Mengamankan File .env dalam Pengembangan Aplikasi Secara Kolaboratif dengan Algoritma ElGamal pada Kurva Eliptik

Muhammad Garebaldhie ER Rahman - 13520029 (*Penulis*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): mgarebaldhie80@gmail.com,, 13520029@std.stei.itb.ac.id

***Abstract*— Pada proses pengembangan aplikasi, file (dot)env seringkali digunakan untuk menyimpan informasi sensitif mengenai konfigurasi aplikasi yang di buat. Menjaga keamanan .env menjadi sangat penting karena kebocoran informasi sensitif dalam file .env dapat memiliki konsekuensi serius bagi keamanan aplikasi dan data yang terkait. Untuk meningkatkan keamanan pada .env dapat digunakan algoritma elgamal pada kurva eliptik untuk mengenkripsi dan dekripsi data yang ada pada .env tersebut**

Keywords—Konfigurasi* (dot)env*; Kriptografi kunci publik; ElGamal; Kurva Eliptik; Enkripsi; Dekripsi

I. PENDAHULUAN

Aplikasi telah menjadi bagian integral dari kehidupan kita dalam era digital saat ini. Dari perangkat seluler hingga komputer desktop, aplikasi hadir dalam berbagai bentuk dan memberikan pengalaman yang beragam kepada pengguna. Dalam konteks ini, pengembangan aplikasi telah menjadi bidang yang dinamis dan inovatif, terus menghadirkan solusi baru untuk memenuhi kebutuhan pengguna modern.

Karena penggunaan aplikasi yang semakin tinggi, para pengembang aplikasi harus juga harus memperhatikan dan menyajikan aplikasi memiliki keamanan yang tinggi sehingga pengguna merasa aman ketika menggunakan aplikasinya.

Dalam proses pengembangan aplikasi, pengembang dapat menyimpan konfigurasi aplikasi tersebut di banyak tempat. Pengembang dapat menyimpan di sebuah config file ataupun langsung menyimpannya di *environment variables* tempat aplikasi tersebut berjalan. *Config file* yang tersedia juga bermacam macam format nya, ada yang menggunakan format TOML (Tom's Obvious Minimal Language), ada juga yang menggunakan format YAML (Yet Another Markup Language) yang berfokus untuk menyediakan konfigurasi yang *human readable*. Namun konfigurasi yang sering digunakan adalah konfigurasi yang memiliki format (dot)env karena penggunaannya yang sangat mudah sehingga dapat mempercepat penyediaan dan penggunaan konfigurasi.

Namun, selain kelebihan yang ditawarkan oleh (dot)env, (dot)env memiliki kekurangan yaitu konfigurasinya tidak boleh di unggah ke internet sehingga hal ini akan sangat menyulitkan proses kolaborasi khususnya untuk para pengembang baru.

Makalah ini bertujuan untuk mempermudah proses kolaborasi tersebut dengan mengamankan file (dot)env dengan mengenkripsi menggunakan algoritma Elgamal pada kurva eliptik, Sehingga file tersebut dapat di unggah ke *github* atau platform kolaborasi lainnya.

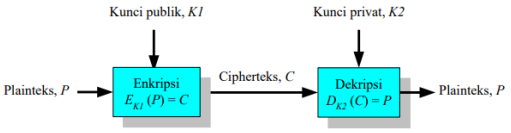
II. DASAR TEORI

A. Kriptografi kunci publik

Algoritma kunci publik seringkali disebut kriptografi kunci-nirsimetri *(asymmetric-key cryptography)* karena kunci yang digunakan untuk proses enkripsi dan deskripsi berbeda. algoritma kunci publik ini menyelesaikan masalah pada kunci simetri karena perlu dilakukan pengiriman kunci pada saluran yang dapat dipastikan keamanannya.

Dasar teori di balik kriptografi kunci publik didasarkan pada masalah matematis yang sulit dipecahkan, seperti faktorisasi bilangan besar atau komputasi diskrit logaritma. Dengan memanfaatkan properti matematis ini, sistem kriptografi kunci publik memungkinkan pengguna untuk menghasilkan pasangan kunci yang terdiri dari kunci publik dan kunci privat.

Ide dari algoritma kunci publik ini yaitu pengguna yang memiliki sepasang kunci privat dan publik dapat kita sebut K1 dan K2, kunci publik (K1) akan digunakan untuk mengenkripsi pesan yang akan dikirim sedangkan kunci privat (K2) akan digunakan untuk mendekripsi pesan yang diterima[1]



Gambar 1. Skema algoritma kunci publik[1]

Dengan menggunakan skema seperti ini, kunci yang akan digunakan untuk mengenkripsi pesan sehingga kunci dapat dikirim melalui saluran yang tidak aman. Hal ini juga akan

mempersulit kriptanalis sehingga serangan mitm akan semakin sulit untuk dilakukan.

Skema kunci publik ini memiliki beberapa keuntungan diantaranya

1. Hanya kunci privat yang perlu dijaga kerahasiaannya dan kunci privat tidak dibutuhkan untuk proses enkripsi
2. Pasangan kunci public dan kunci privat tidak perlu sering diubah, bahkan dalam periode waktu yang panjang.
3. Dapat digunakan untuk mengamankan pengiriman kunci simetri

Selain kelebihan yang disebutkan, algoritma ini juga memiliki beberapa kelemahan diantaranya

1. Enkripsi dan dekripsi pesan umumnya lebih lambat daripada sistem kriptografi simetri, karena enkripsi dan dekripsi menggunakan bilangan yang besar dan melibatkan operasi perpangkatan yang besar
2. Ukuran cipherteks lebih besar daripada plainteks (bisa dua sampai empat kali ukuran plainteks).
3. Ukuran kunci relatif lebih besar daripada ukuran kunci simetri.
4. Karena kunci publik diketahui secara luas dan dapat digunakan setiap orang, maka cipherteks tidak memberikan informasi mengenai otentikasi pengirim.
5. Tidak ada algoritma kunci-publik yang terbukti aman (sama seperti block cipher). Kebanyakan algoritma mendasarkan keamanannya pada sulitnya memecahkan persoalan-persoalan aritmetik (pemfaktoran, logaritmik, dsb) yang menjadi dasar pembangkitan kunci.

B. ECDLP

Salah satu permasalahan algoritma kunci publik didasari dari persoalan logaritma diskrit, persoalan logaritma diskrit mengharuskan kita mencari x yang memenuhi persamaan berikut

$$g^x = y \,(mod\, p) \qquad (1)$$

Jika meninjau pada kurva eliptik, permasalahan ini berubah yang awalnya dari DLP (*Discrete Logarithm Problem)* menjadi ECDLP (*Discrete Logarithm Problem)*. Pada ECDLP, karena bekerja pada kurva eliptik, permasalahan di rubah untuk mencari sebuah kunci privat k yang memenuhi persamaan berikut dimana Q merupakan kunci publik dan P merupakan titik yang telah ditentukan pada generator

$$Q = kP \qquad (2)$$

Karena nilai K merupakan angka yang besar, maka mencari nilai kP sangatlah sulit. karena pada kurva eliptik operasi perkalian merupakan operasi pangkat

$$kP = P^k = Q \qquad (3)$$

Sehingga persoalani ini kembali ke persoalan logaritma diskrit awal yaitu mencari k jika diberikan Q dan P.

C. Akar primitif

Jika n merupakan bilangan bulat, maka a disebut sebagai akar primitif dari n jika

$$a,\ a^2,\ \ldots,\ a^{tot(n)}\ (mod\ n) \qquad (4)$$

Menghasilkan nilai yang berbeda dan bukanlah sebuah kelipatan yang berulang yang merupakan bilangan relatif prima dengan n

Secara khusus, jika p adalah bilangan prima, maka a dapat disebut sebagai akar primitif dari p jika

$$a,\ a^2,\ \ldots,\ a^{p-1}\ (mod\ p) \qquad (5)$$

Sebagai contoh, misalkan $p = 7$. $a = 3$ merupakan sebuah akar primitif dari 7 karena

$$1 \le x < p - 1\ pada\ 3^x \qquad (6)$$

menghasilkan nilai yang berbeda dan tidak berulang.

x	3^x	$3^x\ mod\ 7$
1	3	3
2	9	2
3	27	6
4	81	4
5	243	5
6	729	1
7	2187	3
8	6561	2

Dapat dilihat untuk x yang memiliki nilai 1 hingga 6 memiliki nilai yang berbeda dan ketika x = 7,8 dan seterusnya memperlihatkan pola yang berulang.

D. ElGamal

ElGamal merupakan algoritma yang dibuat oleh Taher ElGamal (1985). Algoritma elgamal termasuk kedalam algoritma kunci publik yang didasarkan dari persoalan logaritma diskrit yang dapat pada persamaan (1).

Berikut merupakan properti yang ada pada ElGamal[2]

1. Bilangan prima, p (publik)
2. Bilangan acak, g (g < p, g adalah akar primitif dari p) (tidak rahasia)
3. Bilangan acak, x ($2 \leq x \leq p-1$) (rahasia, kunci privat)
4. $y = g^x \pmod p$ (tidak rahasia, kunci publik)
5. m (plainteks) (rahasia, masukan)
6. a dan b (cipherteks) (tidak rahasia

Hasil dari algoritma ElGamal ini merupakan sebuah triplet dan tuple

1. Kunci Publik: triple (y, g, p)
2. Kunci Privat: tuple (x, p)

Prosedur enkripsi pada ElGamal mengharuskan membagi pesan menjadi beberapa bagian blok yang berada pada selang

$$0 \leq m_i \leq p-1, \; untuk\ setiap\ i$$

Setelah pesan dibagi, diambil sebuah bilangan acak k. lalu setiap blok pesan akan di enkripsi dengan cara

$$a = g^k \pmod p \qquad (7)$$

$$b = y^k m \pmod p \qquad (8)$$

Hasil dari proses enkripsi ini merupakan pasangan (a, b) sehingga ukuran *ciphertext* akan menjadi dua kali lebih besar dibandingkan *plaintext* nya.

Proses dekripsi nya akan menggunakan kunci privat x dan p. Untuk setiap *ciphertext* dapat didekripsi dengan cara membagi b dengan modulo invers dari a pangkat x

$$a^{(x)-1} = a^{p-1-x} \text{ (mod p)} \quad (9)$$

Hitung *plaintext* dengan cara membagi b dengan (9)

$$m = b * a^{(x)-1} \text{ (mod p)} \quad (10)$$

E. Grup

Group merupakan sebuah sistem aljabar yang terdiri dari dua properti yaitu[3]

1. Sebuah himpunan G
2. Sebuah operasi biner *

Sebuah grup haruslah berlaku keempat aksioma[3] berikut yaitu

1. Closure: a * b harus berada di dalam G
2. Asosiatif: a * (b * c) = (a * b) * c
3. Elemen netral: Haruslah terdapat sebuah $e \in G$ sehingga $a * e = e * a = a$
4. Elemen invers: terdapat $a' \in G$ sehingga $a' * a = a * a' = e$

Sebuah grup dapat dinotasikan sebagai $< G, * >$. $< R, + >$, $< Z, + >$, $dan < Z, \bullet >$ merupakan sebuah grup karena memenuhi keempat aksioma yang telah disebutkan diatas

Sebuah grup dapat dikatakan abelian (komutatif) jika berlaku aksioma komutatif yaitu

$$a * b = b * a,\ a, b \in G$$

F. Medan

Medan (field) adalah himpunan elemen (disimbolkan dengan F) dengan dua operasi biner, biasanya disebut penjumlahan (+) dan perkalian (•).

Sebuah struktur aljabar $< F, +, \bullet >$dapat dikatan medan jika dan hanya jika[3]

1. $< F, + >$ adalah grup abelian
2. $< F - \{0\}, \bullet >$ adalah grup abelian
3. Opearsi • menyebar terhadap operasi + (sifat distributif)

Sebuah medan dikatakan berhingga jika himpunannya memiliki jumlah elemen yang berhingga. Medan berhingga sering kali disebut sebagai **Galois Field**. Selain itu medan berhingga F_p harus memenuhi dua operasi berikut

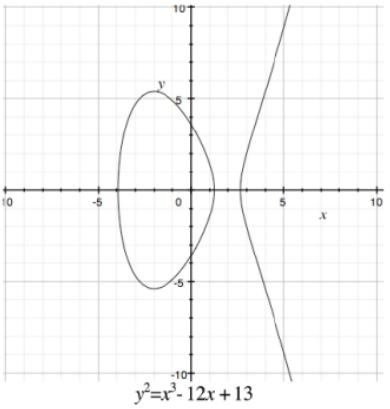
1. Penjumlahan
   Jika $a, b \in F_p$, dapat dicari sebuah r yang memenuhi r = (a + b) mod p, $0 \leq r \leq p-1$
2. Perkalian
   Jika $a, b \in F_p$, dapat dicari sebuah s yang memenuhi s = (a • b) mod p, $0 \leq r \leq p-1$

G. Kurva eliptik

Kurva eliptik merupakan sebuah kurva yang memiliki bentuk umum (11) dan memenuhi syarat (12)

$$y^2 = x^3 + ax + b \text{ (mod p)} \qquad (11)$$

$$4a^3 + 27b^2 \neq 0 \text{ (mod p)} \qquad (12)$$

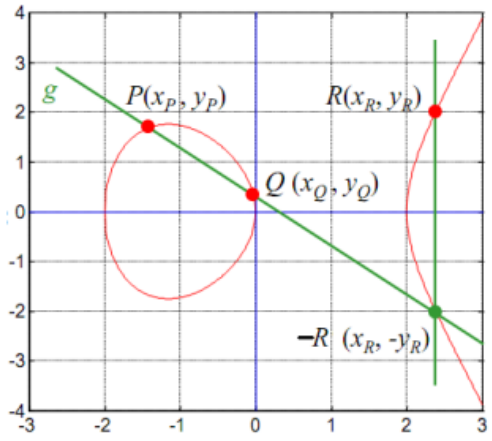


Gambar 2. Contoh kurva eliptik dengan persamaan $y^2 = x^3 - 12x + 13$ [3]

Gambar 3. Contoh beberapa kurva eliptik lainnya.[3]

Kurva eliptik mendefinisikan titik yang memiliki nilai y yang tak terhingga sebagai titik $O(x, \infty)$. Semua titik yang berada pada kurva eliptik membentuk sebuah grup dengan operasi penjumlahan (+).

Kurva eliptik memiliki beberapa aturan untuk penjumlahan maupun perkalian titik. Untuk penjumlahan titik jika diberikan titik $P(x_p, y_p)\ dan\ Q(x_q, y_p)$ dapat dicari sebuah titik $R(x_r, y_r)$ yang merupakan hasil penjumlahan dari titik P dan Q.



Gambar 4. Visualisasi penjumlahan pada kurva eliptik[3]

Kita dapat mencari gradient dari garis yang dibentuk oleh kedua titik P dan Q

$$m = \frac{y_p - y_q}{x_p - x_q} \text{ (mod p)}, p \neq q \text{ (13)}$$

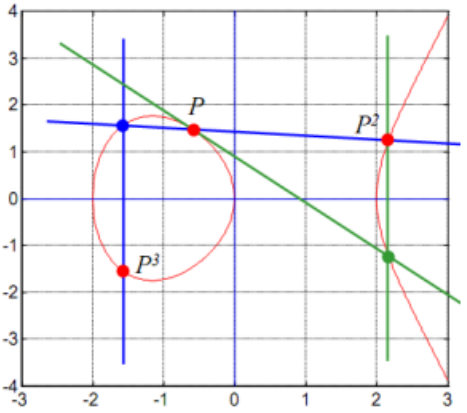
$$m = \frac{3x_p^{\ 2} + a}{2y_p} \text{ (mod p)}, p = q \text{ (14)}$$

Setelah mendapat gradiennya kita dapat mencari titik r dengan memanfaatkan gradien yang telah ditemukan

$$x_r = m^2 - x_p - x_q \quad \text{(mod p) (15)}$$

$$y_r = m\ (x_p - x_r)\ -\ y_p \quad \text{(mod p) (16)}$$

Untuk perkalian titik pada kurva eliptik konsepnya memiliki konsep yang sama seperti penjumlahan dan diulang sebanyak k kali untuk k skalar, p dan q merupakan titik pada kurva yang dapat dilihat pada persamaan (2) dan (3)



Gambar 5. Visualisasi perkalian pada kurva eliptik[3]

ECC memiliki kelebihan yaitu memiliki kunci yang lebih pendek namun memiliki tingkat keamanan yang sama dengan RSA yang memiliki kunci yang lebih panjang. Keamanan ECC dengan panjang 256bit memiliki keamanan yang sama dengan RSA dengan panjang kunci 3072 bit. Hal ini dapat mempercepat komputasi karena angka yang harus dioperasikan tidak besar.

H. ECEG (Elliptic Curve Elgamal)

ECEG merupakan sebuah sistem kriptografi kurva eliptik yang diadopsi menggunakan algoritma Elgamal[3], [4]. ECEG mengadopsi masalah yang telah dijelaskan pada bagian sebelumnya yaitu ECDLP.

Berikut merupakan langkah langkah yang perlu dilakukan untuk mengenkripsi pesan. Misalkan terdapat dua pihak yaitu Alice dan Bob yang memiliki pesan M yang akan dienkripsi.

1. Alice dan Bob memiliki titik point awal yang telah disepakati sebelumnya (titik generator) misalkan kita sebut B
2. Alice dan Bob perlu membuat kunci privat dan publiknya masing masing dengan cara mengalikan kunci privatnya masing masing a dan b dengan titik generator B yang merupakan perkalian titik pada kurva eliptik.

$$A = a * B \quad \text{(mod p) (17)}$$

$$B = b * B \quad \text{(mod p) (18)}$$

3. Alice ingin mengenkripsi pesan M, lalu mentransformasi pesan tersebut menjadi sebuah titik pada kurva P_m
4. Alice memilih bilangan acak k yang berada pada selang $1 \leq k \leq p - 1$
5. Hasil enkripsi akan berupa pasangan titik yaitu $P_c = (kB, P_m + kP_b)$
6. Jika bob menerima pesan alice dan ingin mendekripsi pesan, bob akan menghitung hasil kali titik pertama

dari P_c. Lalu bob perlu mengurangkan tuple kedua dengan hasil kali tersebut

a. Hitung P_c kali dengan B

b. Kurangkan $P_m + kP_b$ dengan kbB

$$P_m = P_m + kP_b - kbB = P_m + kbB - kbB \quad (19)$$

Sehingga plaintext P_m dapat di *recover* dari perhitungan (19)

III. IMPLEMENTASI

A. Kurva eliptik

Pendefinisian kurva eliptik dilakukan dengan mengikuti standard spesifikasi kurva **p521** pada NIST[5]

```
p =
0x01ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff

a =
0x01fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffc

b =
0x0051953eb9618e1c9a1f929a21a0b68540eea2da725b99b
315f3b8b489918ef109e156193951ec7e937b1652c0bd3bb1
bf073573df883d2c34f1ef451fd46b503f00
```

Dengan ketiga parameter tersebut kurva eliptik $y^2 = x^3 + ax + b \ (mod\, p)$ sudah terdefinisi. Selanjutnya titik generator point juga dipilih dari spesifikasi NIST[5]

```
G =
(0x00c6858e06b70404e9cd9e3ecb662395b4429c64813905
3fb521f828af606b4d3dbaa14b5e77efe75928fe1dc127a2ffa
8de3348b3c1856a429bf97e7e31c2e5bd66,
0x011839296a789a3bc0045c8a5fb42c7d1bd998f54449579
b446817afbd17273e662c97ee72995ef42640c550b9013fad0
761353c7086a272c24088be94769fd16650)
```

Titik G merupakan sebuah titik generator yang telah disepakati bersama. Titik ini berupa pasangan titik x dan y yang bergabung menjadi sebuah tuple.

Selain itu dipilih juga sebuah order n dan sebuah cofactor dari kurva eliptik yang juga mengikuti spesifikasi dari NIST[5]

```
n =
0x01fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffa51
868783bf2f966b7fcc0148f709a5d03bb5c9b8899c47aebb6f
b71e91386409
```

```
h = 0x1
```

B. ECEG

Setelah kurva eliptik terdefinisi. maka proses enkripsi dengan menggunakan ECEG dapat dilakukan. Prosesnya dapat dibedakan menjadi dua proses yaitu enkripsi dan dekripsi

Sebelum melakukan enkripsi maupun deskripsi ECEG perlu *generate private* dan *public* key yang akan digunakan. Berikut merupakan alur program.

1. Program meminta masukan password yang berfungsi sebagai private key (x)
2. Password tersebut akan diubah *encoding* nya menjadi angka dan disimpan sebagai *private key (x)*.
3. Program akan menghitung *public key (Q)* nya dengan cara mengalikan *private key* dengan generator point (G)

Setelah kunci publik dan privat tersedia, selanjutnya masuk ke tahap enkripsi dan dekripsi. Secara umum tahap enkripsi bekerja sebagai berikut

1. Program akan memilih nilai k random yang berada pada rentang 1 hingga orde n.
2. Program akan menghitung nilai C1 yang merupakan hasil perkalian titik generator G dengan nilai k tadi dan pastikan titik C1 berada pada kurva eliptik.

$$C1 = kG \quad (20)$$

3. Pesan yang ingin dienkripsi harus dibagi sehingga masuk ke dalam rentang $1 \leq m \leq p - 1$ lalu di *encode* menjadi titik pada kurva.
4. Program akan menghitung nilai C2 yang merupakan perkalian kunci publik dengan nilai k ditambah dengan pesan

$$C2 = kQ + P_m \quad (21)$$

5. Hasil enkripsi berupa pasangan tuple C1 dan C2

Sementara itu proses deskripsi bekerja sebagai berikut.

1. Program akan menerima input C1 dan C2
2. Program akan menghitung hasil pengurangan C2 dengan C1

$$m = C2 - C1 * x \quad (22)$$

3. Karena pada kurva eliptik nilai y bisa berperiodik maka kita hanya perlu mengambil absisnya (x) saja pada nilai m agar dapat menghasilkan pesan yang diinginkan

C. Konfigurasi file (dot)env

File(dot)env merupakan pasangan *key value* yang dipisahkan dengan tanda =. Program dapat menerima input berupa path dari file .env yang ingin di enkripsi. Program tidak akan mengenkripsi key nya hanya nilai dari value nya saja yang akan di enkripsi

D. Proses enkripsi & enkripsi (dot)env

Setelah semua terkumpul, maka untuk mengenkripsi file (dot)env, akan dilakukan pemisahan nilai key dan value pada file tersebut. Setelah itu value dari setiap key di enkripsi dengan menggunakan enkripsi ECEG lalu file hasil tersebut ditulis kembali ke file dengan ekstensi .enc untuk menandakan bahwa file tersebut telah di enkripsi

IV. PENGUJIAN DAN PEMBAHASAN

A. Pengujian kode program

Untuk menguji kode program akan dibuat file .env seperti berikut ini

.env

```
AKU=TAMPAN
ASISTEN=KEREN
NILAI_KRIPTO=A
```

Password yang dimasukan yaitu `gare` dan kode akan menghasilkan kunci publik sebuah titik Q pada kurva sebagai berikut

```
Your public key is:
(407908480567476787588061448644961912027079314923404889456411244029678960679836194785142418472554966229664422383267866375460928763023321766478090792773384771,
6831886733599083837618126953930796067933958221741926163948145309730602494102152318688923544101033448059540633703052931189024798669454369011214159316669937 25)
```

Berikut merupakan hasil enkripsi dengan kunci `gare`, hasil enkripsi merupakan pasangan C1 dan C2 untuk setiap pasangan kunci pada env

.env.enc

```
AKU=('0085620d68b4320466b8a0f311601a34f9d0b285427e24a88e42a13f197c2af1aec3861128028d45b1af87cac73b0fc4e2366a1233cebf2144badb13a49d3d3e7f44004316ea1e41b0d3e3558114dca821b9a1ab892d9e92c537c1034440d54df3cea19a5bdfb996e9ad119c4806dda076eb661919572dced1503e19f86006e6253b3e57',
'0149be03a55dc4b319e8045a3699feb8699a766248e14a31895e5a6c0af9f08dc83a2ae2726109a2c795ba249bc7ddc2f50e20947c86f92bb2e4b5b12e4a22021fe5')
ASISTEN=('0071111b4bf81eedcfc45672079d4ff66de23219dafd293bd56b6c2c48305413faac5740809b20f421e65c7504f37bf9abee2f9facf2b151905aa6d53c762af9ea25014c49a13885608d0194584a3ea1a8dfea3eadc8ef83871833d3f0e93a800dd2f9660ab480c2b203136169b1a33993e3955ce0d4697277eb0b11e813af68144bc728',
'd7b57e8d934a00e3f74574ca5de1918a5cd4c70d2f1ea4a818e40c71986332806f5c690d859c0f18f04bdbde7fe46d8949eea555e05b944a7517538e475cd304436')
NILAI_KRIPTO=('0025cec14e849b3bc559714889ddc76c0ffb88cebb6aed6b788348fc7fa9803df2d33a10b57f057115273e8b07a36e0ecc7d267d275183b3259a6b1ff0979fcca909019ed12bcd8df8f7c1e297fa6eba485d7699cf24ad3edf9ce7ebda7861ee2a2d385ab9a433548c6dddc53eca1bf7dd40fd4f9849efa04be66ed3eac616b5529296ac',
'14b293f8739d40da661b9c03b0d15acd2fc9619bac52903d8dc6ce5c04d2e3f870aa05360026f2a237065f1e84a5654176124d525cb12bb414c6b001b5a5a9d51b')
```

Akan dicoba untuk melakukan dekripsi file .env.enc dengan password yang salah. Password yang dimasukan yaitu `gare123`. Program akan menghasilkan tulisan wrong password untuk setiap key value dan .env kosong'

```
Wrong password, decryption failed
Wrong password, decryption failed
Wrong password, decryption failed
```

Apabila digunakan dengan password yang benar yaitu `gare` maka akan menghasilkan kembali file .env semula sehingga proses enkripsi dan dekripsi berhasil.

.env

```
AKU=TAMPAN
ASISTEN=KEREN
NILAI_KRIPTO=A
```

B. Kriptanalisis dan keamanan

Pada proses generate key, key merupakan sebuah input user yang berada pada rentang $1 \leq x \leq n-1$, n merupakan order dari kurva eliptik tersebut. Jika kriptanalisis ingin melakukan brute force untuk mencari nilai kunci privat nya maka mereka harus menemukan nilai x sehingga memenuhi persamaan (23). Persamaan 23 merupakan permasalahan ECDLP yaitu persoalan logaritma diskrit.

$$xG = G^x = Q \bmod p \qquad (23)$$

Untuk menemukan x yang mungkin berarti kita harus mencoba nilai x dari 1 hingga n - 1 dan memangkatkan nilai tersebut dengan point generator G yang apabila di modulo dengan p menghasilkan Q akan memakan waktu yang besar.

Asumsi untuk komputer sekarang operasi paling cepat adalah $10^8 operasi$ per detik maka untuk nilai n = 0x01fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffa51868783bf2f966b7fcc0148f709a5d03bb5c9b8899c47aebb6fb71e91386409 dibutuhkan sebanyak 6.86479766013061e+148 detik atau 2.176813058133755e+141 tahun untuk membruteforce nilai kunci rahasia pada algoritma ini.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Selama pengembangan aplikasi, keamanan file .env sangat penting karena file tersebut seringkali berisi informasi sensitif terkait konfigurasi aplikasi. Kebocoran informasi sensitif dalam file .env dapat mengakibatkan masalah keamanan serius pada aplikasi dan data terkait.

Untuk meningkatkan keamanan file .env, salah satu solusi yang dapat digunakan adalah menggunakan algoritma ElGamal pada kurva eliptik. Dengan menerapkan algoritma ini, data dalam file .env dapat dienkripsi dan hanya dapat di-dekripsi oleh pihak yang memiliki kunci privat yang sesuai. Dengan demikian, hanya pihak yang diotorisasi yang dapat mengakses informasi sensitif dalam file tersebut.

Penerapan algoritma ElGamal pada kurva eliptik memberikan lapisan perlindungan tambahan terhadap kemungkinan serangan dan kebocoran informasi.

B. Saran

Saran yang dapat ditingkatkan ialah perluasan pengimpelmentasian algoritma ini. Karena algoritma ini sudah berfungsi untuk seluruh jenis file maka proses enkripsi dan dekripsi file dengan tipe *data in rest* masih dapat diperluas lagi dan tidak terbatas pada .env saja.

TAUTAN KODE PADA GITHUB

Tautan kode dari implementasi program dapat diakses pada link berikut ini. https://github.com/IloveNooodles/secure-env

UCAPAN TERIMA KASIH

Ucapan terima kasih penulis nyatakan kepada Tuhan Yang Maha Esa, karena karunia-Nya penulis bisa diberikan kesempatan untuk menyelesaikan dan bisa memberikan kontribusi nyata dalam memberikan ide yang dituliskan pada makalah ini. Penulis juga mengucapkan terima kasih kepada Dr. Rinaldi Munir atas dedikasinya dalam memberikan ilmu pengetahuan terkait mata kuliah kriptografi kepada penulis

REFERENSI

[1] Munir, Rinaldi. 2023. Slide Kuliah IF4020 Kriptografi: Kriptografi Kunci-Publik diakses pada tanggal 20 Mei 2023

[2] Munir, Rinaldi. 2023. Slide Kuliah IF4020 Kriptografi: Algoritma ElGamal diakses pada tanggal 20 Mei 2023

[3] Munir, Rinaldi. 2023. Slide Kuliah IF4020 Kriptografi: Elliptic Curve Cryptography diakses pada tanggal 20 Mei 2023

[4] https://rtullydo.github.io/cryptography-notes/section-ecdlp.html Diakses pada tanggal 20 Mei 2023

[5] P-521 | Standard curve database (neuromancer.sk) Diakses pada tanggal 20 Mei 2023

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.

Yogyakarta, 20 Mei 2023

Muhammad Garebaldhie ER Rahman
13520029