Implementasi RSA dan Shamir's Secret Sharing pada Web Group Chat System untuk Mengirim Pesan dan File Rahasia

Fitrah Ramadhani Nugroho - 13520030
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520030@std.stei.itb.ac.id

***Abstract*—Penggunaan aplikasi *chat* untuk mengirim pesan dan *file* merupakan hal yang mudah. Namun, seorang penyadap dapat dengan mudah membaca pesan dan *file* yang dikirim oleh pengguna. Oleh karena itu dibutuhkan kriptografi untuk menagamankan pengiriman pesan dan *file*. RSA merupakan salah satu metode untuk mengenkripsi pesan. Selain itu, digunakan juga *Shamir's secret sharing scheme* yang merupakan salah satu skema algoritma yang mengenkripsi suatu pesan kepada beberapa pihak. Makalah ini mencoba untuk membuat sistem *Webchat* dengan menggunakan Shamir's *secret sharing scheme* dan RSA.**

Keywords—webchat; pesan; file;* RSA*; Shamir's secret sharing scheme;

I. PENDAHULUAN

Kriptografi adalah sebuah ilmu dan seni dalam mengembangkan dan menggunakan algoritma untuk menjaga dan melindungi informasi. Kata *Cryptography* berasal dari Bahasa Yunani yaitu "cryptós" (*secret*) dan gráphein(*writing*). Tujuan dari penggunaan kriptografi adalah melindungi dan mengaburkan informasi sehingga hanya dapat diakses oleh pihak yang dituju. Dengan demikian, kriptografi melindungi informasi sehingga pihak yang tidak tidak dapat mengaksesnya.

Kriptografi memiliki empat prinsip utama yaitu:

a. Kerahasiaan pesan (*Confidentiality*)
b. Keaslian pesan (*Data integrity*)
c. Keaslian pengirim dan penerima pesan (*Authentication*)
d. Anti penyangkalan (*Nonrepudiation*)

Teknik yang umum digunakan pada kriptografi dibagi menjadi dua yaitu enkripsi dan dekripsi pada pesan yang akan dikirimkan. Enkripsi adalah penyempbunyian informasi menjadi *cipher* (kode rahasia) ketika pesan akan dikirim. Dekripsi adalah proses pengembalian bentuk pesan dari *cipher* menjadi pesan semula. Umumnya kedua proses diatas memerlukan kunci rahasa yang telah disepakati oleh kedua pihak.

Kriptografi berdasarkan pengembangannya terbagi menjadi dua jenis yaitu kriptografi klasik dan kriptografi modern. Kriptografi klasik umumnya dibuat dan digunakan sebelum zaman komputer sedangkan, kriptografi modern dibuat dan digunakan pada zaman komputer modern hingga saat ini. Kriptografi modern terbagi lagi menjadi dua bagian yaitu kriptografi simetris dan kriptografi asimetris. Kriptografi simetris yaitu proses kriptografi dimana proses enkripsi dan dekripsi menggunakan kunci yang sama. Contoh dari kriptografi simetris adalah AES (Advanced Encryption Standard) dan DES (Data Encryption Standard). Pada kriptografi asimetrik, proses dekripsi dan enkripsi menggunakan dua buah kunci yang berbeda yaitu kunci publik dan kunci privat. Pengirim akan mengenkripsi pesan menggunakan kunci publik. Selanjutya, penerima akan mendekrip pesan tersebut dengan kunci privat. Salah satu contoh dari kriptografi asimetrik adalah RSA (Rives-Shamir-Adleman).

Selain pembagian kunci menjadi kunci publik dan privat, terdapat metode pembagian kunci yang membagikan kunci lebih dari dua orang. Metode ini juga membutuhkan sejumlah orang untuk membangkitkan kembali kunci tersebut. Metode ini dinamakkan metode *secret sharing scheme* yang diperkenalkan oleh salah satu pencipta algoritma kunci public RSA yaitu Adi Shamir. Metode ini memungkinkan kunci untuk dekripsi dipecah menjadi beberapa bagian yang disebarkan oleh berbagai pihak. Rekonstruksi kunci ini pula memerlukan bagian-bagian dari beberapa pihak yang telah ditentukan.

Dalama makalah ini metode *Shamir's secret sharing* akan diterapkan dalam aplikasi web pengiriman pesan yang memiliki fitur *group chat*. Metode ini dapat diterapkan sehingga pesan atau *file* rahasia hanya dapat diakses ketika semua pihak di dalam *group chat* menyetujui untuk membuka pesan tersebut. Selain itu, RSA juga akan digunakan untuk mengenkripsi pesan yang dikirim sehingga tidak dapat diakses oleh pihak yang tidak diinginkan.

II. LANDASAN TEORI

A. *Shamir's Secret Sharing Scheme*

Skema pembagian rahasia Shamir adalah sebuah algoritma kriptografi yang diciptakan oleh Adi Shamir. Tujuan dari

algoritma ini adalah untuk membagi rahasia (*secret*) yang dienkripsi dan telah dibagikan menjadi beberapa bagian unik sejumlah *n* bagia. Hal ini dilakukan untuk mencegah hanya satu pihak yang bisa mengakses rahasia tanpa sepengetahuan pihak lainnya. Oleh karena itu, algoritma ini memerlukan *threshold* atau batas berupa *k* dengan nilai $0 < k \leq n$ sebagai banyaknya bagian yang diperlukan untuk mengkonstruksi ulang *secret* yang telah dibagi.

Shamir's Secret Sharing memanfaatkan ide pembagian *secret* dari persoalan interpolasi. Persamaan linear yaitu

$$y = \alpha_1 + \alpha_2 x$$

Dibutuhkan dua buah titik (x1, y1), (x2, y2) dalam menyusun persamaan linear diatas sedangkan dalam persamaan kuadratik yaitu

$$y = \alpha_1 + \alpha_2 x + \alpha_3 x^2$$

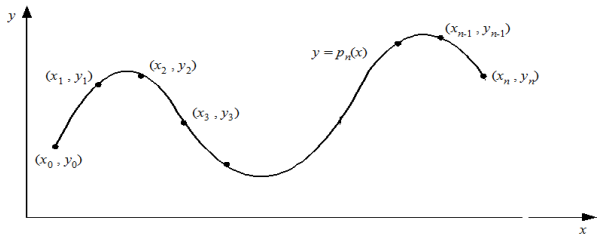
Persamaan kuadratik diatas akan membutuhkan tiga titik (x1, y1), (x2, y2), (x3, y3) untuk menyusunnya. Sehingga untuk persamaan sebesar polynomial *n* yaitu

$$y = \alpha_1 + \alpha_2 x + \cdots + \alpha_n x^{n-1}$$

Dalam persamaan derajat n diatas diperlukan minimal n+1 buah titik untuk melakukan rekonstruksi, yaitu titik (x1, y1), (x2, y2), …, (xn+1, yn+1).

Penalaran interpolasi diatas dapat disimpulkan bahwa polinom interpolasi derajat *n* yang melakukan intepolasi titik-titik adalah:

$$y = p_n(x) = a_0 + a_1 x + a_2 x_2 + \ldots + a_n x^n$$



Gambar 2.1 Ilustrasi Grafik Interpolasi Polinom
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/36-Skema-Pembagian-Data-Rahasia-2023.pdf

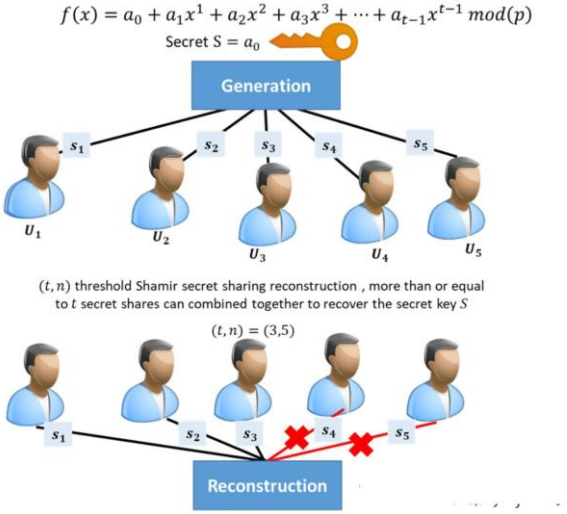
Berdasarkan ilustrasi grafik diatas, untuk mendapatkan persamaan polynomial yang sempurna diperlukan substitusi titik-titik (x0, y0), (x1, y1), ..., (xn, yn) ke dalam $y = p_n(x)$. Hasil dari substitusi ini diperoleh *n* buah sistem persamaan lanjar dalam a0, a1, a2, …, an.

$$a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0$$
$$a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y_1$$
$$a_0 + a_1 x_2 + a_2 x_2^2 + \ldots + a_n x_2^n = y_2$$
$$\ldots \qquad \ldots$$
$$a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n$$

Hasil substitusi ini sesuai dengan gambar diatas menghasilkan *n+1* buah persamaan lanjar yang perlu diselesaikan untuk mendapatkan nilai-nilai a0, a1, a2, …, an yang membentuk persamaan polinom.

Shamir's Secret Sharing menggunakan skema ambang (*treshold scheme*) dalam pembagian *secret*. Skema ambang adalah metode pembagian *secret S* kepada *n* partisipan sedemikian sehingga sembarang himpunan bagian yang terdiri *t* partisipan dapat merekonstruksi *S*, tetapi *S* tidak dapat direkonstruksi jika partisipan kurang dari *t*. Algoritma pembagian menggunakan skema ambang sebagai berikut:

1. Pilih bilangan prima *p*, yang harus lebih besar dari semua kemungkinan nilai secret *S* dan juga lebih besar dari jumlah *n* partisipan. Semua komputasi dilakukan dalam modulus *p*.
2. Pilih Pilih t – 1 buah bilangan bulat acak dalam modulus p, misalkan a1, a2, …, at – 1 dan nyatakan polinomial:
$$f(x) \equiv S + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} \pmod p$$
sedemikian sehingga f (0) ≡ S (mod p).
3. Untuk setiap *n* partisipan dipilih integer berbeda, x1, x2, …, xn (mod p) sehingga setiap orang memperoleh *share* (xi, yi) yang dalam hal ini
$$y_i \equiv f(x_i) \pmod p$$
Misalnya, untuk n orang dipilih x1 = 1, x2 = 2, …, xn = n.



Gambar 2.3 Ilustrasi Pembagian Share pada Skema Ambang
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/36-Skema-Pembagian-Data-Rahasia-2023.pdf

B. RSA (Rivest-Shamir-Adleman)

RSA merupakan salah satu algoritma kunci-publik yang paling terkenal dan paling banyak aplikasinya. RSA diciptakan dari tiga peneliti MIT (Massachussets Institute of Technology), yaitu Ronald Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1976. RSA memanfaatkan dari sulitnya memfaktorkan bilang bulat besar yang menjadi faktor-faktor prima untuk memastikan keamanannya. Semakin besar nilai kunci yang digunakan maka akan semakin besar pula nilai faktornya. Dengan demikian semakin besar nilai kunci enkripsi yang digunakan, semakin kuat pula keamanan enkripsi tersebut. Nilai kunci RSA biasanya sepanjang 1024 atau 2048 bit.

Terdapat tujuh properti yang Menyusun algoritma RSA yaitu:

1. *p* dan *q* yang merupakan bilangan prima (rahasia)
2. $n = p \cdot q$ (tidak rahasia)
3. $\Phi(n) = (p - 1)(q - 1)$ (rahasia)
4. *e* (kunci enkripsi) dengan syarat: PBB(e,Φ(n)) = 1, PBB (pembagi bersama terbesar) (tidak rahasia)
5. *d* (kunci dekripsi),

   d dihitung dari d≡e-1 mod (Φ (n)) (rahasia)
6. *m* (teks asli atau plainteks) (rahasia)
7. *c* (teks cipher atau cipherteks) (tidak rahasia)

Dari tujuh property diatas maka proses pembangkitan kunci public dan privat sebagai berikut:

1. Pertama pilih dua bilangan prima *p* dan *q* secara acak dengan $p \neq q$ kemudian hitung nilai $n = pq$.
2. Selanjutnya, hitung nilai dari $\Phi(n) = (p - 1)(q - 1)$
3. Pilih sebuah bilangan bulat *e* sebagai kunci publik dengan syarat, *e* harus relatif prima terhadap Φ(n).
4. Hitung kunci privat, d, dengan persamaan $ed \equiv 1 \pmod{\phi(n)}$ atau $d \equiv e^{-1} \pmod{(\phi(n))}$
5. Dari hasil algoritma diatas akan didapat kunci publik dengan pasangan (*e*, *n*) dan kunci privat dengan pasangan (*d*, *n*).

Proses enkripsi pesan adalah dengan mengecek ukuran pesan tersebut. Jika pesan berukuran besar maka nyatakan pesan menjadi blok-blok *plainteks* yang lebih kecil seperti *m1, m2, m3, dst.* Pastikan ukuran blok *plainteks* lebih kecil dari *n-1*. Kemudian hitung cipherteks c_i untuk *plainteks* m_i menggunakan kunci public *e* dengan persamaan

$$c_i = m_i^e \bmod n$$

Proses dekripsi pesan adalah kebalikan dari proses enkripsi. Jika blok *cipherteks* terbagi menjadi *c1, c2, c3,* dst., maka hitung kembali blok *plainteks* m_i dari blok *cipherteks* c_i menggunakan kunci privat *d* dengan persamaan

$$m_i = c_i^d \bmod n,$$

III. Implementasi

Bab III membahas cara implementasi RSA dan *Shamir's Secret Sharing Scheme* dalam *webchat*. Implementasi program dilakukan dengan menggunakan bahasa pemrograman Javascript. *Webchat* yang digunakan memiliki sistem *server* dan *client*. *Server* berfungsi untuk menerima pesan dari *client* dan mengirim pesan tersebut ke semua *client* sedangkan *client* berfungsi sebagai *user interface* untuk mengirim pesan dan menerima pesan dari *server*. *Webchat* ini menggunakan *framework* Express.js untuk *server* program dan React.js untuk *client* program. Dalam implementasi *prototype*, digunakan *webchat* yang diambil dari *repository* Github sebagai berikut:

https://github.com/yusufsefasezer/ysChat

Penggunaan *webchat* dari *repository* diatas ditujukan untuk mempercepat implementasi RSA dan *Shamir's Secret Sharing Scheme* pada *Webchat*. Proses penggunaan RSA dan *Shamir's Secret Sharing Scheme* dalam pengiriman pesan dan *file* adalah sebagai berikut:

1. *Client* melakukan *log in* ke *server* dengan menghasilkan *public* dan *private key* RSA. *Client* mengirim *private key* RSA ke *server* untuk dekripsi dan menyimpan *public key* RSA untuk enkripsi.
2. *Server* menerima *private key* dari *client* dan menyimpannya
3. Saat *client* akan mengirim pesan atau *file* rahasia, *client* menghasilkan *public* dan *private* key RSA baru untuk keperluan enkripsi dan dekripsi pesan rahasia.
4. Untuk membedakan antara *key* RSA saat *login* dan *key* RSA saat mengirim pesan rahasia, maka *private* dan *public key* RSA yang dihasilkan saat *login* akan disebut sebagai *private* dan *public key login* sedangkan *private* dan *public key* RSA yang dihasilkan saat pengiriman pesan akan disebut *private* dan *public key* pesan.
5. Kemudian *client* mengenkripsi pesan rahasia menggunakan *public key* pesan lalu mengenkripsi *private key* pesan dengan *public key log in*.
6. *Client* mengirim pesan yang dienkripsi beserta *private key* pesan yang telah dienkripsi juga ke *server*.
7. *Server* menerima pesan dan *private key* pesan yang dienkripsi lalu mendekripsi *private key* pesan tersebut dengan *private key login.*
8. *Private key* pesan yang telah didekripsi ini menjadi *secret* yang dibagi menjadi *shares* sejumlah partisipan *client* dengan *Shamir's Secret Sharing Scheme.*
9. *Shares* akan dibagikan ke setiap klien oleh *server* sehingga setiap klien akan memiliki satu *share*. *Server* juga mengirim pesan yang dienkripsi ke semua klien.
10. Klien perlu meminta *shares* ke partisipan klien lainnya sesuai dengan jumlah minimum untuk dapat merekontruksi *secret*.
11. Klien yang telah memiliki *shares* dengan jumlah mnimum dapat merekontruksi *secret* yang berupa *private key* pesan
12. Pesan yang dienkripsi dapat didekripsi dengan *private key* yang dimiliki oleh klien

Berikut ini adalah gambar kode *javascript* untuk menghasilkan *public* dan *private key* RSA. Dalam gambar 3.1 terlihat bahwa variabel *p* dan *q* mencari bilangan prima secara acak sepanjang 128 bit. Selanjutnya, program mendapatkan nilai *n* dengan mengalikan nilai *p* dan *q* beserta nilai dari Φ(n). Program mencari nilai *e* sebagai kunci public dengan *e* relative prima dengan Φ(n). Terakhir nilai *d* didapatkan dari perhitungan

$$ed \equiv 1 \pmod{\phi(n)} \text{ atau } d \equiv e^{-1} \pmod{(\phi(n))}$$

```
const generateRSA = ()=>{
    //Menghasilkan public dan private key RSA
    let p = bigintCryptoUtils.primeSync(128);
    let q = bigintCryptoUtils.primeSync(128);
    let n = p*q;

    let tot = (p-1n) * (q-1n)
    let e = bigintCryptoUtils.primeSync(128);
    while (bigintCryptoUtils.gcd(e,tot) !==1n){
        e = bigintCryptoUtils.primeSync(128);
    }
    let d = powMod(e,-1n,tot);
    let publicKey = {
        e : e.toString(),
        n : n.toString()
    };
    let privateKey = {
        d : d.toString(),
        n : n.toString()
    };
    return {publicKey, privateKey};
}
```

Gambar 3.1 Gambar kode menghasilkan *private* dan *public key* RSA

Perhitungan enkripsi dan dekripsi RSA mengikuti perhitungan yang telah dibahas di subab B RSA dari bab II yang dapat dilihat pada gambar 3.2 dan 3.3. Nilai *plaintext* yang diterima akan diubah dulu menjadi angka agar *ciphertext* dapat dihitung. Begitu pula nilai *plaintext* hasil dekripsi akan diubah menjadi *string* untuk mendapatkan pesan asli.

```
const encryptRSA = (plaintext,publicKeyE,publicKeyN)=>{
    let ciphertext = powMod(convertMSGtoBigInt(plaintext),BigInt(publicKeyE),BigInt(publicKeyN));
    return ciphertext.toString();
}
```

Gambar 3.2 Gambar kode enkripsi RSA

```
const decryptRSA = (ciphertext,privateKey)=>{
    let plaintext = powMod(BigInt(ciphertext),BigInt(privateKey.d),BigInt(privateKey.n));
    return convertBigInttoMSG(plaintext);
}
```

Gambar 3.3 Gambar kode dekripsi RSA

Implementasi *Shamir's Secret Sharing Scheme* mengikuti skema yang telah dijelaskan di bab II. Berikut ini adalah kode untuk menghasilkan *shares* dari *secret*

```
const generateShares = (secret, participant, minimum )=>{
    //Menghasilkan pemecahan pada secret menjadi
    //sejumlah participant
    //Nilai minimum digunakan untuk menghasilkan
    //koefisien beserta partisipan minimum untuk
    //rekonstruksi secret
    let coeff = generateCoeff(minimum-1n)
    let p = bigintCryptoUtils.primeSync(512)
    while (p < secret){
        p = bigintCryptoUtils.primeSync(512);
    }
    let shares = [];
    for(let i=0n;i<participant;i++){
        let y = secret;
        for(let j=0n;j<coeff.length;j++){
            y += coeff[j]*((i+1n)**(j+1n));
        }
        shares.push({
            t: i+1n,
            n: y,
        });
    }
    return {shares,p};
}
```

Gambar 3.4 Gambar kode pembagian *shares*

Gambar 3.4 merupakan blok kode untuk melakukan partisi *secret* ke dalam sejumlah *shares* yang telah ditentukan.

```
const generateCoeff = (t)=>{
    //Melakukan generate list koefisien untuk polinom
    let coeff=[]
    for(let i=0;i<t-1n;i++){
        coeff.push(bigintCryptoUtils.randBetween(1000000n,1n));
    }
    return coeff
}
```

Gambar 3.4 Gambar kode penghasilan koefisien

Blok kode pada gambar 3.4 adalah blok kode untuk menghasilkan koefisien sesuai dengan nilai *t* dengan konstanta adalah nilai acak dari 1 hingga 1000000.

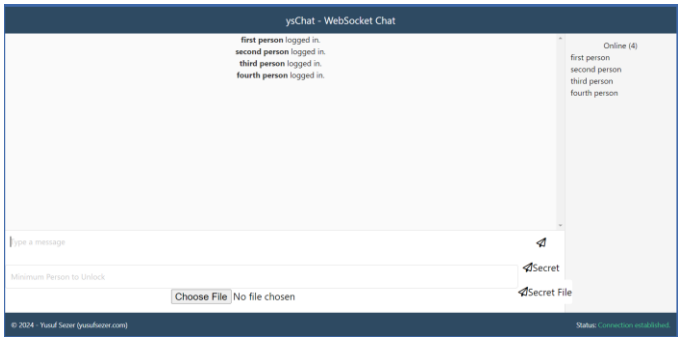
```
const constructSecret = (shares,p) =>{
    //Merekonstrusi secret berdasarkan shares dan nilai p
    let secret=0n;
    for(let i=0n;i<shares.length;i++){
        let numerator =1n;
        let denominator = 1n;
        let negative= 1n;
        for(let j=0n;j<shares.length;j++){
            if(j!==i){
                numerator *= (0n-shares[j].t);
                denominator *= (shares[i].t-shares[j].t);

            }
        }
        if(denominator<0n) negative=-1n;
        secret += ((shares[i].n*numerator*negative) % p *
        powMod(bigintCryptoUtils.abs(denominator),-1n,p))%p

    }
    return secret % p;
}
```

Gambar 3.5 Gambar kode rekonstruksi *secret*

Blok kode pada gambar 3.5 adalah blok kode untuk merekonstruksi ulang *secret* sesuai dengan *shares* yang dikumpulkan. Funsi ini akan mengembalikan *secret* yang benar jika potongan *secret* memang benar nilainya dan jumlah *shares* sudah memenuhi batas ambang yang ditentukan. Fungsi ini menggunakan interpolasi *lagrange* untuk mendapatkan nilai *secret*.

IV. Hasil Pengujian dan Analisis

Pengujian makalah ini berupa pengiriman pesan dan *file* rahasia dari salah satu partisipan. Berikut ini adalah tampilan dari aplikasi *webchat* yang digunakan:

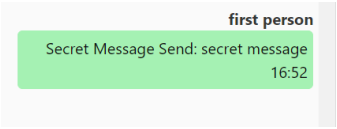


Gambar 4.1 Gambar tampilan aplikasi *webchat*

Dari gambar 4.1 terlihat terdapat tiga bagian dari tampilan *webchat*. Bagian pertama adalah isi pesan dari semua partisipan dan notifikasi untuk partisipan yang baru saja bergabung maupun keluar. Bagian kedua di sebelah kanan adalah daftar nama partisipan yang bergabung di *webchat* tersebut. Jika batas ambang tidak ditentukan oleh pengguna maka program secara otomatis mengisi batas ambang sebanyak semua partisipan dikurangi satu. Hal ini berarti semua partisipan kecuali pengirim pesan harus menyertakan *shares* mereka dan dimasukkan ke dalam *shares.* Bagian ketiga di bagian bawah merupakan bagian untuk mengirim pesan, menentukan batas ambang minimum untuk merekonstruksi kunci privat, dan mengunggah *file* untuk dikirim. Di bagian

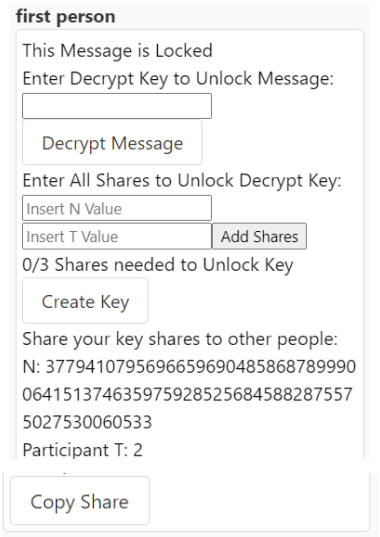
tersebut terdapat tiga mode pengiriman yaitu pengiriman pesan biasa dengan logo pesawat kertas, pengiriman pesan rahasia dengan logo pesawat kertas serta tulisan “secret” dan pengiriman *file* rahasia dengan tulisan “secret file”.

Selanjutnya untuk pengujian ditunjukkan tampilan *webchat* ketika salah satu pengguna mengirim pesan rahasia.



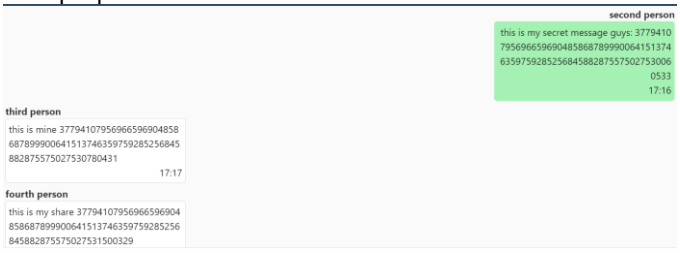
Gambar 4.2 Gambar partisipan mengirim pesan rahasia

Pada gambar 4.2 partisipan dengan nama “first person” mengirim pesan rahasia dengan isi “secret message” ke semua partisipan. Partisipan yang menerima pesan tersebut akan memiliki tampilan sesuai gambar 4.3

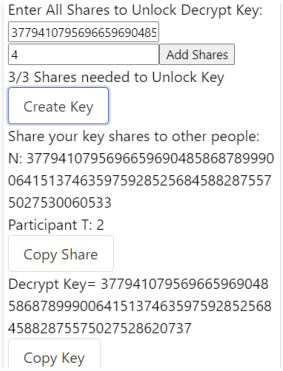


Gambar 4.3 Gambar partisipan penerima pesan rahasia

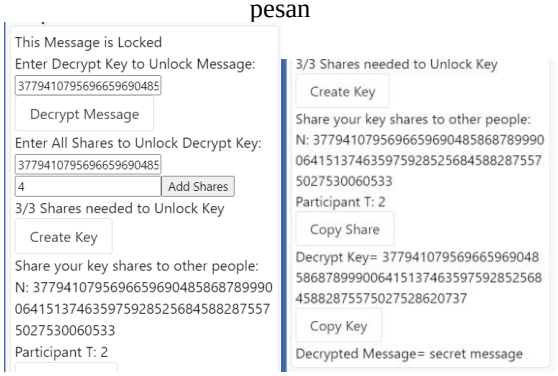
Gambar 4.3 menunjukkan pesan yang diterima oleh partisipan selain pengirim pesan rahasia. Untuk membuka pesan asli yang dikirimkan, partisipan harus memasukkan semua *shares* dengan cara meminta partisipan lainnya untuk mengirim *share* milik mereka. Setelah partisipan memasukkan semua *shares* mereka dan *shares* dari partisipan lainnya, partisipan mengklik tombol “Create Key” untuk mendapatkan *private key* yang nantinya akan dimasukkan ke *input* paling atas. Kemudian partisipan mengklik tombol *decrypt message* utuk membuka pesan rahasia. Berikut ini adalah proses dekripsi pesan rahasia



Gambar 4.4 Gambar partisipan mengirim shares masing-masing



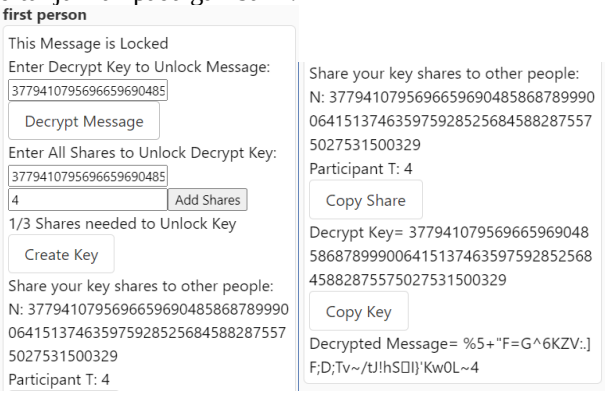
Gambar 4.5 Gambar partisipan yang membuka *private key* pesan



Gambar 4.6 Gambar partisipan yang membuka pesan rahasia

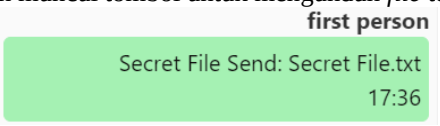
Sesuai penjelasan sebelumnya, pada gambar 4.4 partisipan perlu membagi *share* mereka ke partisipan lainnya. Kemudian pada gambar 4.5 menunjukkan seorang partisipan yang berhasil memasukkan semua *shares* yang diperlukan dan membuka *private key* untuk mendekrip pesan rahasia. Terakhir pada gambar 4.6 pada bagian akhir pesan akan terlihat pesan rahasia yang berhasil di dekripsi dengan isi pesan sesuai dengan pesan rahasia yang dikirim pada gambar 4.2.

Jika *shares* yang dimasukkan tidak sesuai atau belum memenuhi batas ambang yang ditentukan maka *private key* akan tidak sesuai dengan *private key* aslinya. Jika *private key* ini digunakan untuk mendekrip pesan maka akan keluar pesan yang tidak sesuai dengan pesan aslinya seperti yang ditunjukkan pada gambar 4.7

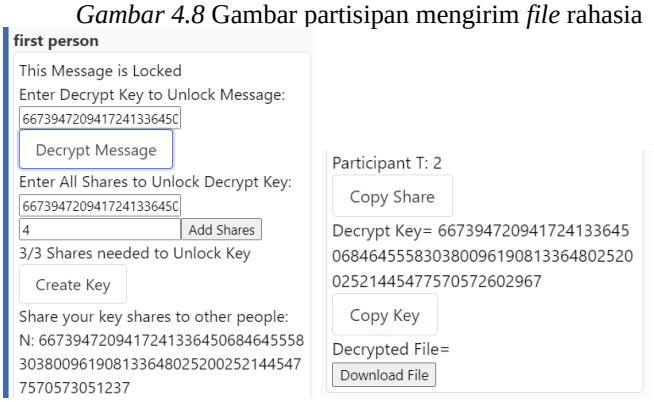


Gambar 4.7 Gambar partisipan yang salah membuka pesan rahasia

Pengiriman *file* rahasia memiliki proses yang sama dengan pengiriman pesan rahasia. Perbedaannya adalah partisipan yang mengunggah *file* rahasia memiliki pemberitahuan berupa "Secret File Send" ditambah nama *file* tersebut. Perbedaan lainnnya partisipan lainnya yang berhasil mendekrip *file* rahasia akan muncul tombol untuk mengunduh *file* tersebut.



Gambar 4.8 Gambar partisipan mengirim *file* rahasia



Gambar 4.9 Gambar partisipan mendekrip *file* rahasia

Gambar 4.10 Gambar partisipan mendekrip *file* rahasia

Gambar 4.9 menunjukkan partisipan yang berhasil mendekrip *file* rahasia sedangkan gambar 4.10 menunjukkan partisipan yang mengunduh *file* rahasia yang telah didekripsi. Dari penjelasan sebelumnya beserta gambar yang ditunjukkan dari gambar 4.1 hingga 4.10 menunjukkan bahwa RSA dan *Shamir's Secret Sharing* berhasil digunakan untuk mengirim pesan dan *file* rahasia. Penggunaan *Shamir's Secret Sharing* menunjukkan perlunya kolaborasi dari anggota *webchat* yang terlibat untuk dapat membuka pesan dan *file* rahasia yang dikirim. Hal ini mencegah hanya salah satu partisipan yang mengetahui pesan rahasia tersebut. Penggunaan RSA memastikan pesan rahasia tidak dapat dibaca oleh pihak penyerang.

PRANALA GITHUB

https://github.com/fitrahrn/WebChat-with-RSA-and-Shamir-s-Secret-Sharing-Scheme

V. KESIMPULAN DAN SARAN

Pengiriman pesan membutuhkan keamanan dan privasi untuk memastikan pesan tidak dapat dibaca oleh pihak yang tidak diinginkan. Salah satu cara untuk mengamankan pesan adalah dengan menggunakan kriptografi untuk mengubah pesan menjadi *ciphertext*. RSA sebagai salah satu algoritma kriptografi yang paling banyak digunakan untuk komunikasi di internet dapat pula digunakan untuk mengenkripsi pesan yang dikirim. Selain itu, algoritma *shamir's secret sharing* dan konsep dari *secret sharing scheme* sangat berguna untuk mengamankan *file* dan pesan yang dikirim serta untuk memastikan semua partisipan terlibat untuk melakukan rekonstruksi ulang pesan *secret*. Percobaan di bab sebelumnya menunjukkan bahwa RSA dan *Shamir's Secret Sharing* berhasil di implementasi dalam *webchat*. Percobaan sebelumnya juga menunjukkan bahwa memang jika jumlah partisipan kurang dari jumlah yang dibutuhkan maka terjadi kegagalan dalam rekonstruksi *private key* RSA dan pesan rahasia yang dikirim.

Implementasi yang dilakukan dalam makalah ini belum sempurna sehingga dapat dikembangkan dalam makalah selanjutnya. Salah satu pengembangan yang dapat dilakukan adalah memperbaiki *user interface webchat* untuk pengiriman pesan dan file rahasia serta untuk melakukan dekripsi kunci dan pesan rahasia. Pengembangan lainnya yang bisa dilakukan adalah memperbaiki panjang pesan dan besar *file* yang dapat dikirim karena saat ini program hanya bisa mengirim pesan pendek dan *file* dengan ukuran kecil.

VI. UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa, karena atas kasih, Rahmat, dan karuna-Nya sehingga penulis dapat menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir selaku dosen pengampu mata kuliah terkait atas bimbingan dan pengajaran yang telah diberikan beliau. Terakhir, penulis juga berterima kasih kepada keluarga dan teman-teman penulis yang telah memberikan dukungan selama proses belajar mata kuliah IF4020 Kriptografi dan pengerjaan makalah ini.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/35-Skema-Pembagian-Data-Rahasia-2024.pdf [Accessed: 10-June-2024].

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/18-Algoritma-RSA-2024.pdf [Accessed: 8-June-2024].

[3] https://www.ibm.com/topics/cryptography [Accessed: 8-June-2024].

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

Fitrah Ramadhani Nugroho