Enkripsi Berkas untuk *Multiparticipant Party* memanfaatkan Skema Enkripsi ECIES

Bayu Samudra - 13520128
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520128@std.stei.itb.ac.id

***Abstract*—Keamanan informasi merupakan hal yang sangat penting pada masa kini. Salah satu aspek keamanan yang perlu dijaga adalah terkait dengan *confidentiality*. Kontrol yang dapat dilakukan untuk melakukan hal tersebut adalah dengan melakukan proses enkripsi. Saat ini, terdapat pendekatan yang dilakukan oleh ZIP archive unruk melakukan proses enkripsi banyak partisipan. Kelemahan dari sistem ini adalah tidak adanya perlindungan intergitas dari pesan. Pada makalah ini, dilakukan pembahasan terkait dengan penerapan skema enkripsi ECIES pada pengiriman berkas untuk *multiparticipant party*. Pada makalah ini, juga dibahas terkait dengan pemanfaatan *authentication data* pada mode blok GCM dan penerapannya pada algoritma ECIES. Makalah ini juga membahas terkait dengan hak akses pada berkas terenkripsi, serta mekanisme penjagaan integritas pada berkas terenkripsi melalui tanda tangan digital. Pengujian menunjukan bahwa metode ini dapat digunakan untuk melakukan enkripsi berkas untuk *multiparticipant party*.**

Index Terms*—ECIES, GCM, AEAD, multiparticipant party, *digital signature

I. Pendahuluan

Keamanan informasi pada masa kini merupakan hal yang sangat penting. Dengan perkembangan teknologi digital saat ini, ancaman keamanan pada saat ini semakin meningkat. Salah satu bentuk ancaman keamanan yang sering terjadi adalah terkait dengan permasalahan *confidentiality*. *Confidentiality* merupakan salah satu aspek penting dalam keamanan informasi yang berarti bahwa informasi hanya dapat diakses oleh pihak yang berhak. Salah satu cara untuk menjaga *confidentiality* adalah dengan menggunakan teknik enkripsi. Enkripsi merupakan proses mengubah informasi agar tidak dapat dibaca oleh pihak yang tidak berhak.

Penerapan enkripsi pada berkas umumnya dilakukan dengan menggunakan teknik enkripsi kunci simetris. Kelemahan dari metode enkripsi kunci simetris adalah distribusi kunci yang sulit. Hal ini dikarenakan penyerang dapat saja menyadap komunikasi saat kedua belah pihak melakukan pertukaran kunci. Salah satu cara untuk mengatasi masalah ini adalah dengan menggunakan teknik enkripsi kunci asimetris. Algoritma kunci simetris menggunakan dua buah kunci yaitu kunci publik dan kunci privat. Kunci publik digunakan untuk mengenkripsi pesan, sedangkan kunci privat digunakan untuk mendekripsi pesan. Salah satu contoh algoritma yang termasuk dalam algoritma ini adalah RSA.

Kelemahan dari penggunaan algoritma kunci asimetris adalah *resource* yang dibutuhkan untuk melakukan operasi kriptografi ini cukup besar. Hal ini berdampak pada kinerja dari algoritma ini. Oleh karena itu, salah satu cara untuk mengatasi masalah ini adalah dengan melakukan *hybrid cryptography*. *Hybrid cryptography* merupakan kombinasi dari algoritma kunci simetris dan kunci asimetris. Pada *hybrid cryptography*, pesan akan dienkripsi menggunakan algoritma kunci simetris, kemudian kunci simetris tersebut akan dienkripsi menggunakan algoritma kunci asimetris.

Penerapan *hybrid cryptography* dalam pengiriman berkas pada saat ini masih terbatas pada penggunaan untuk satu peserta saja. Salah satu contoh penerapan hal ini adalah pada protokol TLS. Pada protokol TLS, kedua belah pihak akan membentuk terlebih dahulu kunci simetris dengan memanfaatkan algoritma pertukaran kunci Diffie-Hellman. Setelah itu, kunci simetris tersebut akan digunakan untuk mengenkripsi pesan. Kelemahan dari penggunaan teknik ini adalah hanya dua peserta saja yang dapat mengetahui isi pesan.

Penerapan lain dari *hybrid cryptography* adalah teknik yang dilakukan pada aplikasi kompresi menggunakan ZIP. Pada aplikasi kompresi ZIP, pengguna dapat melakukan enkripsi berkas menggunakan kunci simetris. Kunci simetris tersebut kemudian dienkripsi menggunakan kunci asimetris. Hasil enkripsi tersebut kemudian disimpan dalam berkas ZIP. Kelemahan dari metode ini adalah tidak adanya mekanisme penjagaan intergritas pada berkas. Selain itu, pada metode ini tidak ada mekanisme untuk mengatur hak akses pada berkas terenkripsi tersebut.

Pada makalah ini, akan dibahas mengenai penerapan *hybrid cryptography* pada pengiriman berkas untuk *multiparticipant party*. Pada *multiparticipant party*, terdapat lebih dari dua peserta yang terlibat dalam pengiriman berkas. Pada makalah ini, akan dibahas mengenai penerapan skema enkripsi ECIES pada enkripsi berkas untuk *multiparticipant party*. Makalah ini juga akan membahas terkait pemanfaatan *authentication data* pada mode blok GCM untuk parameter ECIES. Makalah ini akan membahas mengenai terkait dengan hak akses pada berkas terenkripsi, serta mekanisme penjagaan integritas pada berkas terenkripsi melalui tanda tangan digital.

II. Landasan Teori

A. Elliptic Curve Cryptography

Menurut [1], Kriptografi kurva eliptik merupakan salah satu metode kriptografi yang menggunakan operasi pada kurva

eliptik. Kurva eliptik merupakan kurva pada dimensi dua yang memenuhi persamaan 1. Syarat dari persamaan ini agar disebut sebagai kurva eliptik adalah $4a^3 + 27b^2 \neq 0$.

$$y^2 = x^3 + ax + b \tag{1}$$

Menurut [2], Kekuatan Kriptografi dari ECC terletak pada permasalahan *Elliptic Curve Discrete Logarithm Problem* (ECDLP). ECDLP merupakan permasalahan untuk mencari nilai k pada persamaan $kP = Q$. Pada persamaan ini, P merupakan titik pada kurva eliptik, sedangkan Q merupakan hasil perkalian titik P dengan skalar k. Bila dibandingkan dengan permasalahan *Discrete Logarithm Problem* (DLP) pada kriptografi RSA, ECDLP memiliki kualitas yang lebih baik dalam hal efisiensi. Hal ini dikarenakan kunci yang digunakan lebih pendek dibandingkan dengan kunci yang digunakan pada RSA.

B. Eliptic Curve Integrated Encryption Scheme (ECIES)

Elliptic Curve Integrated Encryption Scheme (ECIES) merupakan salah satu skema enkripsi hibrida. Skema ini diusulkan oleh Abdala, Bellare, dan Rogaway pada tahun 2021. Menurut [3], ECIES merupakan pengembangan dari metode pertukaran kunci berbasis Diffie-Hellman. Pada ECIES, pesan akan dienkripsi menggunakan algoritma kunci simetris. Kunci simetris tersebut kemudian dienkripsi menggunakan algoritma kunci asimetris. Pada ECIES, kunci simetris yang digunakan akan dihasilkan dari pertukaran kunci Diffie-Hellman yang dilakukan proses pembangkitan kunci (KDF).

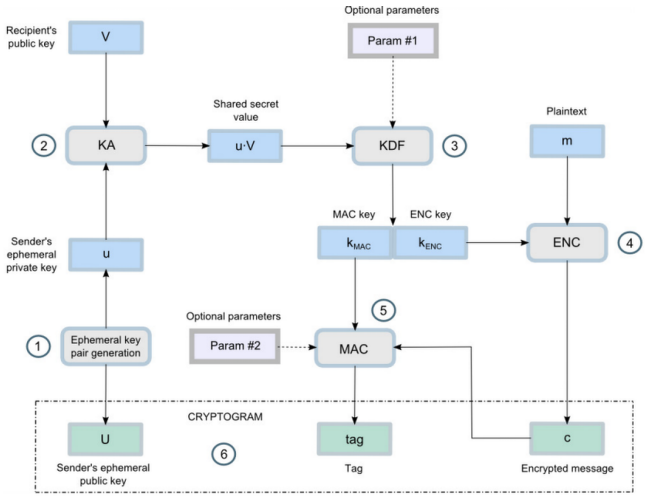

Fig. 1. Ilustrasi proses enkripsi menggunakan ECIES (Sumber: [6])

Pada skema ini, hal yang dilakukan adalah pembangkitan kunci. Ilustrasi dari proses enrkipsi memanfaatkan sistem ini ditunjukan pada Figur 1. Untuk menjelaskan tata cara enkripsi, asumsikan terdapat dua buah pihak yang melakukan komunikasi, yaitu Alice dan Bob. Alice akan mengirimkan pesan m kepada Bob. Proses pembangkitan kunci dilakukan dengan cara sebagai berikut [2]:

1) Alice dan Bob akan menyepakati parameter kurva eliptik yang digunakan. Parameter yang diperlukan adalah a, b, p, dan G. Parameter G adalah titik pada kurva eliptik yang digunakan sebagai titik pembangkit (*generator point*).
2) Bob akan memilih skalar d_B sebagai kunci privatnya.
3) Bob akan menghitung titik $Q_B = d_B \cdot G$ sebagai kunci publiknya.
4) Nilai Q_B akan dikirimkan kepada Alice.

Saat Alice menerima nilai Q_B, Alice akan melakukan proses enkripsi sebagai berikut:

1) Alice memilih skalar d_A sebagai nilai sementara untuk enkripsi.
2) Alice menghitung titik $Q_m = d_A \cdot G$ sebagai kunci publik sementara.
3) Alice menghitung *shared key* $K = Q_A + Q_m$.
4) Alice menghitung kunci enkripsi dan MAC. Hasil dari fungsi KDF ini merupakan konkatenasi dari kunci enkripsi k_1 dan kunci MAC k_2.
5) Alice akan mengenkripsi pesan m menggunakan kunci enkripsi k_1. Hasil enkripsi ini disebut dengan $c = E(m, k_1)$.
6) Hasil enkripsi c akan dihitung nilai MAC-nya menggunakan kunci MAC k_2. Hasil dari MAC ini disebut dengan $t = MAC(c, k_2)$.
7) Alice akan mengirimkan pesan $C = Q_m \parallel c \parallel t$ kepada Bob.

Saat Bob menerima pesan C, Bob akan melakukan proses dekripsi sebagai berikut:

1) Bob akan memecah terlebih dahulu pesan C menjadi Q_m, c, dan t.
2) Bob akan menghitung *shared key* $K = Q_m \cdot d_B$.
3) Bob akan menghitung kunci enkripsi dan MAC. Hasil dari fungsi KDF ini merupakan konkatenasi dari kunci enkripsi k_1 dan kunci MAC k_2.
4) Bob akan menghitung nilai MAC dari pesan c menggunakan kunci MAC k_2. Hasil dari MAC ini disebut dengan $t' = MAC(c, k_2)$.
5) Bob akan melakukan verifikasi MAC. Bila $t = t'$, maka pesan c dianggap valid.
6) Bila pesan c valid, Bob akan mendekripsi pesan c menggunakan kunci enkripsi k_1. Hasil dekripsi ini disebut dengan $m = D(c, k_1)$.

Dengan metode seperti itu, pesan m yang dikirimkan oleh Alice dapat dibaca oleh Bob. Selain itu, keunggulan dari metode ini adalah integritas pesan yang dikirimkan dapat diverifikasi. Keunggulan metode ini juga adalah enkripsi dilakukan menggunakan kunci simetris, sehingga proses enkripsi dan dekripsi dapat dilakukan lebih cepat.

C. Tanda Tangan Digital Elliptic Curve Digital Signature Algorithm *(ECDSA)*

Menurut [1], Algoritma ECDSA merupakan algoritma tanda tangan digital yang dikembangkan dari algoritma tanda tangan ElGammal. Algoritma ini dikembangkan oleh Abdala, Bellare,

dan Rogaway pada tahun 1999. Asumsikan terdapat dua buah partisipan, yakni Alice dan Bob yang akan melakukan komunikasi. Alice akan mengirimkan pesan m kepada Bob. Alice ingin membuat penjagaan integritas pada pesan m memanfaatkan ECDSA. Proses pembangkitan kunci dilakukan dengan cara sebagai berikut [2]:

1) Alice dan Bob akan menyepakati parameter kurva eliptik yang digunakan. Parameter yang diperlukan adalah a, b, p, dan G. Parameter G adalah titik pada kurva eliptik yang digunakan sebagai titik pembangkit (*generator point*).
2) Alice akan memilih skalar d_A sebagai kunci privatnya.
3) Alice akan menghitung titik $Q_A = d_A \cdot G$ sebagai kunci publiknya.
4) Nilai Q_A akan dikirimkan kepada Bob.
5) Bob akan memilih skalar d_B sebagai kunci privatnya.
6) Bob akan menghitung titik $Q_B = d_B \cdot G$ sebagai kunci publiknya.
7) Nilai Q_B akan dikirimkan kepada Alice.

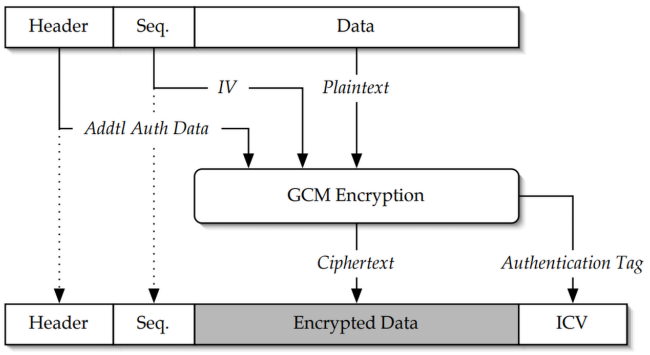

Fig. 2. Contoh penerapan AEAD pada mode blok GCM. Referensi [5]

Setelah melakukan pembangkitan kunci, Alice dapat membuat tanda tangan digital pada pesan m. Prosedur pembangkitan tanda tangan digital pada pesan m adalah sebagai berikut:

1) Alice akan menghitung nilai hash dari pesan m. Nilai hash ini disebut dengan $h = H(m)$.
2) Alice akan memilih skalar k sebagai nilai sementara untuk pembangkitan tanda tangan. Nilai ini harus berada pada rentang $1 \leq k < p$.
3) Alice akan menghitung titik $R = (R_x, R_y) = k \cdot G$.
4) Alice akan menghitung nilai $r = x_R \mod p$. Bila $r = 0$, maka Alice akan memilih nilai k yang lain.
5) Alice akan menghitung nilai $s = k^{-1} \cdot (h + d_A \cdot r) \mod p$. Bila $s = 0$, maka Alice akan memilih nilai k yang lain.
6) Tanda tangan digital yang dihasilkan adalah pasangan (r, s).

Bob akan melakukan verifikasi tanda tangan digital yang diterima dari Alice. Proses verifikasi tanda tangan digital dilakukan sebagai berikut:

1) Bob melakukan verifikasi bahwa nilai r dan s berada pada rentang $1 \leq r < n$ dan $1 \leq s < n$.
2) Bob mengambil nilai kunci publik Alice, yakni Q_A.
3) Bob akan menghitung nilai hash dari pesan m. Nilai hash ini disebut dengan $h = H(m)$.
4) Bob akan menghitung nilai $w = s^{-1} \mod p$.
5) Bob akan menghitung nilai $u_1 = h \cdot w \mod p$ dan $u_2 = r \cdot w \mod p$.
6) Bob akan menghitung titik $U = u_1 \cdot G + u_2 \cdot Q_A$.
7) Bila $U = O$, maka tanda tangan digital yang diterima tidak valid. Bila benar, tanda tangan digital yang diterima valid.

Dengan metode seperti itu, Bob dapat memastikan bahwa pesan yang diterima dari Alice tidak diubah oleh pihak lain. Hal ini membuktikan *integrity* dari data tersebut. Selain itu, Bob juga dapat memastikan bahwa pesan tersebut benar-benar berasal dari Alice. Hal ini membuktikan *authenticity* dari pengirim data tersebut.

D. Authenticated Encryption with Associated Data *(AEAD)*

Menurut [4], AEAD merupakan pengembangan sistem otentikasi pesan terenkripsi yang dilakukan dengan cara yang lebih efisien. Pada AEAD, proses enkripsi serta otentikasi pesan dilakukan dalam satu proses saja. Metode ini merupakan metode yang berbeda bila dibandingkan dengan metode pada umumnya. Sebelum AEAD muncul, proses enkripsi pesan dilakukan dengan melakukan enkripsi pesan terlebih dahulu. Setelah itu, pesan akan diperiksa integritasnya dengan melakukan proses otentikasi. Pada AEAD, proses otentikasi dilakukan pada saat proses enkripsi berlangsung. Salah satu contoh dari enkripsi berbasis AEAD adalah AES dengan mode blok GCM.

Cipher yang menerapkan sistem enkripsi dalam kelas AEAD dapat memiliki *associate data* (AD). Secara desain, data ini tidak dienkripsi, tetapi data ini akan diikutsertakan dalam proses otentikasi yang dihitung bersamaan dengan data yang dienkripsi. Fitur yang ditawarkan pada AEAD ini dapat digunakan untuk menjaga integritas data, seperti untuk menjaga integritas dari *header* pesan. Contoh penerapan AEAD adalah menggunakan blok GCM seperti yang ditunjukan pada figur 2.

GCM merupakan mode operasi dari enkripsi simetris berbasis blok yang termasuk dalam kelas AEAD. Menurut [5], mode ini dapat diimplementasikan sangat baik pada level *hardware* sehingga menghasilkan kecepatan enkripsi yang cepat. Ilustrasi dari proses enkripsi berbasis GCM ditunjukan pada figur 3. Menurut [5], terdapat empat input yang dapat terlibat dalam operasi blok GCM. Input tersebut adalah sebagai berikut:

1) *Plaintext*: Pesan yang akan dienkripsi.
2) *Additional Authenticated Data* (AAD): Data tambahan yang akan diikutsertakan dalam proses otentikasi.
3) Kunci yang digunakan untuk melakukan enkripsi.
4) IV *Initialization Vector*: Nilai yang digunakan untuk menginisialisasi counter. Nilai ini juga digunakan untuk menghitung *authentication tag*.

Output dari mode blok ini adalah sebagai berikut:

1) *Ciphertext*: Pesan yang telah dienkripsi. Panjang dari *ciphertext* berukuran sama dengan panjang dari *plaintext*.

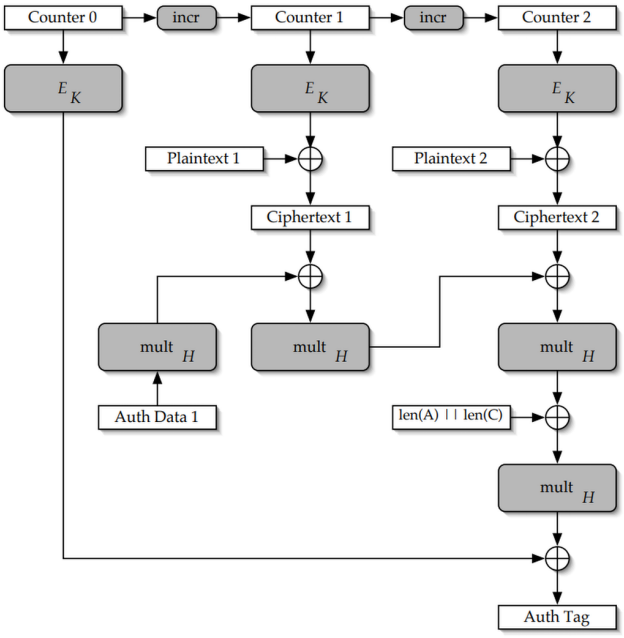

Fig. 3. Ilustrasi proses enkripsi berbasis GCM (Sumber [5])

2) *Authentication Tag*: Nilai yang digunakan untuk memverifikasi integritas pesan dan AAD.

Metode enkripsi yang dilakukan pada mode GCM pada dasarnya mengikuti proses enkripsi pada mode *counter* (CTR). Proses Enkripsi dilakukan dengan menghitung *counter* untuk blok yang akan dienkripsi. Nilai counter akan dilakukan *increment* dari nilai sebelumnya. Nilai *counter* akan dilakukan proses enkripsi. Hasil enkripsi ini akan dilakukan operasi XOR dengan blok pesan sehingga mendapatkan *ciphertext*. Hasil dari *ciphertext* ini akan diikutsertakan dalam proses otentikasi. Data yang diikutsertakan dalam proses otentikasi adalah *additional data* dan *ciphertext*. Hasil dari otentikasi ini disebut dengan *authentication tag*.

E. PBKDF2

Menurut [7], PBKDF2 merupakan sebuah fungsi random yang digunakan untuk membangkitkan kunci. Fungsi ini dapat menghasilkan derivasi kunci dengan ukuran yang tidak terbatas. Fungsi ini memerlukan beberapa parameter, seperti *password* (P), *salt* (S), *iteration* c, dan *key length* $dkLen$. Secara matematis, fungsi ini dapat dituliskan dengan persamaan 2.

$$\text{DK} = \text{PBKDF2}_{\text{PRF}}(P, S, c, dkLen) \tag{2}$$

Pada persamaan 2, DK merupakan hasil dari fungsi PBKDF2. PRF merupakan fungsi hash yang digunakan, seperti *keyed HMAC* ataupun fungsi hash. Menurut [8], PBKDF2 dapat digunakan untuk memperlambat proses *brute-force* yang dilakukan pada saat melakukan proses dekripsi.

III. Rancangan Sistem

A. Integrasi Algoritma Enkripsi ECIES dengan Mode Blok GCM

Berdasarkan bagian II-D, mode blok berbasis GCM mendukung penggunaan *additional data* (AAD) dalam proses enkripsi. Pada bagian II-B, algoritma enkripsi ECIES memerlukan sebuah *authentication tag* untuk memastikan integritas pesan yang dikirimkan. Perhitungan nilai ini pada dasarnya dapat dilibatkan dengan menambahkan *additional data* pada proses enkripsi menggunakan mode blok GCM. Ilustrasi dari usulan integrasi ini ditunjukan pada figur 4.

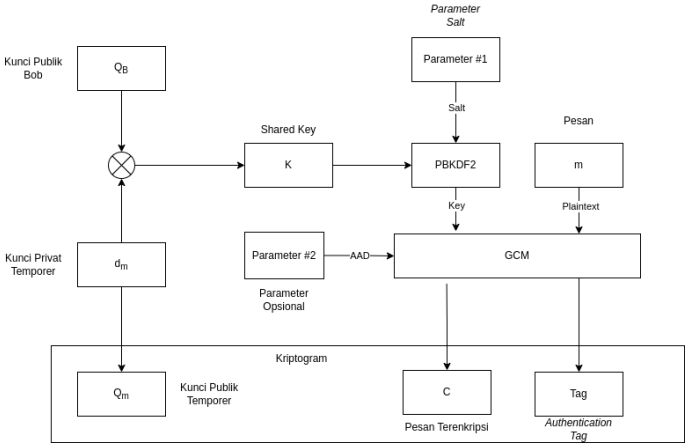

Fig. 4. Ilustrasi Integrasi ECIES dengan Mode Blok GCM

Pada ilustrasi tersebut, terdapat beberapa usulan perubahan. Usulan pertama terkait dengan penggunaan PBKDF2 untuk membentuk kunci enkripsi setelah proses pembangkitan kunci. Penggunaan PBKDF2 ini bertujuan untuk memperlambat dari serangan *brute-force* saat dilakukan dekripsi. Hal ini dapat menambah kualitas dari *confidentiality* dari pesan tersebut. Konsekuensi penggunaan PBKDF2 ini, parameter pertama dari fungsi KDF pada ECIES menjadi wajib untuk diisi.

Usulan kedua terkait dengan penggunaan *additional data* pada proses enkripsi sebagai parameter kedua. Keuntungan menggunakan metode ini adalah proses otentikasi dari parameter opsional ini dapat dilakukan secara bersamaan dengan proses dekripsi dari pesan. Hal ini dapat mengefisiensikan proses otentikasi pesan pada parameter kedua tersebut.

B. Pengaturan Hak Akses pada Berkas Terenkripsi

Operasi yang dapat dilakukan pada sebuah berkas diantaranya adalah operasi untuk melakukan pembacaan berkas ataupun pengubahan dari berkas tersebut. Penerapan kedua operasi ini pada berkas terenkripsi dapat dilakukan dengan memanfaatkan algoritma kunci publik dan juga algoritma kunci simetris. Algoritma kunci simetris digunakan pada saat melakukan enkripsi pada berkas tersebut. Kunci dari berkas tersebut lalu dapat dienkripsi menggunakan algoritma kunci publik sesuai dengan partisipan yang dapat membaca berkas tersebut.

Permasalahan dari implementasi tersebut adalah setiap partisipan yang dapat membaca berkas tersebut dapat melakukan pengubahan. Untuk mencegah hal tersebut, dapat digunakan mekanisme *digital signature*. Tanda tangan digital untuk integritas berkas dibuat setiap kali proses enkripsi dilakukan pertama kali. Tanda tangan digital ini untuk memberikan informasi terkait dengan integritas berkas tersebut. Bila terdapat perubahan pada berkas tersebut, proses dekripsi dapat langsung dibatalkan.

Informasi kunci privat dari tanda tangan berkas tersebut dapat disimpan sebagai hak akses timpa untuk berkas tersebut. Setiap kali proses timpa dilakukan, tanda tangan digital ini harus diperbaharui kembali. Hal ini agar partisipan tetap dapat membaca berkas tersebut.

Informasi terkait dengan kunci publik berkas, kunci dekripsi berkas, kunci tanda tangan berkas perlu ditandatangani oleh pembuat berkas. Hal ini menyatakan integritas dari hak akses yang diberikan oleh pembuat berkas. Informasi tersebut disebut dengan *header file*. Informasi pada *header file* dan pesan terenkripsi juga perlu dilakukan proses penandatanganan digital menggunakan kunci privat dari berkas. Secara garis besar, rancangan proses enkripsi menggunakan skema ini ditunjukan pada gambar 5.

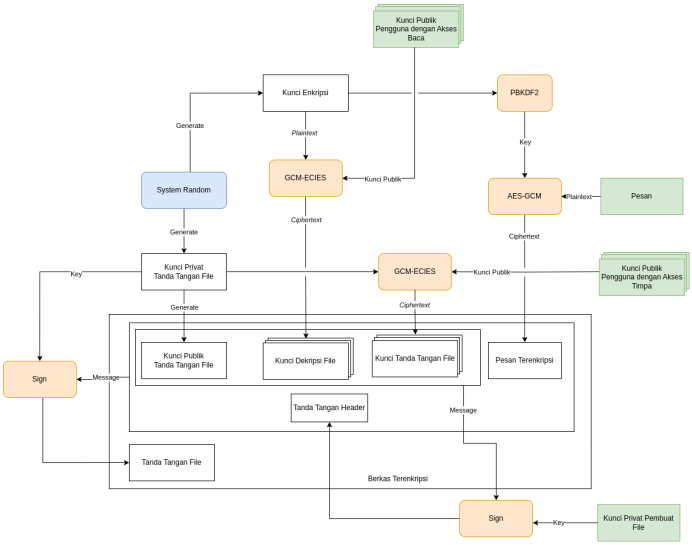

Fig. 5. Rancangan Proses Enkripsi Berkas

Bila dilihat pada gambar 5, kunci publik pengguna yang akan diberi akses dapat lebih dari satu. Proses enkripsi ECIES dilakukan untuk tiap-tiap kunci publik yang diberikan. Oleh karena itu, proses enkripsi berkas ini disebut dengan *multiparticipant party*.

C. Struktur Data Berkas

Berkas yang terenkripsi perlu didefinisikan terkait dengan struktur data yang digunakan. Hal ini untuk memberikan informasi terkait dengan berkas berkas tersebut. Seperti yang telah dijelaskan pada bagian III-B, terdapat dua bagian utama pada berkas, yaitu *header file* dan pesan terenkripsi. Informasi yang terdapat pada berkas mencakup informasi berikut:

1) *Decryption Key Dictionary*
   Informasi pada bagian ini menjelaskan terkait dengan kunci dekripsi yang digunakan untuk membaca berkas tersebut. Struktur data yang digunakan adalah *dictionary* yang berisi pasangan hash dari kunci publik pengguna dan kunci dekripsi yang telah dilakukan enkripsi dengan kunci publik tersebut. Hash yang digunakan adalah SHA-256.
2) *Signing Key Dictionary*
   Informasi pada bagian ini menjelaskan terkait dengan kunci tanda tangan digital yang digunakan untuk menandatangani berkas tersebut. Struktur data yang digunakan adalah *dictionary* yang berisi pasangan hash dari kunci publik pengguna dan kunci tanda tangan digital yang telah dilakukan enkripsi dengan kunci publik tersebut. Hash yang digunakan adalah SHA-256. Tanda tangan digital ini digunakan untuk menimpa pesan terenkripsi pada berkas tersebut.
3) *File Public Key*
   Kunci publik ini merupakan kunci publi dari tanda tangan digital file. Kunci ini digunakan untuk memverifikasi integritas *header* dan pesan terenkripsi.
4) Tanda Tangan Pemilik
   Tanda tangan digital pemilik berkas ini mencakup poin-poin di atas. Tanda tangan ini digunakan untuk memverifikasi integritas dari *header* file. Pengujian terhadap tanda tangan ini diperlukan untuk memastikan bahwa hak akses yang diberikan oleh pemilik berkas memang valid.
5) Pesan Terenkripsi
   Pesan terenkripsi ini merupakan pesan yang telah dienkripsi menggunakan algoritma kunci simetris. Pesan ini juga telah diotentikasi menggunakan mode blok GCM.
6) Tanda Tangan File
   Tanda tangan ini mencakup lima poin yang ada pada bagian di atas. Tanda tangan ini digunakan untuk memverifikasi integritas dari berkas tersebut.

Struktur data ini bila diilustrasikan dalam bentuk diagram akan terlihat seperti pada gambar 6. Pada diagram tersebut terdapat tiga buah *nonce* yang berbeda. Fungsi dari *nonce* ini adalah untuk menghindari adanya serangan *dictionary attack* pada struktur data *decryption key dictionary* dan *signing key dictionary*. Ancaman ini dapat terjadi dikarenakan kunci publik yang digunakan oleh pengguna dapat diketahui oleh pihak lain dengan mudah. Oleh karena itu, *nonce* ini digunakan untuk mengacak nilai kunci dekripsi dan kunci tanda tangan digital yang digunakan. Akibat dari proses ini, serangan *dictionary attack* tidak akan mangkus untuk dilaksanakan.

IV. IMPLEMENTASI

Implementasi dari rancangan sistem ini memanfaatkan bahasa pemrograman Go. Pemilihan dari bahasa pemrograman ini dikarenakan Go memiliki dukungan yang cukup baik terhadap kriptografi. Implementasi dari rancangan sistem ini akan dibagi menjadi beberapa bagian, yaitu:

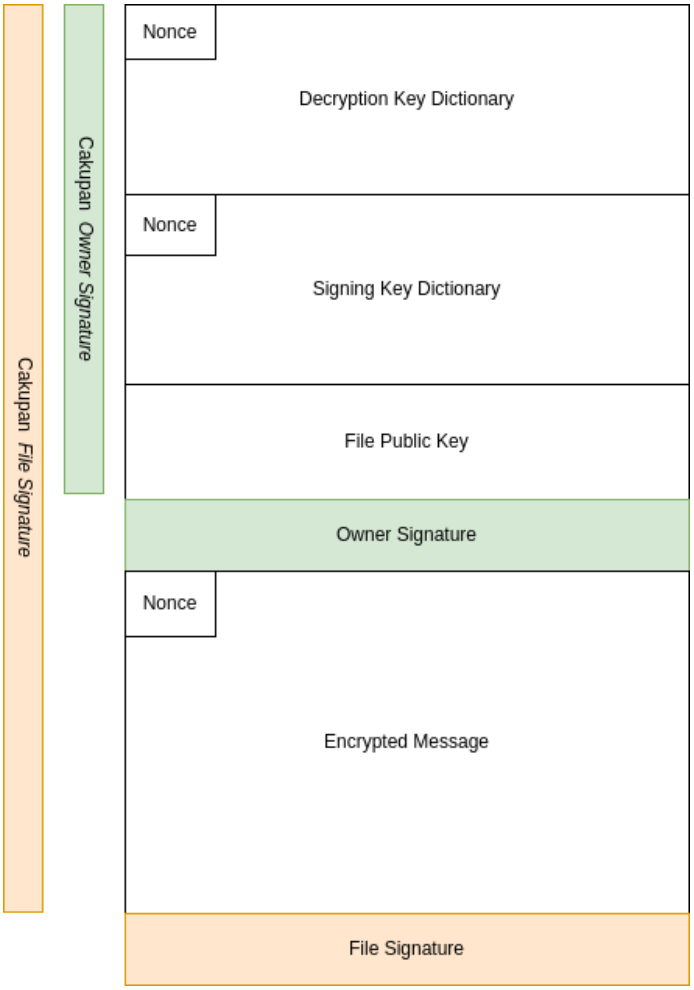

Fig. 6. Struktur Data Berkas Terenkripsi

1) Implementasi dari algoritma ECIES.
2) Implementasi dari tanda tangan digital ECDSA.
3) Serialisasi Struktur Data.

A. *Implementasi Algoritma ECIES*

Algoritma ECIES diimplementasikan dengan memanfaatkan pustaka ECDH. Pemilihan dari pustaka ini dikarenakan pustaka ini memiliki karakteristik yang mirip dengan ECIES. Pustaka ini juga memiliki dukungan yang cukup baik pada berbagai bahasa. Kunci privat dari parameter ECDH disimpan dalam format PKCS#8. Kunci publik dari parameter ECDH disimpan dalam format PKIX.

Pada implementasi yang dilakukan, terdapat beberapa parameter yang diatur. Parameter tersebut adalah sebagai berikut:

1) Kurva Eliptik
   Jenis kurva eliptik yang digunakan pada operasi ECIES adalah kurva eliptik P-256. Kurva eliptik ini dipilih dikarenakan kurva eliptik ini memiliki panjang kunci yang pendek serta memiliki kualitas yang cukup baik.
2) *Seed* dari PBKDF2
   Seed yang digunakan pada PBKDF2 adalah nilai acak yang dihasilkan dari sistem operasi. Seed ini digunakan untuk memperkuat kunci yang dihasilkan dari PBKDF2. Seed ini pada akhirnya akan disimpan berdampingan dengan berkas yang dienkripsi.
3) Paremeter #2 pada ECIES
   Parameter ini digunakan sebagai otentikasi dari tanda tangan pemilik pesan. Parameter ini akan memastikan bahwa tanda tangan pemilik belum berubah sejak terakhir kali berkas tersebut dienkripsi. Bila telah berubah, proses dekripsi akan gagal.

B. *Implementasi Tanda Tangan Digital ECDSA*

Tanda tangan digital ECDSA diimplementasikan dengan memanfaatkan pustaka ECDSA. Pemilihan dari pustaka ini dikarenakan pustaka ini memiliki dukungan yang cukup baik pada berbagai bahasa. Kunci privat dari parameter ECDSA disimpan dalam format PKCS#8. Kunci publik dari parameter ECDSA disimpan dalam format PKIX. Jenis kurva eliptik yang digunakan pada operasi ECDSA adalah kurva eliptik P-256. Kurva eliptik ini dipilih dikarenakan kurva eliptik ini memiliki panjang kunci yang pendek serta memiliki kualitas yang cukup baik.

C. *Serialisasi Stuktur Data*

Struktur data berkas diimplementasikan dengan melakukan konkatenasi dari larik byte yang ada. Struktur data saat menyatukan larik byte tersebut ditunjukan pada figur 7.

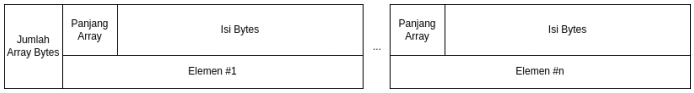

Fig. 7. Struktur Data Berkas Terenkripsi

Pada gambar tersebut, ditunjukan bahwa larik akan diawali dengan jumlah elemen dari larik tersebut. Hal ini dilakukan untuk mempermudah proses deserialisasi dari struktur data tersebut. Besar dari tiap elemen byte juga akan disertakan pada larik tersebut. Hal ini dilakukan untuk menghemat ukuran dari struktur data tersebut.

V. Pengujian

Pengujian dari rancangan sistem ini dilakukan dengan memanfaatkan beberapa skenario. Ketentuan yang perlu dipenuhi pada saat pengujian adalah sebagai berikut:

1) Program dapat mengenkripsi berkas dengan benar.
2) Program dapat mendekripsi berkas dengan benar.
3) Program dapat menolak melakukan dekripsi pada pengguna yang tidak memiliki hak akses.
4) Program dapat menolak melakukan operasi timpa pada pengguna yang tidak memiliki hak akses.
5) Program dapat melakukan operasi timpa.
6) Program dapat melakukan dekripsi pesan pada file yang telah ditimpa.

Dari hasil pengujian menunjukan bahwa keenam dari skenario tersebut terpenuhi. Hal ini dapat menyimpulkan bahwa teknik ini dapat digunakan untuk mengamankan berkas yang ada. Bukti dari pengujian ini dapat dilihat pada repository kode implememtasi. Bukti ini disimpan pada folder bernama `test`.

REFERENCES

[1] Munir, Rinaldi. 2019. Kriptografi. Bandung: Informatika.

[2] Ali, S., & Abdul, A. (2016). Data Security for cloud computing based on Elliptic Curve Integrated Encryption Scheme (ECIES) and modified identity based cryptography (MIBC). International Journal of Applied Information Systems, 10(6), 7-13. https://doi.org/10.5120/ijais2016451517.

[3] Abdalla, M., Bellare, M., & Rogaway, P. (2021). DHIES: An Encryption Scheme Based on the Diffie-Hellman Problem.

[4] Rogaway, Phillip. 2002. Authenticated-encryption with Associated-data. In Proceedings of the 9th ACM conference on Computer and communications security (CCS'02). Association for Computing Machinery, New York, NY, USA, 98-107. https://doi.org/10.1145/586110.586125.

[5] McGrew, D.A., & Viega, J. (2005). The Galois/Counter Mode of Operation (GCM). https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf.

[6] Martinez, V. G., Encinas, L. H., & Ávila, C. S. (2010). A Survey of the Elliptic Curve Integrated Encryption Scheme. Journal of Computer Science and Engineering, 2(2), 7-13.

[7] Kaliski, B. (2000). PKCS #5: Password-Based Cryptography Specification Version 2.0. https://doi.org/10.17487/rfc2898.

[8] Raeburn, K. (2005). Advanced Encryption Standard (AES) Encryption for Kerberos 5. https://doi.org/10.17487/rfc3962.

REPOSITORI KODE

Implementasi dari sistem ini dapat dilihat pada repositori kode berikut: https://github.com/bayusamudra5502/multiparticipant-encryptor

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, 12 Juni 2024

Bayu Samudra