mengakses fitur utama berupa enkripsi dan dekripsi file. Proses enkripsi dilakukan menggunakan algoritma *Advanced Encryption Standard* (AES) 128-bit, yang merupakan algoritma kriptografi simetris dengan tingkat keamanan tinggi dan efisiensi komputasi yang baik (Azhari dkk., 2022). Dalam implementasinya, digunakan pustaka *Java Cryptography Architecture* (JCA), di mana pengguna dapat memilih file dan memasukkan kunci enkripsi. File yang telah dienkripsi kemudian disimpan dalam format *ciphertext* pada direktori lokal maupun direktori pada *cloud* secara opsional. Sebaliknya, proses dekripsi dilakukan dengan memasukkan file terenkripsi dan kunci yang sesuai, untuk mengembalikan file ke bentuk aslinya.

Setelah fitur enkripsi lokal selesai dikembangkan, dilanjutkan dengan integrasi ke layanan Google Drive untuk memungkinkan penyimpanan *cloud*. Proses ini dilakukan menggunakan Google Drive API, dengan sistem autentikasi berbasis OAuth 2.0 melalui akun *Email* milik masing-masing pengguna. Setelah koneksi berhasil, pengguna dapat melakukan proses unggah (*upload*) file terenkripsi ke *cloud*, serta mengunduh (*download*) file dari *cloud,* menghapus (*delete*) file, serta mengubah folder (*select folder*) tujuan dari akun Google Drive mereka secara langsung melalui antarmuka aplikasi.

Selama proses pengembangan, digunakan berbagai perangkat lunak pendukung untuk mendukung dalam proses pengembangan aplikasi. Aplikasi dikembangkan menggunakan NetBeans IDE sebagai lingkungan utama yang mendukung desain antarmuka pengguna berbasis *Java Swing*. Bahasa pemrograman yang digunakan adalah Java dengan dukungan *Java Development Kit* (JDK). SQLite dipilih sebagai *database* aplikasi karena ringan dan tidak memerlukan instalasi server tambahan, sehingga cocok untuk aplikasi *desktop*. Pengelolaan pada perbaruan kode dilakukan secara manual menggunakan pendekatan modular *versioning*, untuk memastikan perubahan pada setiap fitur dapat dilacak dan diuji dengan lebih mudah, serta meminimalkan potensi konflik antar bagian aplikasi.

3.2.4 Pengujian

Pengujian hasil dilakukan untuk mengevaluasi fungsionalitas aplikasi, efektivitas enkripsi, serta kekuatan algoritma yang digunakan dalam pengembangan aplikasi enkripsi ini. Pengujian yang dilakukan meliputi pengujian fungsionalitas sistem dengan metode *blackbox*, serta beberapa pengujian keamanan dan efisiensi algoritma kriptografi, seperti pengujian *Pearson Correlation*, uji Integritas file, dan perbandingan ukuran file terenkripsi.

3.2.4.1. Pengujian Fungsional Aplikasi (Blackbox Testing)

Pengujian fungsional dilakukan menggunakan metode *blackbox testing*, yaitu dengan memfokuskan pengujian pada *output* yang dihasilkan oleh sistem terhadap *input* tertentu tanpa memperhatikan struktur kode program. Tujuan dari pengujian ini adalah untuk memastikan bahwa semua fitur utama dalam aplikasi berjalan sesuai dengan kebutuhan pengguna dan spesifikasi yang telah dirancang sebelumnya. Sebagai gambaran mengenai cara kerja metode pengujian *blackbox* dapat dilihat pada Gambar 3.12.

Gambar 3.12 Metode Pengujian *Black Box*

Pengujian ini akan dilakukan pada setiap menu yang terdapat pada aplikasi, seperti menu *login*, *register*, *reset password*, *encrypt*, *decrypt*, serta menu *cloud* yang terintegrasi dengan Google Drive untuk proses unggah (*upload*), unduh (*download*), penghapusan (*delete*) file, serta ubah folder (*change folder*). Pada menu *login* dan *register*, pengujian dilakukan dengan skenario mengisi semua *field* dengan data valid maupun mengosongkan salah satu *field*, untuk memastikan sistem dapat membedakan input yang lengkap dan tidak lengkap serta menampilkan pesan validasi yang sesuai. Pada menu *reset password*, pengujian mencakup kondisi ketika *password* baru dan konfirmasi *password* sama, tidak sama, atau terdapat *field*

yang dikosongkan, untuk memastikan sistem memberikan pesan peringatan yang tepat. Pengujian pada menu *encrypt* dilakukan dengan memilih file dan memasukkan kunci enkripsi, termasuk skenario validasi panjang kunci (maksimal 16 karakter), jenis file yang didukung (pdf, docx, xlsx), serta opsi unggah hasil enkripsi ke *cloud*. Menu *decrypt* diuji untuk memastikan file terenkripsi dapat dikembalikan ke bentuk semula. Sementara itu, pada menu *cloud*, pengujian aplikasi mencakup pengujian fungsionalitas fitur unggah (*upload*), unduh (*download*), hapus (*delete*), ubah folder *default* (*change folder*), serta penyegaran daftar file (*refresh*). Jika semua proses berlangsung tanpa ada kesalahan dan hasilnya sesuai, maka fitur tersebut dinyatakan berhasil.

3.2.4.2. Pengujian Kinerja Algoritma AES

Dalam pengembangan aplikasi ini, pengujian kinerja algoritma AES dilakukan melalui tiga pendekatan, yaitu uji *pearson correlation*, uji integritas file menggunakan hash SHA-256, dan perbandingan ukuran file sebelum dengan sesudah proses enkripsi-dekripsi. Ketiga metode pengujian ini bertujuan untuk mengevaluasi sejauh mana algoritma AES mampu menjaga keamanan, keutuhan, dan efisiensi data secara menyeluruh.

1. Pengujian *Pearson Correlation*

Pengujian *Pearson Correlation* digunakan untuk mengukur tingkat hubungan linear antara data *byte* file asli (*plaintext*) dengan file hasil enkripsi (*ciphertext*). Tujuannya adalah untuk memverifikasi bahwa hasil enkripsi tidak memiliki hubungan yang signifikan dengan data asli, sehingga pola data asli tidak dapat ditebak dari *ciphertext* (Stallings, 2017). Nilai korelasi *Pearson* (r) dihitung menggunakan rumus:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Di mana:

X_i adalah nilai byte ke-*i* dari file asli,

Y_i adalah nilai byte ke-*i* dari file terenkripsi,

$\bar{X}$ adalah rata-rata nilai byte file asli,

$\bar{Y}$ adalah rata-rata nilai byte file terenkripsi,

n adalah jumlah total byte.

Pengujian ini akan dilakuan pada 50 sample file yang mencakup berbagai jenis file, seperti PDF, DOCX, dan XLSX, dengan beragam ukuran dan konten. Proses analisis pada pengujian dilakukan dalam beberapa tahap:

1. Setiap file dienkripsi menggunakan algoritma AES melalui aplikasi yang telah dikembangkan, dan dengan kunci yang dihasilkan pada aplikasi tersebut.
2. Dari hasil enkripsi (*ciphertext*) setiap file, dibuat pasangan *byte* yang berurutan—misalnya byte ke-1 dengan ke-2, ke-2 dengan ke-3, dan seterusnya—untuk membentuk model korelasi antar *byte* yang saling berdekatan dalam *ciphertext*.
3. Selanjutnya, dihitung nilai koefisien korelasi *Pearson* (*r*) dan *p-value* untuk masing-masing *ciphertext*.

Evaluasi dilakukan berdasarkan dua kriteria, apabila nilai |*r*| mendekati nol, maka dianggap menunjukkan tidak adanya pola linear dalam *ciphertext*, yang merupakan indikator difusi atau penyebaran yang baik; sementara *p-value* digunakan untuk menilai signifikansi statistik dari korelasi tersebut.

2. Pengujian Integritas File menggunakan SHA-256

Pengujian integritas file bertujuan untuk memastikan bahwa isi file tidak berubah atau dimodifikasi, baik secara sengaja (misalnya serangan *man-in-the-middle*) maupun tidak sengaja (misalnya kerusakan saat transmisi atau penyimpanan). Salah satu metode paling populer untuk melakukan verifikasi

integritas adalah menggunakan hash kriptografis. Menurut Alasmary et al. (2020), fungsi hash kriptografis merupakan sebuah fungsi matematis satu arah yang mengubah data dengan ukuran sembarang menjadi string berukuran tetap. Hasil transformasi ini secara unik dapat merepresentasikan data asli, sehingga setiap perubahan yang tidak sah pada data dapat dengan mudah terdeteksi. Proses pengujian integritas file dengan SHA-256 dilakukan melalui beberapa Langkah, diantaranya.

1. Menghasilkan nilai hash SHA-256 dari 50 file asli (*plaintext*).
2. Melakukan proses enkripsi dan dekripsi menggunakan AES.
3. Menghasilkan nilai hash SHA-256 dari 50 file hasil dekripsi.
4. Membandingkan kedua nilai hash.

Jika nilai *hash* sebelum dan sesudah proses identik, maka dapat disimpulkan bahwa file tidak mengalami perubahan dan integritasnya terjaga. Sebaliknya, jika nilai *hash* berbeda, berarti ada perubahan dalam isi file. Hal ini dimungkinkan karena fungsi hash bersifat deterministik dan sangat sensitif terhadap perubahan, sehingga bahkan perubahan satu bit pun dalam file akan menghasilkan nilai hash yang benar-benar berbeda.

3. Pengujian Perbandingan Ukuran File

Pengujian ini bertujuan untuk mengevaluasi efisiensi pada hasil proses enkripsi-dekripsi dari sisi ukuran file. Dalam hal ini, algoritma AES yang baik seharusnya tidak mengubah ukuran file secara signifikan, khususnya saat proses dekripsi yang mengembalikan file ke bentuk aslinya. Langkah pengujian dilakukan dengan:

1. Mencatat ukuran asli dari file sebelum dienkripsi.
2. Melakukan proses enkripsi dan mencatat ukuran file *ciphertext*.
3. Melakukan dekripsi dan mencatat ukuran file hasil dekripsi.

Pengujian akan dilakukan kepada 50 file dengan format berbeda (PDF, DOCX, XLSX) akan digunakan sebagai sampel pengujian. Jika pada hasil pengujian menunjukkan bahwa setiap ukuran file sebelum dienkripsi, setelah

dienkripsi, dan setelah didekripsi tetap sama, maka hal ini menunjukkan bahwa algoritma AES mampu mempertahankan ukuran file saat digunakan untuk mengenkripsi berbagai jenis file.