BAB 3
METODOLOGI PENELITIAN

Metode-metode yang digunakan untuk mengimplementasikan *Galois/Counter Mode* (GCM) dengan metode Rijndael adalah sebagai berikut.

3.1 Analisis Kebutuhan

Berdasarkan latar belakang masalah yang sudah dibuat, ditemukan kebutuhan untuk membantu perusahaan agar dapat menjaga keamanan data pada sistem dengan mengimplementasikan metode enkripsi untuk mencegah terjadinya kebocoran data.

3.2 Studi Literatur

Dalam studi literatur, telah dipelajari teori-teori yang dapat membantu dalam proses implementasi *Galois/Counter Mode* (GCM) dengan metode rijndael. Teori-teori tersebut adalah teori algoritma Rijndael (AES), *Galois/Counter Mode* (GCM), AES-GCM, *Vulnerability Assessment*, *OWASP Top Ten*, *Testing for Weak Encryption*, dan *OWASP Risk Rating*. Pembelajaran ini dilakukan dengan membaca berbagai sumber bacaan seperti buku, jurnal, dan skripsi.

3.3 Implementasi

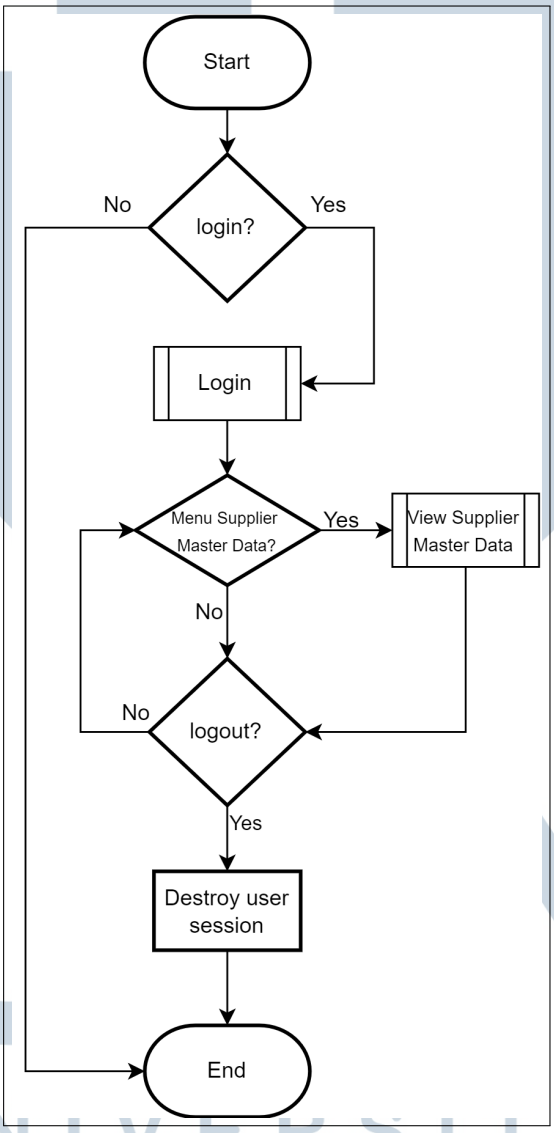
Algoritma AES-GCM akan diimplementasikan pada sistem pengadaan berbasis *web* sesuai dengan dokumentasi pembuatan AES dan GCM. Implementasi AES-GCM ini akan melindungi data sensitif yang berkaitan dengan data *supplier* sehingga data tersebut tidak bisa diambil/dicuri oleh orang yang tidak bertanggung jawab. Bahasa pemrograman yang digunakan dalam sistem ini adalah Go dan HTML.

3.3.1 Flowchart

Berikut merupakan *flowchart* dari sistem pengadaan berbasis *web* yang mengimplementasikan AES-GCM.

1. *Flowchart* Sistem

Gambar 3.1 merupakan *flowchart* dari keseluruhan sistem. Pengguna akan diminta untuk *login* untuk dapat mengakses sistem. Setelah *login*, pengguna dapat mengakses menu *supplier master data*. Jika pengguna sudah selesai menggunakan sistem, pengguna dapat melakukan *logout*.

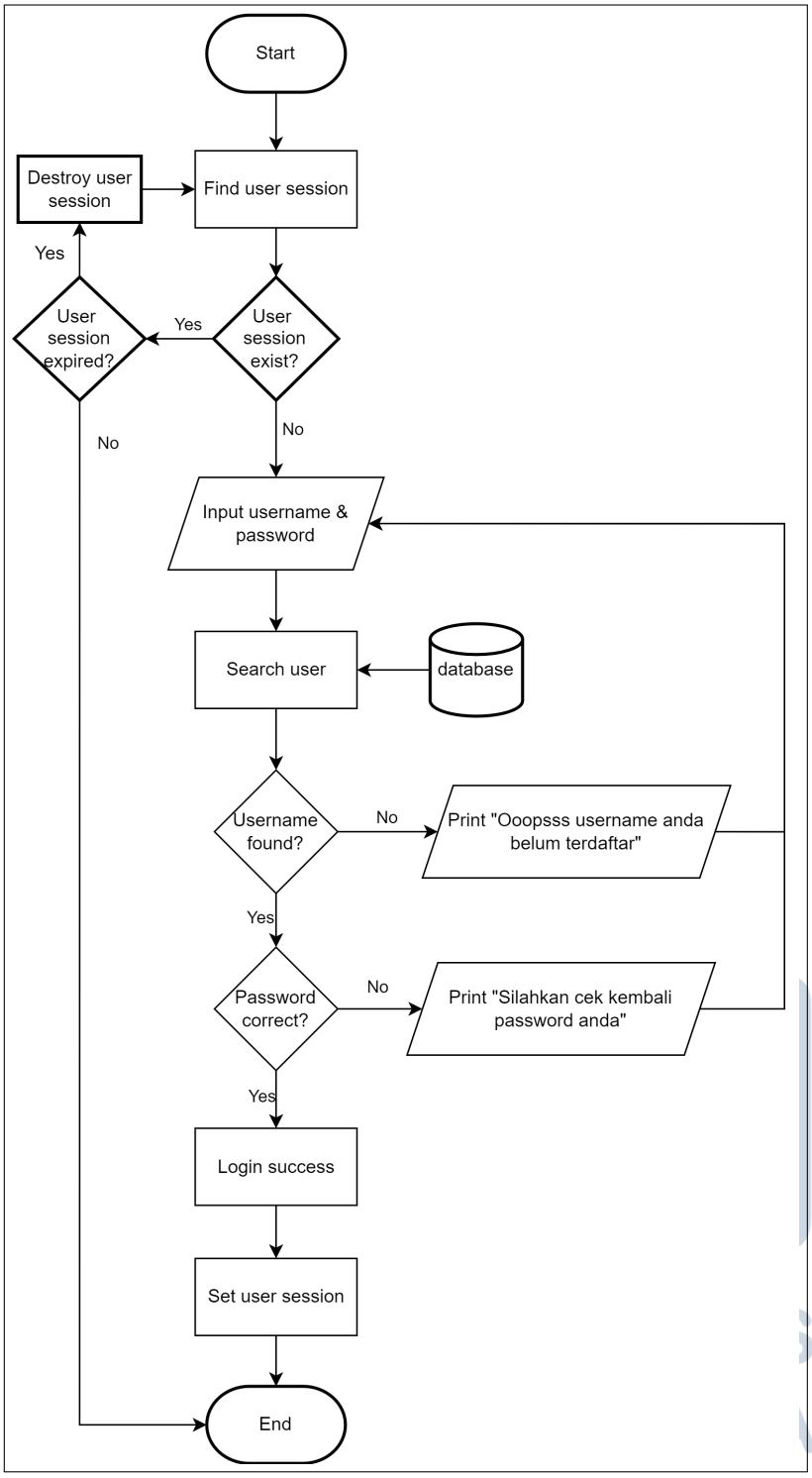


Gambar 3.1. Flowchart sistem

2. *Flowchart Login*

Gambar 3.2 merupakan *flowchart* dari fungsi *login*. Pada awalnya, sistem akan mencari sesi pengguna terlebih dahulu. Jika sesi pengguna ditemukan, sistem akan melihat waktu kadaluarsa dari sesi tersebut. Jika sesi pengguna belum kadaluarsa, pengguna dapat langsung mengakses sistem. Jika sesi pengguna sudah kadaluarsa atau sudah melewati batas waktu penggunaan,

sesi pengguna akan dihancurkan dan sistem akan mencari sesi pengguna tersebut kembali. Jika sesi pengguna tidak ditemukan, sistem akan meminta pengguna untuk mengisi *username* dan *password*. *Username* dan *password* yang dimasukkan akan di cek oleh sistem. Jika *username* tidak ditemukan, sistem akan menampilkan pesan eror ”Oops, username Anda belum terdaftar”. Jika *username* ditemukan, sistem akan membandingkan *password* yang terdaftar dengan *password* yang baru dimasukkan. Jika *password* tidak sama, sistem akan menampilkan pesan eror ”Silakan cek kembali password Anda”. Jika *password* sama, maka *password* yang dimasukkan pengguna untuk *username* tersebut benar dan pengguna berhasil melakukan *login*. Setelah pengguna berhasil *login*, sistem akan membuat sesi pengguna baru.

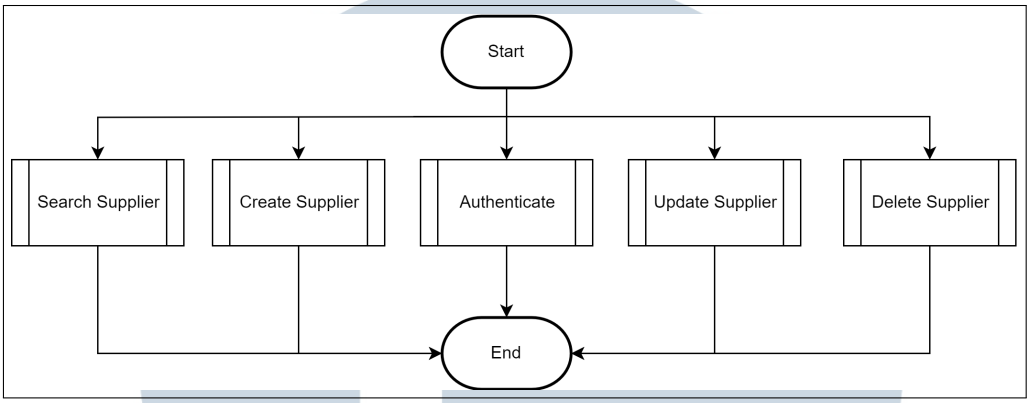


Gambar 3.2. Flowchart login

3. *Flowchart View Supplier Master Data*

Gambar 3.3 merupakan *flowchart* dari halaman *supplier master data*. Halaman *supplier master data* memperbolehkan pengguna yang memiliki

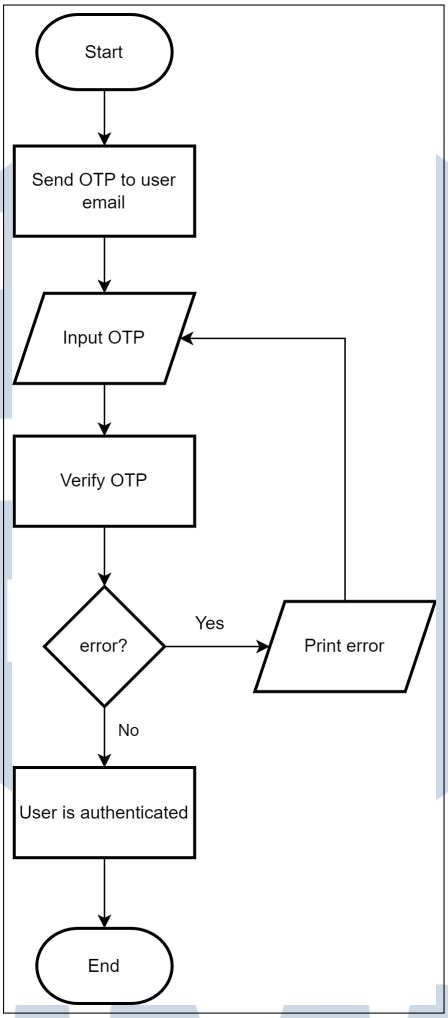
akses halaman tersebut untuk melakukan CRUD (*Create, Read, Update, Delete* pada data *supplier*.



Gambar 3.3. *Flowchart* menu *supplier master data*

4. *Flowchart Authenticate*

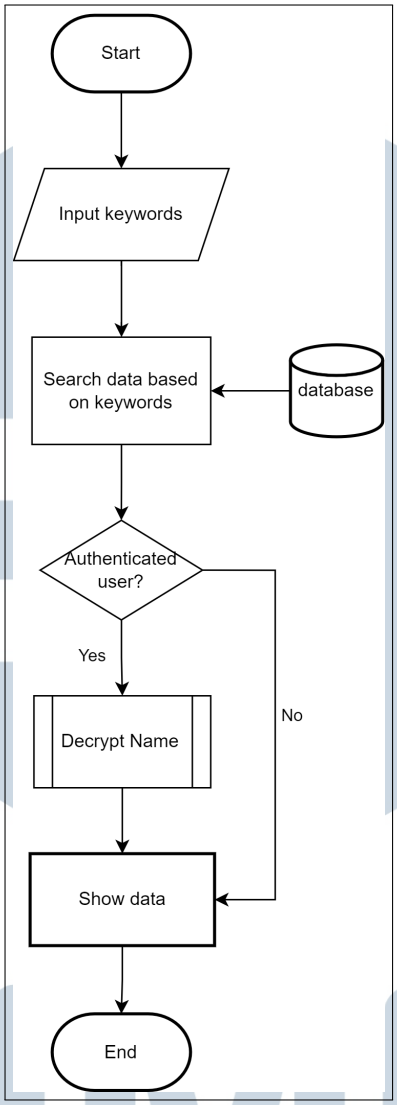
Gambar 3.4 merupakan *flowchart* dari proses *Authenticate*. Pertama, sistem akan mengirimkan *One-Time Password* (OTP) ke *email* pengguna. Setelah pengguna mendapatkan *email* tersebut, pengguna diminta untuk memasukkan OTP tersebut. Hasil *input* OTP dari pengguna akan dicek oleh sistem. Jika salah atau eror, sistem akan menampilkan pesan eror dan meminta pengguna untuk memasukkan OTP kembali. Jika benar, sistem akan menandakan bahwa pengguna dari akun tersebut sudah terautentikasi.



Gambar 3.4. *Flowchart Authenticate*

5. *Flowchart Search Supplier*

Gambar 3.5 merupakan *flowchart* dari proses *Search Supplier*. Pada proses ini, pengguna dapat mencari dan melihat data *supplier* dengan memasukkan kata kunci berdasarkan apa yang ingin dicari, lalu sistem akan mencari data tersebut di *database* berdasarkan kata kunci yang dimasukkan pengguna. Setelah itu, sistem akan menampilkan data-data tersebut. Namun, jika user belum melakukan autentikasi, nama dari *supplier* akan tetap terenkripsi. Jika sudah, nama dari *supplier* akan didekripsi terlebih dahulu, baru ditampilkan.

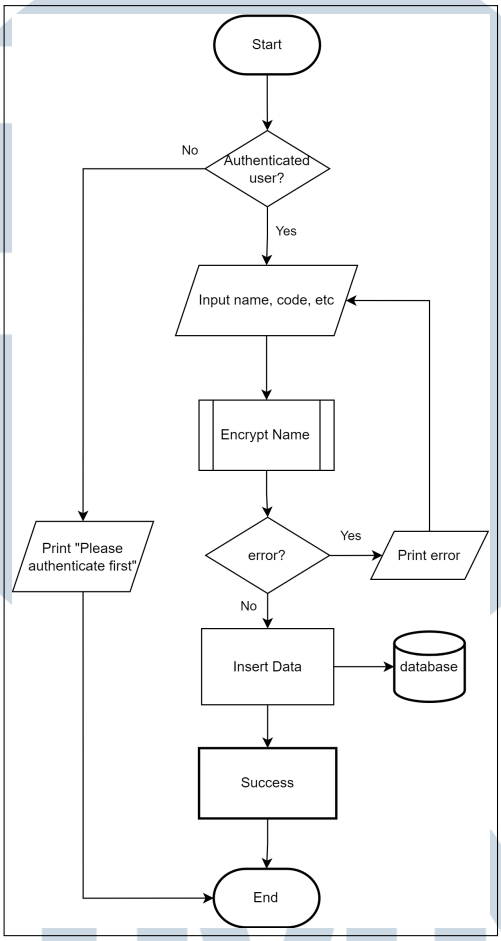


Gambar 3.5. *Flowchart Search Supplier*

6. *Flowchart Create Supplier*

Gambar 3.6 merupakan *flowchart* dari proses *Create Supplier*. Sebelum pengguna dapat memasukkan data *supplier* baru, sistem akan melakukan pengecekan pada autentikasi pengguna. Jika pengguna belum terauntetikasi, akan muncul pesan ”Please authenticate first” dan proses ini akan berhenti. Jika sudah, pengguna baru dapat memasukkan data *supplier* baru dengan cara memasukkan data-data yang dibutuhkan untuk melengkapi data *supplier* seperti nama, *code*, dan lain-lain. Sistem akan melakukan enkripsi pada nama *supplier* terlebih dahulu untuk memastikan keamanan identitas *supplier*. Jika proses enkripsi gagal, akan ditampilkan pesan eror dan pengguna harus membetulkan *field* yang memicu eror tersebut. Jika proses enkripsi

berhasil, sistem akan memasukkan data-data tersebut ke dalam *database*. Apabila proses pembuatan *supplier* berhasil, sistem akan menampilkan pesan berhasil.

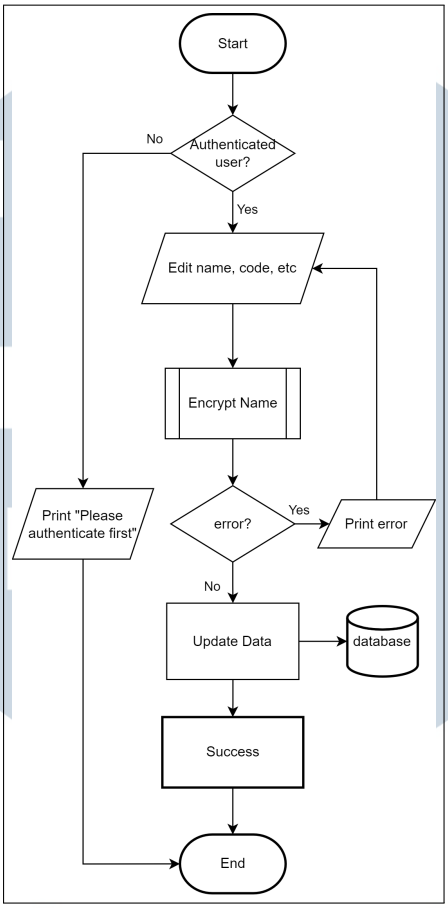


Gambar 3.6. *Flowchart Create Supplier*

7. *Flowchart Update Supplier*

Gambar 3.7 merupakan *flowchart* dari proses *Update Supplier*. Sebelum pengguna dapat memperbaharui data *supplier*, sistem akan melakukan pengecekan pada autentikasi pengguna. Jika pengguna belum terautentikasi, akan muncul pesan *"Please authenticate first"* dan proses ini akan berhenti. Jika sudah, pengguna baru dapat mengganti data yang ingin diganti. Sistem akan melakukan enkripsi pada nama *supplier* terlebih dahulu. Jika proses enkripsi gagal, akan ditampilkan pesan eror dan pengguna harus membetulkan *field* yang memicu eror tersebut. Jika proses enkripsi berhasil, sistem akan memperbaharui data *supplier* yang berada di *database*. Apabila proses pembaharuan data *supplier* berhasil, sistem akan menampilkan pesan

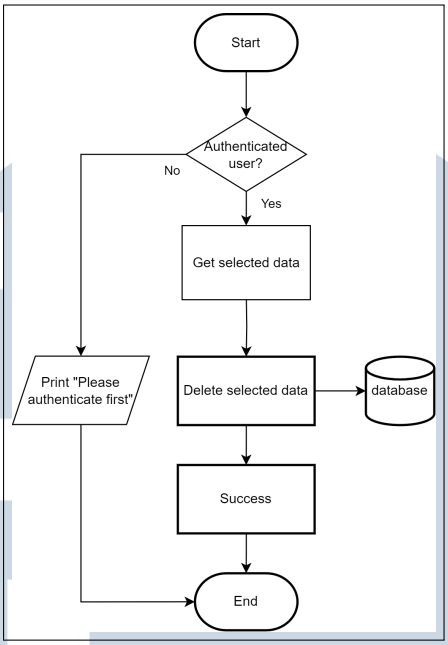
berhasil.



Gambar 3.7. *Flowchart Update Supplier*

8. *Flowchart Delete Supplier*

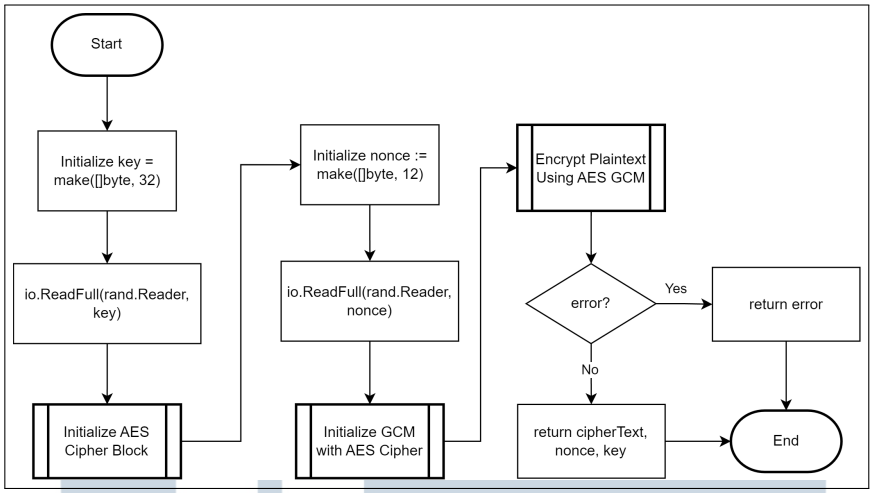
Gambar 3.8 merupakan *flowchart* dari proses *Delete Supplier*. Sebelum pengguna dapat menghapus data *supplier*, sistem akan melakukan pengecekan terhadap autentikasi pengguna. Jika pengguna belum terautentikasi, akan muncul pesan *"Please authenticate first"* dan proses ini akan berhenti. Jika sudah, pengguna baru dapat menghapus data *supplier* dengan memilih data *supplier* yang ingin dihapus. Sistem akan menerima data tersebut dan menghapus data *supplier* yang ada di *database* berdasarkan data yang dipilih oleh pengguna. Apabila proses penghapusan data *supplier* berhasil, sistem akan menampilkan pesan berhasil.



Gambar 3.8. *Flowchart Delete Supplier*

9. *Flowchart Encrypt Name*

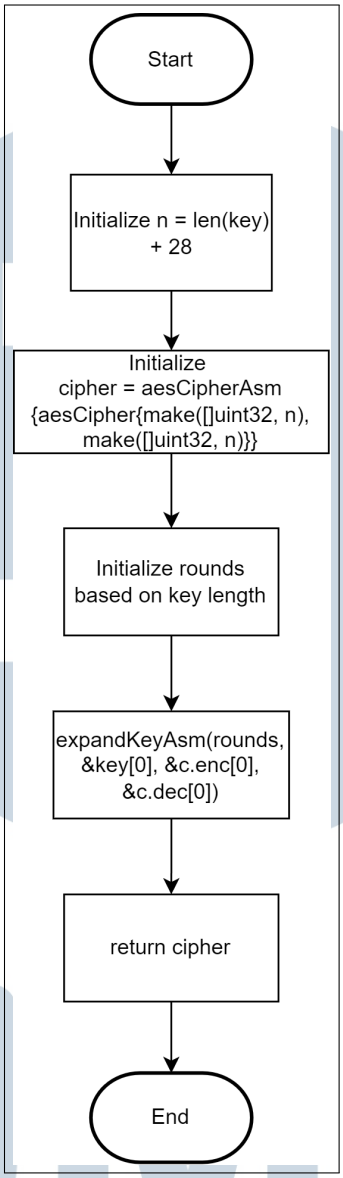
Gambar 3.9 merupakan *flowchart* dari proses *Encrypt Name*. Sistem akan membuat *key* dengan panjang 32 *byte* dan memasukkan *byte* yang bersifat acak dan unik ke dalam variabel *key*. Kemudian, sistem akan membuat *cipher block* AES yang berisikan seluruh *round key* yang akan digunakan selama proses enkripsi. Selanjutnya adalah inisialisasi *nonce* dengan panjang 12 *byte* dan akan diisi dengan *byte* acak. Diikuti dengan pembuatan *cipher block* baru yang beroperasi dalam GCM berdasarkan *cipher block* AES yang sudah dibuat. Terakhir, sistem akan melakukan enkripsi terautentikasi pada *plaintext* dengan menggunakan *cipher block* GCM. Proses enkripsi terautentikasi tersebut akan menghasilkan *ciphertext*. Jika proses enkripsi terautentikasi berhasil, maka nilai *ciphertext*, *nonce*, dan *key* akan dikembalikan. Jika gagal, maka nilai yang dikembalikan adalah *string* eror.



Gambar 3.9. *Flowchart Encrypt Name*

10. *Flowchart Initialize AES Cipher Block*

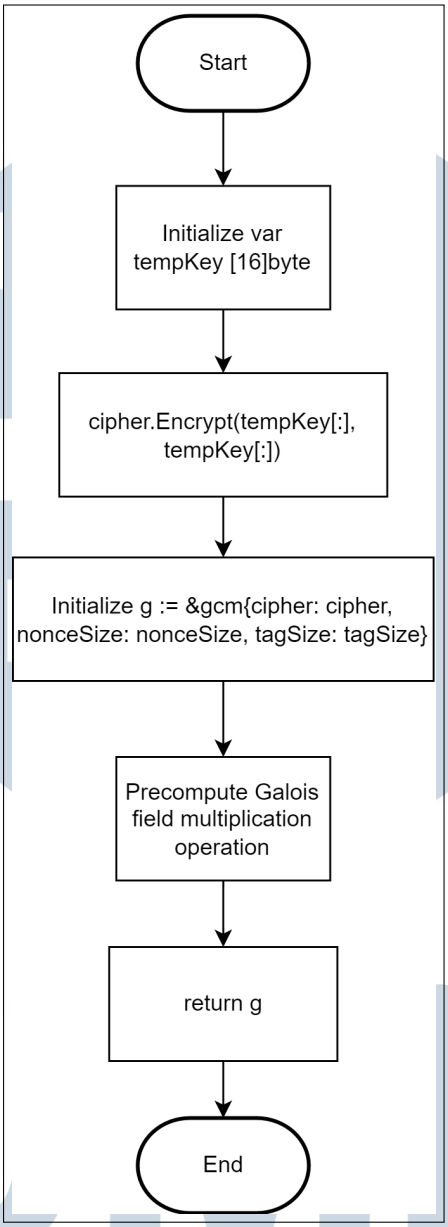
    Gambar 3.10 merupakan *flowchart* dari proses *Initialize AES Cipher Block*. Pertama sistem akan menginisiasi variabel $n = len(key) + 28$. Angka 28 didapat dari 60 (maksimum jumlah kata yang dibutuhkan) dikurangi 32 (ukuran *key* dalam kata). Setelah nilai *n* sudah didapatkan, selanjutnya adalah membuat sebuah *instance* baru dari struktur data *aesCipher*. Struktur *aesCipher* memiliki dua *field*, yaitu *enc* dan *dec*, yang masing-masing merupakan *slice* dari uint32. *Slice* ini digunakan untuk menyimpan *round key* dengan jumlahnya yang bergantung dengan variabel *rounds*. Seluruh *roundkey* ini akan dibuat pada proses expandKeyAsm dan digunakan dalam proses enkripsi (*enc*) dan dekripsi (*dec*).



Gambar 3.10. *Flowchart Initialize AES Cipher Block*

11. *Flowchart Initialize GCM with AES Cipher*

    Gambar 3.11 merupakan *flowchart* dari proses *Initialize GCM with AES Cipher*. Sistem akan menginisialisasi variabel *tempKey* sepanjang 16 *byte*. Variabel tersebut akan di enkripsi dengan menggunakan *cipher block* AES dan hasil enkripsi nya akan dimasukkan lagi ke dalam variabel *tempKey*. Variabel *tempKey* akan digunakan pada proses *precompute* operasi perkalian *galois field*. Setelah itu, sistem juga akan menginisialisasi variabel *g* yang berisi *instance* baru dari struktur GCM. Kemudian, *field cipher*, *nonceSize*, *tagSize*, dan operasi perkalian *galois field* diinisialisasi ke dalam variable *g*.



Gambar 3.11. *Flowchart Initialize GCM with AES Cipher*

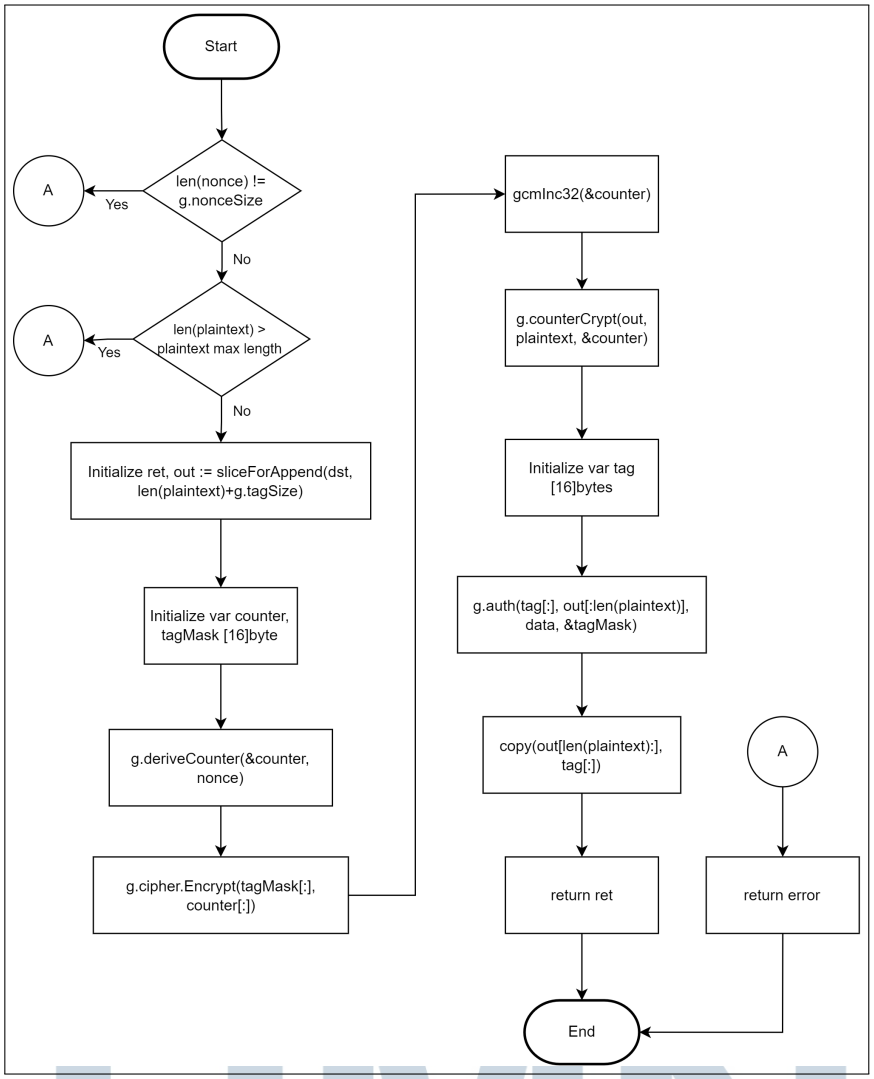
12. *Flowchart Encrypt Plaintext using AES GCM*

Gambar 3.12 merupakan *flowchart* dari proses *Encrypt Plaintext using AES GCM*. Sebelum proses ini mulai, sistem akan membandingkan ukuran *nonce* yang sudah dibuat dengan ukuran *nonce* yang diharapkan, yaitu 12 *byte*. Jika ukurannya tidak sama, sistem akan mengembalikan *string* eror dan proses ini tidak dilanjutkan kembali. Jika sama, sistem akan melakukan pengecekan kedua, yaitu membandingkan panjang *plaintext* dengan panjang maksimum *plaintext*. Jika ukuran *plaintext* melebihi batas maksimum, maka

fungsi *Encrypt Name* akan mengembalikan *string* eror dan proses ini tidak dilanjutkan kembali. Jika tidak, sistem akan menginisialisasi variabel *ret* dan *out* yang keduanya berbagi *array* yang sama, akan tetapi ukuran *out* sudah diperbesar untuk menampung *plaintext* dan *authentication tag*.

Hal pertama yang akan dilakukan oleh sistem dalam proses enkripsi dengan menggunakan AES-GCM adalah menginisialisasi *counter* dan menghitung nilai awalnya berdasarkan *nonce*. Kemudian, *counter* awal akan dienkripsi dengan menggunakan *cipher block* AES. Hasil enkripsi disimpan dalam variabel *tagMask* yang akan digunakan untuk menghitung *authentication tag*/memverifikasi integritas data. Selanjutnya adalah melakukan *increment* pada *counter*. *Counter* ini akan mengalami *increment* setelah setiap blok data dienkripsi, sehingga setiap blok data dienkripsi dengan nilai *counter* yang berbeda. Terakhir, *plaintext* akan dienkripsi menggunakan *counter* dari *cipher block* AES dan menulis *ciphertext* ke variabel *out*.

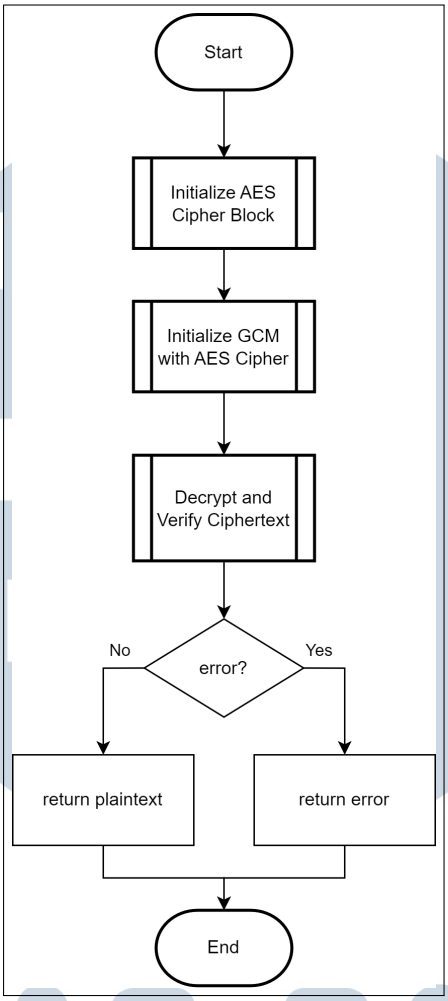
Sistem dapat melakukan autentikasi enkripsi tersebut dengan menghitung *authentication tag* untuk *ciphertext* dan data tambahan jika ada. Hasilnya disimpan dalam *tag*. Variabel *tag* akan disalin ke dalam variabel *out* setelah *ciphertext*.



Gambar 3.12. *Flowchart Encrypt Plaintext using AES GCM*

13. *Flowchart Decrypt Name*

Gambar 3.13 merupakan *flowchart* dari proses *Decrypt Name*. Pertama, sistem akan membuat *cipher block* AES yang berisikan seluruh *round key* yang akan digunakan selama proses dekripsi. Selanjutnya adalah pembuatan *cipher block* baru yang beroperasi dalam GCM berdasarkan *cipher block* AES yang sudah dibuat. Terakhir, sistem akan melakukan dekripsi terautentikasi pada *ciphertext* dengan menggunakan cipher block GCM. Jika proses dekripsi terautentikasi tersebut berhasil, maka nilai *plaintext* akan dikembalikan. Jika gagal, maka nilai yang dikembalikan adalah *string* eror.



Gambar 3.13. *Flowchart Decrypt Name*

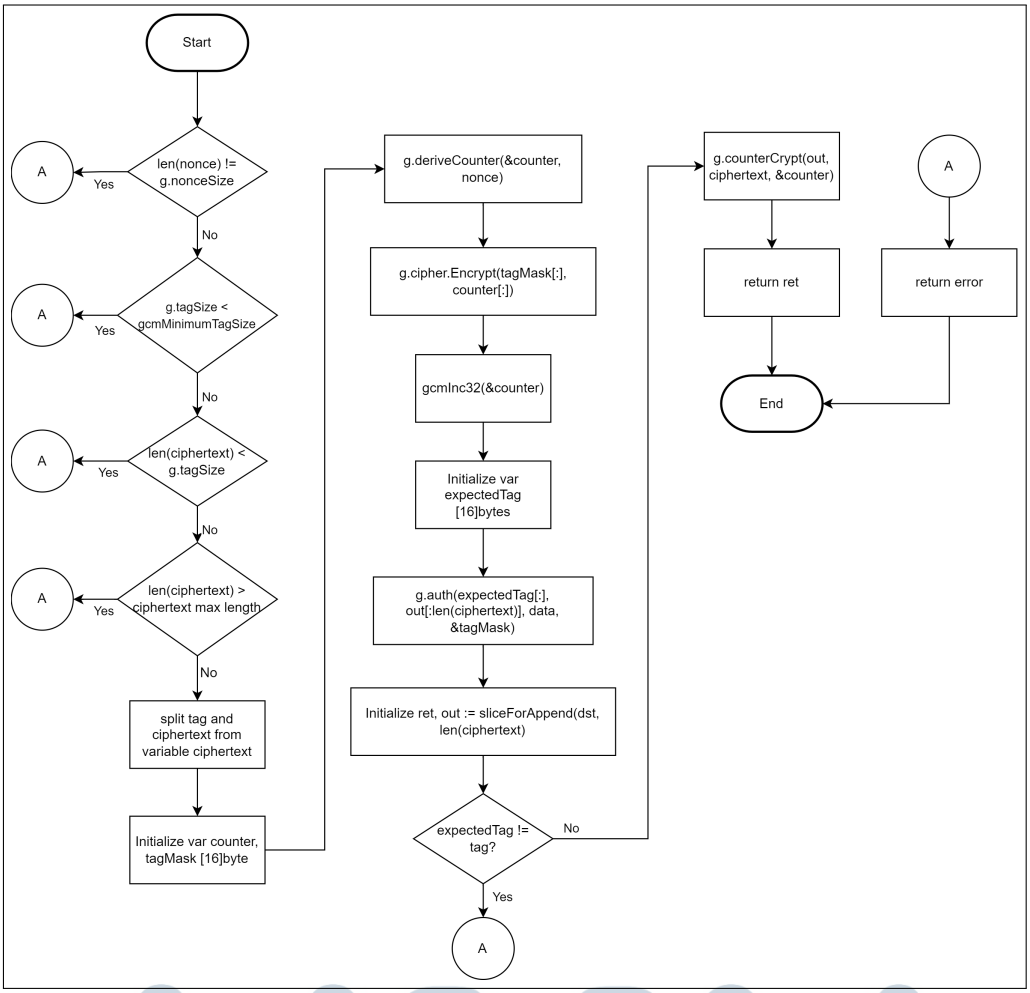
14. *Flowchart Decrypt and Verify Ciphertext*

Gambar 3.14 merupakan *flowchart* dari proses *Decrypt and Verify Ciphertext*. Sebelum proses ini mulai, sistem akan membandingkan ukuran *nonce* yang digunakan saat melakukan enkripsi dengan ukuran *nonce* yang diharapkan, yaitu 12 byte. Jika ukurannya tidak sama, sistem akan mengembalikan *string* eror dan proses ini tidak dilanjutkan kembali. Jika sama, sistem selanjutnya akan membandingkan ukuran *tag* yang sudah diinisiasi dengan ukuran minimum *tag* yang diharapkan, yaitu 12 *byte*. Jika ukuran *tag* yang sudah diinisiasi lebih kecil dari ukuran minimum *tag* yang diharapkan, sistem akan mengembalikan *string* eror dan proses ini tidak dilanjutkan kembali. Jika lebih besar, sistem akan melanjutkan pengecekan ketiga, yaitu membandingkan ukuran *ciphertext* dengan ukuran *tag* yang sudah diinisiasi. Jika ukuran *ciphertext* lebih kecil dari ukuran *tag* yang sudah diinisiasi,

sistem akan mengembalikan *string* eror dan proses ini tidak dilanjutkan kembali. Jika lebih besar, sistem akan melakukan pengecekan terakhir, yaitu membandingkan panjang *ciphertext* dengan panjang maksimum *plaintext*. Jika ukuran *ciphertext* melebihi batas maksimum, maka sistem akan mengembalikan *string* eror dan proses ini tidak dilanjutkan kembali. Jika tidak, sistem akan memisahkan *ciphertext* dan *tag* yang ada pada variabel *ciphertext*.

Hal pertama yang akan dilakukan oleh sistem dalam proses dekripsi dengan menggunakan AES-GCM adalah menginisialisasi *counter* dan menghitung nilai awalnya berdasarkan *nonce*. Kemudian, *counter* awal akan dienkripsi dengan menggunakan *cipher block* AES. Hasil enkripsi disimpan dalam variabel *tagMask* yang akan digunakan untuk menghitung *expected authentication tag*. Selanjutnya adalah melakukan *increment* pada *counter*. Kemudian, sistem akan menginisialisasi variabel *expectedTag* dan akan menghitung *authentication tag*. Hasilnya akan disimpan dalam variabel expectedTag. Dilanjutkan dengan inisialisasi variabel *ret* dan *out* yang keduanya berbagi *array* yang sama, akan tetapi ukuran *out* sudah diperbesar untuk menampung $len(ciphertext)$ *byte*.

Sebelum sistem melakukan dekripsi pada *ciphertext*, sistem harus melakukan verifikasi pada *authentication tag* dengan membandingkan *expectedTag* dan *tag* yang berada di dalam *ciphertext*. Jika tidak sama, sistem akan mengembalikan *string* eror dan proses ini tidak dilanjutkan kembali. Jika sama, sistem akan mulai melakukan dekripsi pada *ciphertext* dengan menggunakan *counter* dari *cipher block AES* dan menulis *plaintext* ke variabel *out*. Terakhir, sistem akan mengembalikan variabel *ret*.



Gambar 3.14. *Flowchart Decrypt Ciphertext using AES GCM*

3.3.2 Spesifikasi Sistem

Spesifikasi dari perangkat yang digunakan selama proses penelitian ini adalah sebagai berikut.

- Perangkat Lunak
  - Windows 11 Pro 64-bit
  - Google Chrome
  - Visual Studio Code
  - Go 1.21.5
  - DBeaver
- Perangkat Keras

  - *Processor*: AMD Ryzen 5 5600G 6-Core Processor 3,9 GHz
  - RAM: 16GB DDR4 3200 MHz
  - *Power Supply*: Be quiet SP9 500 watt
  - *Motherboard*: Gigabyte B550M Gaming

- *Virtual Private Server* (VPS)
  - *Processor*: Intel Core Processor (Broadwell, IBRS) 2-Core Processor
  - RAM: 2GB
  - *Disk Size*: 20GB

3.4 Evaluasi

Sistem pengadaan berbasis *web* yang sudah menerapkan AES-GCM untuk keamanan data akan dievaluasi. Terdapat 2 jenis evaluasi yang akan dilakukan. Pertama, evaluasi tingkat risiko keseluruhan. Evaluasi tingkat risiko keseluruhan akan dilakukan bersama pakar dengan menggunakan *automated tools*. *Tools* tersebut akan menilai sebuah kerentanan dalam aplikasi dengan menggunakan metodologi OWASP *Risk Rating* berdasarkan kategori kerentanan yang ada pada OWASP *Top Ten*.

Evaluasi kedua berfokus pada kecepatan sistem dalam mengenkripsi dan mendekripsi data. Evaluasi ini dilakukan dengan mengukur durasi perubahan *plaintext* menjadi *ciphertext* dan sebaliknya.

3.5 Dokumentasi

Tahap akhir dari penelitian ini adalah penyusunan laporan yang merangkum seluruh proses dan hasil penelitian. Isi dari laporan ini merupakan dokumentasi penelitian yang lengkap, mulai dari latar belakang, rumusan masalah, metodologi penelitian, hasil, sampai kesimpulan dan saran.