SKRIPSI

PEMBANGUNAN PERANGKAT LUNAK YANG MENGIMPLEMENTASIKAN METODE SECRET SHARING UNTUK BERBAGI PASSWORD

Abraham Sri Paskah Ageng Wahono

NPM: 2012730072

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN

2017

UNDERGRADUATE THESIS

SOFTWARE IMPLEMENTATION OF SECRET SHARING METHODS FOR DISTRIBUTING PASSWORD

Abraham Sri Paskah Ageng Wahono

NPM: 2012730072

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2017

ABSTRAK

Password adalah salah satu teknik otentikasi entitas yang umum digunakan untuk memperoleh hak akses pada suatu sistem. Teknik otentikasi menggunakan *password* masih memiliki kelemahan dari segi keamanan. Permasalahan yang sering kali ditemui adalah hilangnya *password* atau rusaknya kerahasiaan *password* karena adanya serangan dari pihak yang tidak bertanggung jawab. *Password* bisa saja dibuat salinannya dan disimpan dibeberapa tempat, namun cara ini tidak aman karena berisiko menurunkan tingkat kerahasiaan *password* sehingga diperlukan cara untuk membagikan *password* tanpa merusak kerahasiaan dari *password*.

Metode *secret sharing* dapat digunakan untuk membagikan pesan rahasia kepada sejumlah partisipan tanpa memberikan informasi mengenai pesan rahasia tersebut. Pada skema *threshold secret sharing* (k, n), pesan rahasia akan dibagikan ke sejumlah n partisipan. Setiap partisipan akan memperoleh bagian dari pesan rahasia yang berupa *share*. Pesan rahasia hanya dapat diperoleh kembali dengan menggabungkan minimal k buah *share*.

Pada skripsi ini akan dibangun perangkat lunak yang dapat mengimplementasikan penggunaan metode *secret sharing* untuk membagikan *password*. Skripsi ini akan menggunakan dua buah metode *secret sharing* yaitu metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Berdasarkan hasil pengujian yang dilakukan, dapat disimpulkan bahwa perangkat lunak yang dibangun dapat mengimplementasikan penggunaan metode *secret sharing* Shamir dan metode *secret sharing* Blakley untuk membagikan *password*.

Kata-kata kunci: *Password*, Otentikasi Entitas, *Secret Sharing*, metode *Secret Sharing* Shamir, metode *secret sharing* Blakley

ABSTRACT

Password is one of the most commonly used entity authentication technique to prove identity or access approval to gain access to a system. The usage of password as an authentication technique somehow still have several flaws. Some problems that may occur are lost/forgotten password or on several ocassions, malicious attack from intruders that can corrupt the anonymous password. Copies of password can be created and stored on various places, but by doing this the secrecy of password can be harmed.

Secret sharing methods can be used to distribute a secret message to several participants while maintaining the secrecy of said message by not giving any information about the secret message to any participants. By using (k, n) threshold secret sharing scheme, secret message can be distributed amongst group of n participants. Every participants will receive part of the secret message called "share" and secret message can be reconstructed only when a sufficient k number of participants come together and combine their part of the share.

In this undergraduate thesis, a software is developed to implement the usage of secret sharing methods for distributing password. There are two different secret sharing methods that are used in this undergraduate thesis, those methods are Shamir's secret sharing and Blakley's secret sharing. Based on the tests done, it can be concluded that the software built in this undergraduate thesis can implement password distribution using Shamir's secret sharing and Blakley's secret sharing.

Keywords: Password, Entity Authentication, Secret Sharing, Shamir's Secret Sharing, Blakley's Secret Sharing

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

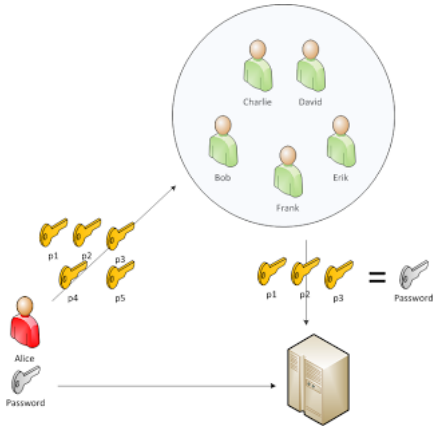
1.1 Latar Belakang

Otentikasi adalah suatu proses verifikasi untuk menentukan keaslian identitas dari pihak yang ingin mengakses sumber daya atau informasi yang terdapat pada sebuah sistem sehingga dapat ditentukan apakah seseorang berhak atau tidak untuk mengakses sumber daya atau informasi pada sistem tersebut [1]. Proses otentikasi memiliki dua jenis yaitu otentikasi pesan dan otentikasi entitas. Perbedaan utama dari kedua proses otentikasi tersebut yaitu pada proses otentikasi pesan, proses otentikasi tidak perlu dilakukan secara *real time* yang artinya pesan yang dikirim dapat diotentikasi kapan saja setelah pesan tersebut diterima. Sementara pada proses otentikasi entitas, proses verifikasi hanya dapat berjalan apabila pihak yang ingin diotentikasi berkomunikasi secara langsung dengan pihak yang mengotentikasi.

Salah satu teknik otentikasi entitas yang umum digunakan adalah *password*. *Password* adalah kode rahasia atau kata sandi yang diketahui oleh entitas (dalam hal ini dapat berupa orang atau proses), yang merupakan kunci untuk bisa mengakses atau membuka suatu sistem yang dikunci [2]. *Password* dapat berupa kombinasi dari karakter alfabet, angka, dan simbol. Teknik otentikasi menggunakan *password* memiliki beberapa kelemahan dari segi keamanan. Permasalahan yang sering kali ditemui adalah hilangnya *password* atau rusaknya kerahasiaan *password* karena adanya serangan dari pihak yang tidak bertanggung jawab. *Password* bisa saja dibuat salinannya dan disimpan di beberapa tempat, namun cara ini tidak aman karena berisiko menurunkan tingkat kerahasiaan *password* [2]. Salinan *password* dapat mengalami kebocoran dan dapat disalahgunakan oleh pihak yang tidak berwenang. Selain itu, untuk suatu sistem yang di dalamnya terdapat informasi yang bersifat sangat penting diperlukan proses otentikasi yang tingkat keamanannya lebih tinggi. Proses otentikasi yang hanya membutuhkan satu entitas tidak cukup aman untuk mengamankan sistem tersebut. Diperlukan suatu cara untuk membagikan *password* ke beberapa entitas tanpa menurunkan tingkat kerahasiaan *password*.

Cara yang dapat digunakan untuk mengatasi permasalahan tersebut adalah dengan menggunakan metode *secret sharing*. *Secret sharing* merupakan metode untuk merahasiakan pesan dengan cara membagikan pesan rahasia kepada beberapa partisipan di mana setiap partisipan akan memperoleh bagian dari pesan rahasia yang disebut dengan *share* [3]. Setiap partisipan akan mendapatkan *share* yang berbeda-beda dan partisipan-partisipan tersebut sama sekali tidak memiliki informasi mengenai pesan rahasia yang dibagikan. Pada skema *threshold secret sharing* (k, n), pesan rahasia S akan dibagikan ke n buah partisipan dan pesan rahasia hanya dapat diperoleh kembali dengan mengumpulkan k buah *share* atau lebih [4]. Metode ini dapat meningkatkan tingkat keamanan

sistem yang proses otentikasinya menggunakan *password* karena dengan membagikan pesan rahasia ke beberapa partisipan, proses otentikasi yang dibutuhkan untuk memperoleh akses ke suatu sistem menjadi berlapis-lapis. Ilustrasi skema *threshold secret sharing* dapat dilihat pada Gambar 1.1.



Gambar 1.1: Skema *threshold secret sharing* $(3, 5)$

Terdapat beberapa metode *threshold secret sharing* (k, n) yang dapat digunakan untuk membagikan pesan rahasia. Pada skripsi ini akan dibahas dua buah metode *threshold secret sharing* (k, n) yang dapat digunakan untuk membagikan pesan rahasia. Metode-metode tersebut adalah metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Pada bab selanjutnya akan dibahas secara lebih dalam mengenai perbedaan antara kedua metode *threshold secret sharing* (k, n) tersebut beserta cara kerja masing-masing metode *secret sharing* Shamir dan *secret sharing* Blakley.

Pada skripsi ini akan dibangun perangkat lunak yang dapat mengimplementasikan metode-metode *secret sharing* yang digunakan untuk membagikan *password*. Perangkat lunak yang dibangun akan mengimplementasikan dua buah metode *secret sharing* yaitu metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Perangkat lunak yang mengimplementasikan metode-metode tersebut akan diuji dengan berbagai kasus. Pengujian yang dilakukan akan menggunakan teknik pengujian fungsional dan pengujian eksperimental. Dari hasil pengujian tersebut akan ditarik kesimpulan apakah perangkat lunak yang dibangun dapat mengimplementasikan metode *secret sharing* Shamir dan metode *secret sharing* Blakley dengan tepat.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas, rumusan masalah pada skripsi ini adalah :

1. Bagaimana cara kerja metode *secret sharing* Shamir?
2. Bagaimana cara kerja metode *secret sharing* Blakley?
3. Bagaimana cara mengimplementasikan penggunaan skema *secret sharing* Shamir dan *secret sharing* Blakley untuk membagikan *password*.

1.3 Tujuan

Untuk menjawab rumusan masalah di atas, maka tujuan yang ingin dicapai dari skripsi ini adalah :

1. Mempelajari cara kerja metode *secret sharing* Shamir.
2. Mempelajari cara kerja metode *secret sharing* Blakley.
3. Membangun perangkat lunak yang dapat mengimplementasikan pembagian *password* menggunakan skema *secret sharing* Shamir dan *secret sharing* Blakley.

1.4 Batasan Masalah

Batasan-batasan masalah untuk skripsi ini adalah sebagai berikut :

1. Masukan *password* tidak mengandung spasi antar karakter.
2. Untuk metode *secret sharing* Blakley, skema *thresold secret sharing* yang dapat digunakan adalah skema (3,3).

1.5 Metodologi

Metodologi yang digunakan dalam penyusunan skripsi ini adalah sebagai berikut :

1. Melakukan studi literatur mengenai dasar-dasar kriptografi.
2. Melakukan studi literatur mengenai *secret sharing*, seperti metode-metode *secret sharing* dan cara kerjanya, beserta cara mengimplementasikan metode-metode *secret sharing*.
3. Melakukan perancangan kelas yang akan digunakan untuk mengimplementasikan *secret sharing* Shamir dan *secret sharing* Blakley.
4. Mengimplementasikan hasil perancangan kelas ke dalam bahasa pemrograman *Java*.
5. Melakukan pengujian terhadap perangkat lunak yang telah mengimplementasikan metode *secret sharing* Shamir dan *secret sharing* Blakley.
6. Menarik kesimpulan berdasarkan hasil pengujian.

1.6 Sistematika Pembahasan

Skripsi ini disusun secara sistematis ke dalam 6 bab yang terdiri dari pendahuluan, dasar teori, analisis, perancangan, implementasi dan pengujian, dan kesimpulan. Sistematika pembahasan dari skripsi ini adalah :

1. Bab 1 Pendahuluan
   Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.
2. Bab 2 Dasar Teori
   Bab ini berisi dasar teori tentang kriptografi, otentikasi pesan, *password*, *secret sharing*, metode *secret sharing* Shamir, metode *secret sharing* Blakley, dan pengkodean ASCII.

3. Bab 3 Analisis
   Bab ini berisi analisis masalah, analisis metode *secret sharing*, dan analisis perangkat lunak yang didalamnya membahas diagram aliran proses, dan diagram kelas awal.

4. Bab 4 Perancangan
   Bab ini berisi perancangan perangkat lunak yang akan dibangun yang di dalamnya meliputi perancangan antarmuka dan diagram kelas yang lengkap.

5. Bab 5 Implementasi dan Pengujian
   Bab ini berisi implementasi antarmuka perangkat lunak, pengujian fungsional perangkat lunak yang mengimplementasikan metode *secret sharing* Shamir dan metode *secret sharing* Blakley, serta pengujian eksperimental perangkat lunak.

6. Bab 6 Kesimpulan
   Bab ini berisi kesimpulan dari awal hingga akhir skripsi dan saran untuk pengembangan selanjutnya.

BAB 2

DASAR TEORI

Pada bab ini akan dibahas teori-teori mengenai dasar kriptografi, otentikasi entitas, *password*, metode-metode *secret sharing*, dan pengkodean ASCII yang digunakan sebagai dasar teori skripsi ini.

2.1 Kriptografi

Kriptografi berasal dari bahasa Yunani "*kryptós*" yang artinya adalah rahasia dan "*graphein*" yang berarti tulisan [5]. Jadi, kriptografi berarti "tulisan rahasia". Definisi kriptografi yang umum dipakai di masa lalu adalah : ilmu dan seni untuk menjaga kerahasiaan pesan. Namun, kriptografi modern saat ini membahas lebih dari sekedar kerahasiaan saja. Pada saat ini kriptografi merupakan ilmu yang mempelajari teknik matematis yang bertujuan memberikan layanan (aspek-aspek) keamanan [5]. Aspek-aspek keamanan dari kriptografi tersebut adalah sebagai berikut :

1. Kerahasiaan (*confidentiality*) : layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak.

2. Integritas data (*data integrity*) : layanan yang menjamin bahwa pesan masih asli/utuh atau belum pernah dimanipulasi selama pengiriman.

3. Otentikasi (*authentication*) : layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*entity authentication*) maupun mengidentifikasi kebenaran sumber pesan (*data origin authentication*).

4. Nirpenyangkalan (*non-repudiation*) : layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan.

Pada kriptografi, aspek keamanan yang berkaitan dengan kerahasiaan (*confidentiality*) dapat dicapai dengan melakukan dua proses dasar yaitu proses enkripsi dan proses dekripsi. Proses enkripsi adalah proses mengubah/menyandikan pesan asli yang disebut dengan plainteks menjadi bentuk lain yang tidak dapat dipahami dengan menggunakan algoritma kriptografi. Pesan yang sudah tersandi tersebut disebut juga sebagai cipherteks. Proses dekripsi adalah proses mengubah kembali cipherteks menjadi plainteks. Bila proses enkripsi dinotasikan sebagai E, proses dekripsi sebagai D, plainteks sebagai P, dan cipherteks sebagai C, maka proses enkripsi dan dekripsi dapat

dinyatakan secara matematis seperti pada Persamaan 2.1 dan Persamaan 2.2 secara berturut-turut.

$$E(P) = C \tag{2.1}$$

$$D(C) = P \tag{2.2}$$

Pada awalnya, tingkat keamanan algoritma kriptografi ditentukan oleh tingkat kerahasiaan algoritmanya. Namun cara tersebut tidak selalu aman karena jika algoritmanya diketahui maka pesan rahasia mudah untuk dipecahkan. Masalah ini diatasi dengan menggunakan kunci sebagai parameter untuk transformasi proses enkripsi dan proses dekripsi. Dengan menggunakan kunci, algoritma tidak perlu lagi dijaga kerahasiaannya namun kunci harus tetap dirahasiakan. Kunci biasanya berupa *string* (deretan huruf) atau deretan bilangan. Dengan menggunakan kunci K, maka proses enkripsi dan proses dekripsi dapat dinyatakan secara matematis seperti pada Persamaan 2.3 dan Persamaan 2.4 secara berturut-turut.

$$E_K(P) = C \tag{2.3}$$

$$D_K(C) = P \tag{2.4}$$

Sehingga memenuhi Persamaan 2.5 seperti berikut :

$$D_K(E_K(P)) = P \tag{2.5}$$

2.2 Otentikasi Entitas

Dalam kriptografi, otentikasi entitas adalah teknik otentikasi yang dirancang untuk memungkinkan suatu pihak membuktikan kebenaran identitas dari pihak lain. Entitas dalam hal ini bisa berupa manusia, proses, *client*, atau *server*. Entitas yang identitasnya ingin dibuktikan kebenarannya disebut sebagai *claimant*, dan pihak yang mencoba untuk membuktikan kebenaran identitas dari *claimant* disebut sebagai *verifier* [2]. Terdapat beberapa perbedaan antara otentikasi pesan (*message authentication*) dengan otentikasi entitas (*entity authentication*). Perbedaan kedua teknik otentikasi tersebut adalah :

- Otentikasi entitas berjalan secara *real time* sementara otentikasi pesan tidak.
- Otentikasi pesan hanya dapat melakukan otentikasi terhadap satu pesan saja; proses otentikasi perlu diulang untuk setiap pesan baru. Sementara pada otentikasi entitas hanya dilakukan satu kali otentikasi dalam satu sesi pertukaran informasi.

Dalam otentikasi entitas, pihak *claimant* harus mengidentifikasikan dirinya kepada pihak *verifier*. Pihak *claimant* perlu memberikan suatu bukti kepada pihak *verifier* untuk membuktikan kebenaran identitasnya. Bukti tersebut dapat berupa sesuatu yang diketahui (*something known*) oleh *claimant*, sesuatu yang dimiliki (*something possessed*) oleh *claimant*, atau sesuatu yang melekat (*something inherent*) pada *claimant*.

1. Sesuatu yang diketahui (*something known*) : adalah rahasia yang hanya diketahui oleh pihak *claimant*, yang dapat dibuktikan kebenarannya oleh pihak *verifier*. Contohnya adalah *password*

dan PIN.

2. Sesuatu yang dimiliki (*something possessed*) : adalah sesuatu yang dapat membuktikan kebenaran identitas dari *claimant*. Contohnya adalah paspor, SIM, KTP, dan kartu kredit.

3. Sesuatu yang melekat (*something inherent*) : adalah sesuatu yang merupakan ciri khas dari *claimant*. Contohnya adalah tanda tangan, sidik jari, retina mata, dan bentuk wajah.

Otentikasi menggunakan *password* adalah salah satu metode otentikasi entitas yang paling sederhana dan paling lama digunakan. *Password* digunakan ketika pengguna (*user*) ingin mengakses suatu sistem untuk menggunakan sumber daya atau mengakses informasi yang tersedia pada sistem tersebut. Otentikasi menggunakan *password* dibagi menjadi dua kelompok [2] yaitu :

1. *Fixed password* : adalah *password* tetap yang dapat digunakan berkali-kali untuk setiap kali akses.

2. *One-time password* : adalah *password* yang hanya berlaku untuk satu kali akses.

2.3 Secret Sharing

Secret sharing adalah metode untuk merahasiakan pesan dengan cara membagikan pesan rahasia ke sejumlah partisipan. Skema *secret sharing* dibuat untuk mengatasi masalah yang dimiliki oleh teknik otentikasi menggunakan kunci. Teknik otentikasi menggunakan *password* merupakan salah satu teknik otentikasi yang umum digunakan untuk mengamankan suatu sistem yang di dalamnya terdapat informasi yang bersifat rahasia. Dalam hal ini, *password* digunakan sebagai kunci rahasia untuk memperoleh akses ke suatu sistem yang diamankan. Namun teknik otentikasi menggunakan *password* masih memiliki banyak kekurangan. Kunci rahasia (*password*) rentan mengalami kerusakan atau hilang apabila kunci hanya disimpan di satu tempat. Namun kunci rahasia juga rawan untuk diperoleh oleh pihak yang tidak bertanggung jawab apabila kunci tersebut diduplikasi dan disimpan di beberapa tempat. Apabila kunci mengalami kerusakan atau hilang, proses otentikasi akan mengalami hambatan karena harus membuat kunci baru atau merombak sistem yang diamankan [4].

Dengan menggunakan skema *secret sharing*, tingkat keamanan sistem akan semakin terjaga dengan baik karena kunci rahasia dipecah dan dibagikan ke sejumlah partisipan di mana setiap partisipan tidak memiliki informasi apapun mengenai pesan rahasia tersebut selain informasi mengenai bagian dari kunci rahasia yang dibagikan.

Pada skema *threshold secret sharing* (k, n) pesan rahasia dibagikan ke n banyak partisipan. Setiap partisipan memperoleh bagian dari pesan rahasia yang disebut dengan *share* [3], dan *share* yang diperoleh setiap partisipan akan berbeda dengan partisipan-partisipan lainnya. Pesan rahasia hanya dapat dibangun kembali (direkonstruksi) dengan mengumpulkan k buah share atau lebih. Nilai k dan n harus memenuhi $1 \leq k \leq n$ [4].

Pada Subbab 2.4 dan Subbab 2.5 akan dibahas dua buah metode *threshold secret sharing* yaitu metode *secret sharing* Shamir dan metode *secret sharing* Blakley.

2.4 Metode Secret Sharing Shamir

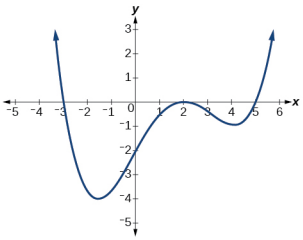
Pada metode *secret sharing* Shamir, pesan rahasia terletak pada suatu fungsi polinomial yang unik dengan derajat $(k-1)$. Pesan rahasia terletak pada fungsi $f(0)$ dan fungsi polinomial tersebut dapat diperoleh dengan memperoleh k buah titik pada polinomial. Ilustrasi grafik fungsi polinomial dapat dilihat pada Gambar 2.1. Pada metode ini, pesan rahasia S dibagikan ke n buah partisipan dengan menggunakan polinomial seperti pada rumus 2.6 di bawah ini :

$$f(x) = S + \sum_{i=1}^{k-1} a_i \cdot x^i \tag{2.6}$$

di mana x menunjukkan partisipan ke-x dan a adalah konstanta yang dapat ditentukan secara acak. Dengan menggunakan rumus di atas, fungsi polinomial $f(x)$ yang diperoleh setiap partisipan akan berbeda-beda. Setiap partisipan kemudian akan memperoleh *share* S_i seperti berikut :

$$S_i = (i, f(i) \ mod \ p) \tag{2.7}$$

di mana i menunjukkan partisipan ke-i dan p adalah bilangan prima acak yang nilainya lebih besar dari pesan rahasia. Pada metode *secret sharing* Shamir ini, bilangan prima p ditentukan secara acak dan digunakan untuk meningkatkan keamanan pesan rahasia agar tidak rawan terhadap serangan *brute force* [1]. *Share* yang diperoleh setiap partisipan ke-i direpresentasikan sebagai titik (x_i, y_i) pada fungsi polinomial.



Gambar 2.1: Contoh grafik fungsi polinomial berderajat 4 dengan nilai $f(0) = -2$

Metode *secret sharing* Shamir menggunakan rumus interpolasi untuk memperoleh kembali pesan rahasia yang terletak pada fungsi polinomial. Interpolasi yang dilakukan akan menghasilkan fungsi polinomial yang melewati titik yang menyimpan pesan rahasia. Metode ini menggunakan rumus interpolasi Lagrange untuk memperoleh kembali fungsi polinomial unik yang mengandung pesan rahasia tersebut [6]. Interpolasi Lagrange dinyatakan secara matematis pada rumus 2.8 berikut :

$$f(x) = \sum_{i=1}^{k} y_i \left(\prod_{j=1, j\neq i}^{k} \frac{x_j}{x_j - x_i} \right) \tag{2.8}$$

di mana x_i dan x_j adalah letak *share* ke-i dan ke-j pada sumbu x, dan y_i adalah letak *share* ke-i pada sumbu y. Sehingga fungsi polinomial $f(0)$ yang menyimpan pesan rahasia diperoleh dengan melakukan operasi modulo fungsi $f(x)$ dengan bilangan prima p yang sudah ditentukan sebelumnya.

Pesan rahasia diperoleh dengan menggunakan rumus 2.9 seperti berikut :

$$f(0) = \sum_{i=1}^{k} y_i \left(\prod_{j=1, j\neq i}^{k} \frac{x_j}{x_j - x_i} \right) \ mod\ p \tag{2.9}$$

2.5 Metode Secret Sharing Blakley

Metode *secret sharing* Blakley menggunakan pendekatan geometri untuk membagikan pesan rahasia kepada n buah partisipan [6]. Pada metode *secret sharing* Blakley ini, pesan rahasia terletak pada suatu titik di bidang k dimensi di mana k buah *hyperplane* beririsan. *Hyperplane* adalah bidang $k-1$ dimensi yang terletak di dalam bidang utama k dimensi yang merepresentasikan *share* yang dimiliki setiap partisipan.

Gambar 2.2: Representasi bidang 3 dimensi dengan *hyperplane* 2 dimensi

Pada skema *secret sharing* Blakley $(3, n)$, pesan rahasia dinotasikan dengan fungsi $Q(x, y, z)$ yang merepresentasikan koordinat letak pesan rahasia di bidang k dimensi. *Hyperplane* untuk setiap partisipan pada *secret sharing* Blakley skema $(3, n)$ direpresentasikan dalam notasi matematika seperti berikut :

$$z \ \equiv \ a_i x + b_i y + c_i \tag{2.10}$$

di mana x, y, dan z adalah pesan rahasia yang ingin dibagikan, a_i dan b_i adalah nilai yang ditentukan secara acak untuk masing-masing partisipan ke-i. Nilai c_i untuk masing-masing partisipan ke-i diperoleh dengan menggunakan persamaan 2.11 berikut :

$$c_i \ \equiv \ (z - a_i x - b_i y) \ mod\ p \tag{2.11}$$

Sama dengan metode *secret sharing* Shamir, pada metode *secret sharing* Blakley ini bilangan prima p digunakan untuk meningkatkan keamanan pesan rahasia. Dengan menggunakan kedua persamaan yang telah disebutkan di atas, masing-masing partisipan akan memperoleh *share* yang berupa bidang 2 dimensi. Pesan rahasia dapat dibangun kembali dengan menggunakan persamaan 2.12 di bawah ini :

$$a_i x + b_i y - z \equiv -c_i \ (mod\ p) \tag{2.12}$$

Persamaan tersebut kemudian direpresentasikan ke dalam bentuk matriks seperti berikut :

$$\begin{pmatrix} a_1 & b_1 & -1 \\ a_2 & b_2 & -1 \\ a_3 & b_3 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -c_1 \\ -c_2 \\ -c_3 \end{pmatrix} \ mod\ p \tag{2.13}$$

Nilai x, y, dan z akan diperoleh dengan melakukan operasi baris elementer pada matriks di atas [7]. Masing-masing nilai x, y, dan z akan di-*mod* dengan bilangan prima p untuk memperoleh kembali pesan rahasia.

2.6 ASCII

Pada skripsi ini perangkat lunak yang dibangun akan mengimplementasikan metode-metode *secret sharing* yang sudah dibahas pada subbab sebelumnya. Untuk melakukan perhitungan pada metode-metode *secret sharing*, dibutuhkan cara untuk mengubah *password* yang berupa karakter ke bilangan desimal. Dalam komputasi terdapat beberapa model pengkodean karakter yang dapat digunakan untuk mengkodekan masukan *password* ke dalam bentuk desimal.

Perangkat lunak yang dibangun akan menggunakan pengkodean ASCII untuk mengkodekan karakter dan simbol ke nilai desimal. Skripsi ini menggunakan pengkodean ASCII karena teknik pengkodean ini paling umum digunakan dan didukung oleh bahasa pemrograman *Java* [8]. Tabel ASCII dapat dilihat pada Tabel 2.1.

Tabel 2.1: Tabel ASCII

Karakter	**Nilai desimal**
!	33
"	34
#	35
$	36
%	37
&	38
‘	39
(	40
)	41
*	42
+	43
,	44
-	45
.	46
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
	Dilanjutkan ke halaman berikutnya

Tabel 2.1 – Lanjutan dari halaman sebelumnya

Karakter	**Nilai desimal**
8	56
9	57
:	58
;	59
<	60
=	61
>	62
?	63
@	64
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
[	91
\	92
]	93
^	94

Dilanjutkan ke halaman berikutnya

Tabel 2.1 – Lanjutan dari halaman sebelumnya

Karakter	**Nilai desimal**
_	95
’	96
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122
{	123
|	124
{	125
~	126

Pada skripsi ini pengkodean ASCII yang digunakan hanya karakter yang nilai desimalnya antara 33 sampai 126. Hal ini disebabkan karakter-karakter yang nilai desimalnya bukan di antara 33 sampai 126 merupakan karakter yang tidak dapat di-*print*/ditampilkan sehingga karakter-karakter tersebut tidak dapat menjadi masukan *password*.

BAB 3

ANALISIS

Bab ini akan membahas analisis masalah, analisis metode *secret sharing* yang di dalamnya mencakup pembahasan metode *secret sharing* Shamir dan metode *secret sharing* Blakley, dan analisis perangkat lunak yang mencakup pembahasan diagram aliran proses dan diagram kelas awal.

3.1 Analisis Masalah

Password merupakan salah satu teknik otentikasi entitas yang umum digunakan untuk mengamankan suatu sistem yang memiliki informasi rahasia. Pada umumnya suatu sistem yang diamankan akan melakukan proses otentikasi menggunakan satu entitas untuk menentukan keaslian identitas atau hak akses untuk mengakses sistem tersebut. Dalam hal ini artinya proses otentikasi hanya dilakukan terhadap satu orang saja. Sistem yang diamankan menggunakan proses otentikasi satu entitas masih memiliki beberapa kelemahan. *Password* dapat saja rusak kerahasiaannya apabila *password* tersebut hilang, dan apabila *password* dibagikan kepada orang lain atau disimpan salinannya, *password* tersebut dapat disalahgunakan. Suatu sistem yang memiliki informasi penting dapat ditingkatkan keamanannya dengan menggunakan otentikasi lebih dari satu entitas. Dengan menggunakan otentikasi lebih dari satu entitas, proses verifikasi yang dilakukan akan berlapis-lapis sehingga dapat meningkatkan keamanan pada suatu sistem.

Metode *secret sharing* dapat digunakan untuk membagikan *password* ke sejumlah partisipan tanpa membocorkan informasi mengenai *password* tersebut. Dengan menggunakan metode *secret sharing*, *password* dapat dibagikan tanpa mengubah teknis otentikasi pada sistem. Pada metode *secret sharing*, partisipan akan memperoleh bagian dari *password* yang disebut dengan *share*. *Share* yang diperoleh oleh partisipan dapat digabungkan sehingga *password* yang dirahasiakan dapat diperoleh kembali.

Pada skripsi ini digunakan dua buah metode *secret sharing* yaitu metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Kedua metode tersebut menggunakan skema *threshold secret sharing* (k, n) di mana dibutuhkan minimal k buah *share* dari n banyak partisipan yang digabungkan untuk dapat merekonstruksi *password*.

Pada analisis ini, terdapat dua proses utama yang akan dilakukan oleh metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Proses pertama adalah proses pembagian *password* di mana pada setiap metode akan dilakukan perhitungan dengan menggunakan algoritma yang berbeda untuk membagikan *password* ke n banyak partisipan. Proses kedua adalah proses penggabungan *share* di mana *password* akan direkonstruksi dengan menggabungkan minimal k buah *share*. Pada proses ini metode *secret sharing* Shamir dan metode *secret sharing* Blakley juga menggunakan

algoritma yang berbeda untuk menggabungkan *share* dan merekonstruksi *password*. Analisis metode *secret sharing* Shamir dan metode *secret sharing* Blakley akan dibahas pada Subbab 3.2.1 dan Subbab 3.2.2.

3.2 Analisis Metode Secret Sharing

Analisis metode *secret sharing* yang dilakukan akan menggunakan skema *secret sharing* $(3, 3)$ di mana dibutuhkan minimal 3 buah *share* dari 3 partisipan untuk merekonstruksi *password*. Pada analisis ini, *password* yang akan dibagikan dengan menggunakan metode *secret sharing* Shamir dan metode *secret sharing* Blakley adalah "IT".

3.2.1 Analisis Metode Secret Sharing Shamir

Pada bagian ini akan dilakukan analisis dari implementasi metode *secret sharing* Shamir dengan menggunakan *threshold* 3, banyak partisipan 3, dan *password* "IT".

1. Proses pembagian *password*

   Pada proses ini langkah pertama yang dilakukan adalah melakukan perhitungan untuk membagikan *password* dengan menggunakan rumus 2.6. Untuk melakukan perhitungan tersebut *password* yang berupa karakter perlu diubah menjadi bilangan desimal dengan menggunakan format pengkodean ASCII.

   Dalam kode ASCII yang dapat dilihat pada Tabel 2.1, nilai desimal dari karakter "I" adalah 73 dan nilai desimal dari karakter "T" adalah 84. Nilai desimal dari karakter "I" dan "T" kemudian akan disatukan menjadi satu bilangan dengan menempel setiap nilai desimal dari karakter *password*. Sehingga representasi *password* dalam bilangan desimal adalah 73084.

   Bilangan tersebut diperoleh dengan menyatukan bilangan 73 dan 84. Karena pada tabel ASCII nilai desimal dari suatu karakter dapat berupa bilangan desimal dengan 2 digit dan 3 digit, maka pada skripsi ini karakter dengan nilai desimal 2 digit akan diubah menjadi 3 digit dengan menambah angka 0 di depan nilai desimal.

   Dengan menggunakan rumus 2.6, akan dibentuk fungsi $f(x)$ untuk setiap partisipan ke-x seperti berikut :

   $$f(1) = 73084 + (115115.1) + (115116.1^2) = 73084 + 115115 + 115116 = 303315$$
   $$f(2) = 73084 + (115115.2) + (115116.2^2) = 73084 + 230230 + 460464 = 763778$$
   $$f(3) = 73084 + (115115.3) + (115116.3^2) = 73084 + 345345 + 1036044 = 1454473$$

   Sehingga pada akhir proses ini, setiap partisipan ke-i akan memperoleh *share* seperti berikut :

   $$\textit{Share } 1 = (1, 303315 \ mod \ 189437) = (1, 113878)$$
   $$\textit{Share } 2 = (2, 763778 \ mod \ 189437) = (2, 6030)$$

Share 3 = (3, 1454473 *mod* 189437) = (3, 128414)

2. Proses penggabungan *share*

Pada metode ini, *password* dapat direkonstruksi dengan menggunakan rumus 2.9. Perhitungan yang dilakukan untuk merekonstruksi *password*/pesan rahasia S adalah seperti berikut :

$$
\begin{aligned}
S &= \left(113878\left(\tfrac{2}{2-1}\cdot\tfrac{3}{3-1}\right) + 6030\left(\tfrac{1}{1-2}\cdot\tfrac{3}{3-2}\right) + 128414\left(\tfrac{1}{1-3}\cdot\tfrac{2}{2-3}\right)\right) \ mod\ 189437 \\
&= \left(113878\tfrac{6}{2} + 6030\tfrac{3}{-1} + 128414\tfrac{2}{2}\right) \ mod\ 189437 \\
&= (341634 - 18090 + 128414) \ mod\ 189437 \\
&= 451958 \ mod\ 189437 \\
&= 73084
\end{aligned}
$$

Pesan rahasia S yang diperoleh dengan menggunakan rumus di atas kemudian diubah kembali ke bentuk karakter dengan menggunakan pengkodean ASCII. Sehingga dengan mengambil setiap 3 digit bilangan desimal akan diperoleh bilangan 73 dan 084. Bilangan tersebut kemudian diubah dengan menggunakan kode ASCII sehingga diperoleh 73 = "I" dan 084 = "T".

3.2.2 Analisis Metode Secret Sharing Blakley

Pada bagian ini akan dilakukan analisis dari implementasi metode *secret sharing* Blakley dengan menggunakan *threshold* 3, banyak partisipan 3, dan *password* "IT".

1. Proses pembagian *password*

Proses pembagian *password* pada metode *secret sharing* Blakley, *password* yang masih berupa karakter perlu diubah menjadi bilangan desimal. *Password* yang masih berupa karakter diubah ke bilangan desimal dengan menggunakan kode ASCII. Hasil bilangan desimal yang diperoleh dengan menggunakan tabel ASCII adalah 73 untuk karakter "I" dan 84 untuk karakter "T". Sehingga bilangan desimal yang merepresentasikan *password* adalah 73084.

Bilangan tersebut diperoleh dengan menyatukan bilangan 73 dan 84. Karena pada tabel ASCII nilai desimal dari suatu karakter dapat berupa bilangan desimal dengan 2 digit dan 3 digit, maka pada skripsi ini karakter dengan nilai desimal 2 digit akan diubah menjadi 3 digit dengan menambah angka 0 di depan nilai desimal.

Seperti yang sudah dibahas pada Subbab 2.5, pesan rahasia direpresentasikan dengan variabel x, y, dan z. Pada skripsi ini, nilai desimal dari *password* akan dipecah menjadi 3 ke dalam variabel x, y, dan z. Sehingga nilai x, y, dan z yang merepresentasikan pesan rahasia adalah sebagai berikut :

$$x = 7$$
$$y = 30$$
$$z = 84$$

Selanjutnya setiap partisipan ke-i akan memperoleh *share* berupa a_i, b_i, dan c_i. Pada skripsi ini nilai dari variabel a_i dan b_i untuk masing-masing partisipan ke-i sudah ditentukan dengan nilai seperti berikut :

$$a_1 = 1$$
$$b_1 = 2$$
$$a_2 = 2$$
$$b_2 = 3$$
$$a_3 = 5$$
$$b_3 = 7$$

Kemudian nilai dari c_i untuk masing-masing partisipan ke-i dapat diperoleh dengan menggunakan rumus 2.11. Berikut adalah perhitungan untuk memperoleh nilai c_i untuk masing-masing partisipan :

$$c_1 = (84 - (1.7) - (2.30)) \ mod \ 197$$
$$= (84 - 7 - 60) \ mod \ 197$$
$$= 17 \ mod \ 197 = 17$$

$$c_2 = (84 - (2.7) - (3.30)) \ mod \ 197$$
$$= (84 - 14 - 90) \ mod \ 197$$
$$= (-20) \ mod \ 197 = 177$$

$$c_3 = (84 - (5.7) - (7.30)) \ mod \ 197$$
$$= (84 - 35 - 210) \ mod \ 197$$
$$= (-161) \ mod \ 197 = 36$$

Sehingga pada akhir proses ini, *share* yang diperoleh setiap partisipan adalah seperti berikut :

Share 1 :
$a_1 = 1$
$b_1 = 2$
$c_1 = 17$

Share 2 :
$a_2 = 2$
$b_2 = 3$

$c_2 = 177$

Share 3 :

$a_3 = 5$

$b_3 = 7$

$c_3 = 36$

2. Proses penggabungan *share*

Pada metode *secret sharing* Blakley, *password* dapat direkonstruksi dengan menggunakan persamaan linear yang direpresentasikan dalam bentuk matriks seperti berikut :

$$\begin{pmatrix} a_1 & b_1 & -1 \\ a_2 & b_2 & -1 \\ a_3 & b_3 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -c_1 \\ -c_2 \\ -c_3 \end{pmatrix} \ mod\ p$$

Dengan memasukkan *share* yang berupa a_i, b_i, dan c_i, dapat diperoleh nilai dari x, y, dan z yang menyimpan pesan rahasia berupa *password*.

$$\begin{pmatrix} 1 & 2 & -1 \\ 2 & 3 & -1 \\ 5 & 7 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -17 \\ -177 \\ -36 \end{pmatrix} \ mod\ 197$$

Langkah berikutnya adalah dengan melakukan operasi baris elementer pada matriks di atas untuk memperoleh matriks segitiga atas. Berikut adalah langkah-langkah operasi baris elementer yang dilakukan :

Operasi 1 :

$$R2 + (-2R1) \rightarrow R2 \quad \left(\begin{array}{ccc|c} 1 & 2 & -1 & -17 \\ 0 & -1 & 1 & -143 \\ 5 & 7 & -1 & -36 \end{array}\right)$$

Operasi 2 :

$$R3 + (-5R1) \rightarrow R3 \quad \left(\begin{array}{ccc|c} 1 & 2 & -1 & -17 \\ 0 & -1 & 1 & -143 \\ 0 & -3 & 4 & 49 \end{array}\right)$$

Operasi 3 :

$$-1R2 \rightarrow R2 \quad \left(\begin{array}{ccc|c} 1 & 2 & -1 & -17 \\ 0 & 1 & -1 & 143 \\ 0 & -3 & 4 & 49 \end{array}\right)$$

Operasi 4 :

$$R3 + 3R2 \rightarrow R3 \quad \left(\begin{array}{ccc|c} 1 & 2 & -1 & -17 \\ 0 & 1 & -1 & 143 \\ 0 & 0 & 1 & 478 \end{array}\right)$$

Dari operasi baris elementer yang sudah dilakukan diperoleh persamaan berikut :

$$x + 2y - z = -17$$
$$0 + y - z = 143$$
$$0 + 0 + z = 478$$

Maka nilai x, y, dan z adalah :
$z = 478$
$y = 143 + z = 143 + 478 = 621$
$x = -17 - 2y + z = -17 - (2.621) + 478 = -17 - 1242 + 478 = -781$

Rekonstruksi pesan rahasia :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 478 \\ 621 \\ -781 \end{pmatrix} \ mod \ 197 = \begin{pmatrix} 7 \\ 30 \\ 84 \end{pmatrix}$$

Nilai x, y, dan z yang diperoleh kemudian disatukan menjadi satu bilangan sehingga pesan rahasia yang diperoleh adalah 73084. Langkah terakhir yang dilakukan adalah mengubah kembali pesan rahasia tersebut ke dalam karakter dengan menggunakan kode ASCII. Dengan mengambil setiap 3 digit bilangan desimal akan diperoleh bilangan 73 dan 084. Bilangan tersebut kemudian diubah dengan menggunakan kode ASCII sehingga diperoleh 73 = "I" dan 084 = "T".

3.3 Analisis Perangkat Lunak

Bagian ini akan membahas mengenai analisis perangkat lunak yang mengimplementasikan penggunaan metode *secret sharing* untuk berbagi *password*. Pembahasan yang dilakukan mencakup diagram aliran proses dan diagram kelas dari perangkat lunak yang dibangun.

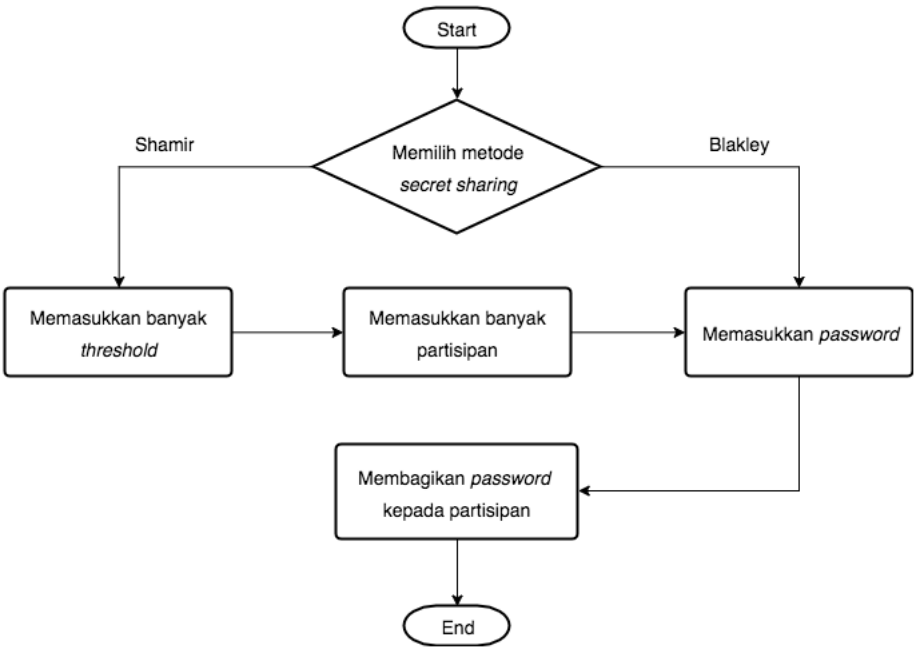
3.3.1 Diagram Aliran Proses

Perangkat lunak yang dibangun akan menjalankan dua buah proses utama yaitu proses pembagian *password* dan proses penggabungan *share*. Pada bagian ini alur dari kedua proses tersebut diilustrasikan dengan menggunakan diagram aliran proses yang dapat dilihat pada Gambar 3.1 dan Gambar 3.2.

Proses awal yang akan dijalankan oleh perangkat lunak adalah proses pembagian *password*. Pada proses ini pertama-tama pengguna akan memilih metode *secret sharing* apa yang ingin digunakan untuk membagikan *password*. Perangkat lunak yang dibangun menyediakan dua buah metode

secret sharing yang dapat digunakan yaitu metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Bila pengguna memilih metode *secret sharing* Shamir, maka pada langkah selanjutnya pengguna akan memasukkan banyak *threshold* dan banyak partisipan. Apabila metode pilihan pengguna adalah metode *secret sharing* Blakley, maka pengguna tidak perlu memasukkan *input* banyak *threshold* dan banyak partisipan. Pada langkah selanjutnya pengguna akan memasukkan *password* yang ingin dibagikan ke partisipan. Langkah terakhir pada proses ini adalah membagikan *password* ke sejumlah partisipan.

Berikut dilampirkan diagram aliran proses pembagian *password* beserta penjelasan dari setiap langkah pada proses pembagian *password* :



Gambar 3.1: Diagram aliran proses pembagian *password*

- Memilih metode *secret sharing*

  Pada tahap ini pengguna dapat memilih metode *secret sharing* yang digunakan untuk membagikan *password*. Pilihan metode yang tersedia pada perangkat lunak adalah metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Pada perangkat lunak yang dibangun, skema *secret sharing* yang dapat digunakan pada metode *secret sharing* Blakley adalah skema $(3, 3)$ di mana banyak *threshold* dan banyak partisipan adalah 3.

- Memasukkan banyak *threshold*

  Pada tahap ini pengguna memasukkan banyak *threshold*. Banyak *threshold* untuk metode *secret sharing* Shamir harus berupa bilangan bulat positif, dan untuk metode *secret sharing* Blakley banyak *threshold* yang dapat digunakan adalah 3.

- Memasukkan banyak partisipan

  Pada tahap ini pengguna menentukan banyak partisipan yang diinginkan. Banyak partisipan harus dalam bilangan bulat positif dan harus sama dengan atau lebih besar dari banyak

threshold. Untuk metode *secret sharing* Blakley banyak partisipan yang dapat digunakan adalah 3.

- Memasukkan *password*

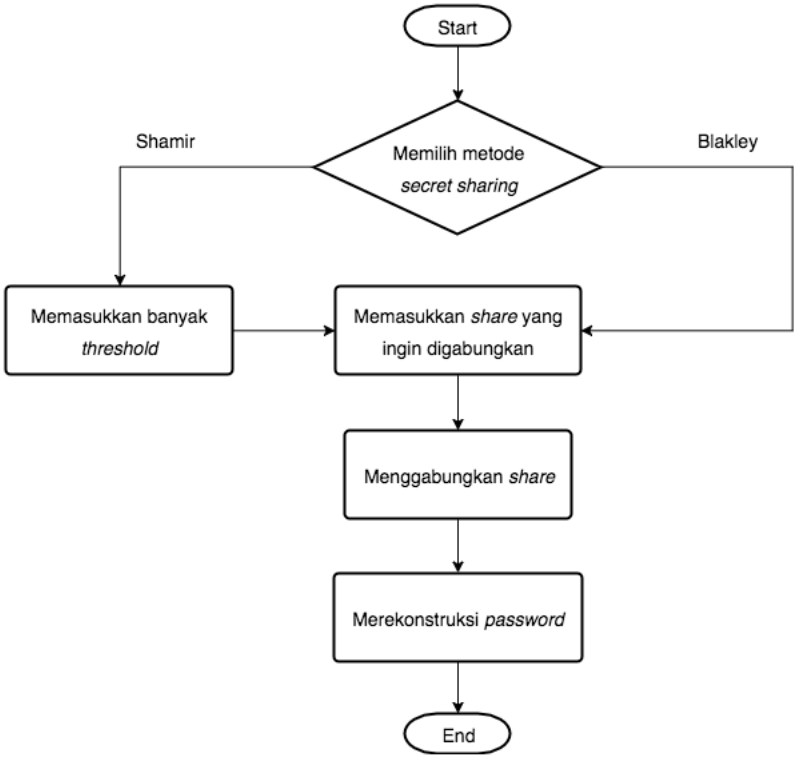
  Pada tahap ini pengguna memasukkan *password* yang ingin dibagikan kepada partisipan. *Password* yang dimasukkan dapat berupa karakter alfabet, angka, simbol atau kombinasi dari ketiganya.

- Membagikan *password* kepada partisipan

  Pada tahap ini perangkat lunak membagikan *password* kepada partisipan sesuai dengan metode *secret sharing* yang dipilih oleh pengguna. Setiap partisipan akan menerima masing-masing bagian dari *password* yang berupa *share*.

Pada proses penggabungan *share*, langkah awal yang dilakukan adalah menentukan metode *secret sharing* yang digunakan. Pada langkah berikutnya pengguna memasukkan banyak *threshold* dan memasukkan *share-share* yang ingin digabungkan. Selanjutnya perangkat lunak akan menggabungkan *share* yang telah dimasukkan oleh pengguna dan merekonstruksi kembali *password*.

Berikut dilampirkan diagram aliran proses penggabungan *share* beserta penjelasan dari setiap langkah pada proses penggabungan *share* :



Gambar 3.2: Diagram aliran proses penggabungan *password*

- Memilih metode *secret sharing*

  Pada tahap ini pengguna dapat memilih metode *secret sharing* yang digunakan untuk membagikan *password*. Pilihan metode yang tersedia pada perangkat lunak adalah metode *secret*

sharing Shamir dan metode *secret sharing* Blakley. Pada perangkat lunak yang dibangun, skema *secret sharing* yang dapat digunakan pada metode *secret sharing* Blakley adalah skema $(3, 3)$ di mana banyak *threshold* dan banyak partisipan adalah 3.

- Memasukkan banyak *threshold*

  Pada tahap ini pengguna memasukkan banyak *threshold*. Banyak *threshold* untuk metode *secret sharing* Shamir harus berupa bilangan bulat positif, dan untuk metode *secret sharing* Blakley banyak *threshold* yang dapat digunakan adalah 3.

- Memasukkan *share* yang ingin digabungkan

  Pada tahap ini pengguna memasukkan *share-share* yang ingin digabungkan. Banyak *share* yang digabungkan harus sama dengan atau lebih banyak dari banyak *threshold*.

- Menggabungkan *share*

  Pada tahap ini perangkat lunak menggabungkan *share* yang telah dimasukkan oleh pengguna pada tahap sebelumnya.

- Merekonstruksi *password*

  Pada tahap ini perangkat lunak merekonstruksi *password* berdasarkan masukan metode *secret sharing* yang telah dipilih oleh pengguna.

3.3.2 Diagram Kelas Awal

Bagian ini akan membahas mengenai diagram kelas awal. Diagram kelas awal yang dibuat merepresentasikan kelas-kelas yang akan digunakan untuk membangun perangkat lunak. Diagram kelas ini menggunakan tiga buah kelas yang dapat dilihat pada Gambar 3.3. Berikut adalah penjelasan dari kelas-kelas dan atribut-atribut yang terdapat di dalam diagram kelas awal :

1. Kelas SecretSharing

   Kelas SecretSharing adalah kelas utama perangkat lunak yang bertipe abstrak dan merupakan kelas *parent* dari kelas SSShamir dan kelas SSBlakley. Kelas SecretSharing memiliki tiga buah atribut yaitu :

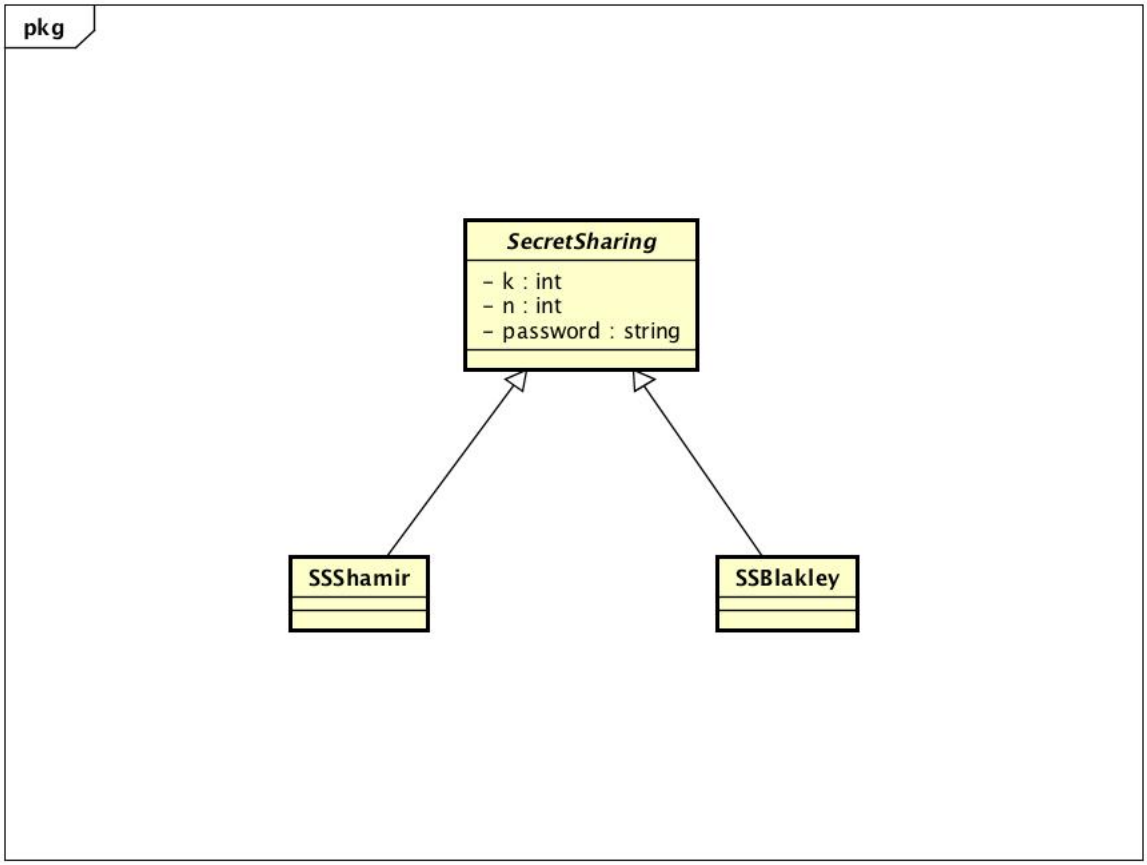
   - Atribut k : Atribut k adalah atribut dengan tipe data Integer yang merepresentasikan banyak *threshold*.
   - Atribut n : Atribut n adalah atribut bertipe data Integer yang merepresentasikan banyak partisipan.
   - Atribut password : Atribut password adalah atribut bertipe data String yang merepresentasikan *password* yang dimasukkan oleh pengguna.

2. Kelas SSShamir

   Kelas SSShamir merupakan kelas turunan/*child* dari kelas SecretSharing. Kelas ini mengimplementasikan pembagian *password* dan rekonstruksi *password* menggunakan metode *secret sharing* Shamir.

3. Kelas SSBlakley

   Kelas SSBlakley merupakan kelas turunan/*child* dari kelas SecretSharing. Kelas ini mengimplementasikan pembagian *password* dan rekonstruksi *password* menggunakan metode *secret sharing* Blakley.



Gambar 3.3: Diagram kelas awal

BAB 4

PERANCANGAN

Pada bab ini akan dibahas mengenai perancangan perangkat lunak yang akan dibangun berdasarkan hasil analisis yang telah dibahas pada bab sebelumnya. Beberapa hal yang akan dibahas di bab ini antara lain adalah kebutuhan masukan dan keluaran perangkat lunak, rancangan antarmuka perangkat lunak, dan diagram kelas rinci beserta rincian metode-metode pada kelas yang digunakan.

4.1 Kebutuhan Masukan dan Keluaran

Pada skripsi ini perangkat lunak yang mengimplementasikan penggunaan *secret sharing* untuk berbagi *password* dibangun menggunakan bahasa pemrograman *Java*. Perangkat lunak yang dibangun tersebut menggunakan metode-metode *secret sharing* seperti metode *secret sharing* Shamir dan metode *secret sharing* Blakley untuk membagikan *password* ke sejumlah partisipan. Pada perangkat lunak ini terdapat dua proses yang dijalankan yaitu proses pembagian *password* dan proses rekonstruksi *password*. Pada proses pembagian *password*, masukan yang dibutuhkan oleh perangkat lunak adalah pilihan metode *secret sharing* yang akan digunakan, banyak *threshold* dan partisipan, dan *password* yang diinginkan oleh pengguna. Berikut adalah kebutuhan masukan perangkat lunak pada proses pembagian *password* :

1. Pilihan metode *secret sharing* yang digunakan untuk membagikan *password*.
2. Banyak *threshold*.
3. Banyak partisipan.
4. *Password* yang diinginkan oleh pengguna.

Khusus untuk metode *secret sharing* Blakley, masukan perangkat lunak yang dibutuhkan hanya *password* karena implementasi metode *secret sharing* Blakley pada perangkat lunak yang dibangun menggunakan skema *secret sharing* $(3, 3)$. Pada skema *secret sharing* $(3, 3)$ banyak *threshold* dan banyak partisipan yang dapat digunakan adalah 3.

Kebutuhan keluaran pada proses pembagian *password* adalah daftar *share* yang diperoleh setiap partisipan berdasarkan metode *secret sharing* yang digunakan.

Pada proses rekonstruksi *password*, masukan yang dibutuhkan oleh perangkat lunak adalah pilihan metode *secret sharing* yang digunakan, banyak *threshold*, dan *share* yang ingin digabungkan. Berikut adalah kebutuhan masukan perangkat lunak pada proses rekonstruksi *password* :

1. Pilihan metode *secret sharing* yang digunakan.

2. Banyak *threshold*.

3. *Share* yang ingin digabungkan.

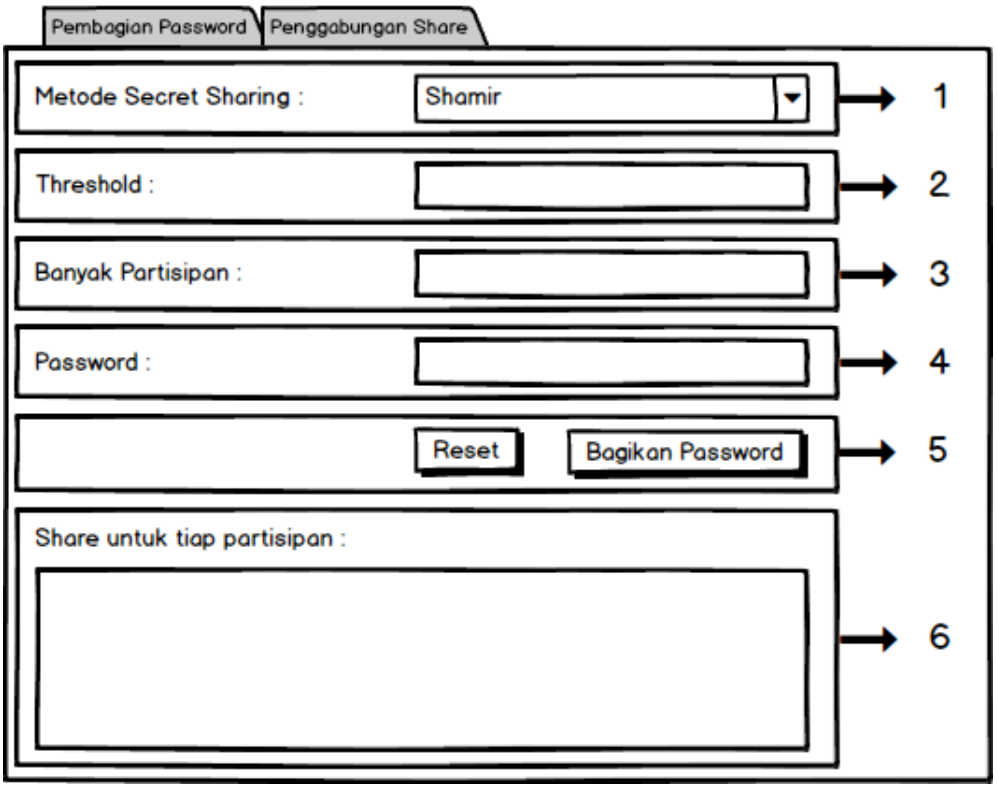
Bila metode pilihan pengguna adalah metode *secret sharing* Blakley maka masukan yang dibutuhkan perangkat lunak hanya *share* yang ingin digabungkan. Seperti yang sudah dibahas sebelumnya, hal ini disebabkan perangkat lunak yang dibangun mengimplementasikan metode *secret sharing* Blakley dengan skema $(3, 3)$.

Kebutuhan keluaran pada proses rekonstruksi *password* berupa *password* yang diperoleh kembali dari hasil penggabungan *share*.

4.2 Rancangan Antarmuka

Perangkat lunak yang mengimplementasikan penggunaan *secret sharing* untuk berbagi *password* memiliki dua buah tampilan antarmuka. Tampilan pertama adalah halaman Pembagian *Password* yang akan menampilkan antarmuka untuk proses pembagian *password*. Tampilan kedua adalah halaman Penggabungan *Share* yang akan menampilkan antarmuka untuk proses penggabungan *share*.

Rancangan antarmuka untuk halaman Pembagian *Password* dapat dilihat pada Gambar 4.1. Pada halaman ini pengguna dapat memasukkan *input* yang dibutuhkan pada proses pembagian *password*. Berikut adalah penjelasan untuk masing-masing bagian dari tampilan antarmuka pada halaman Pembagian *Password* :



Gambar 4.1: Rancangan antarmuka proses pembagian *password*

1. Bagian 1

   Pada bagian ini pengguna dapat memilih metode *secret sharing* apa yang akan digunakan untuk membagikan *password*. Pengguna dapat memilih antara dua pilihan metode yang disediakan pada *combo box* metode *secret sharing*, yaitu metode *secret sharing* Shamir dan metode *secret sharing* Blakley.

2. Bagian 2

   Di bagian ini pengguna dapat menentukan banyak *threshold* yang diinginkan. Bila pada Bagian 1 pengguna memilih metode *secret sharing* Blakley, maka secara otomatis perangkat lunak akan menetapkan banyak *threshold* menjadi 3.

3. Bagian 3

   Pada bagian ini pengguna dapat menentukan banyak partisipan yang diinginkan. Banyak partisipan akan ditetapkan menjadi 3 apabila pada Bagian 1 pengguna memilih metode *secret sharing* Blakley.

4. Bagian 4

   Pada bagian ini pengguna dapat memasukkan *password* yang ingin dibagikan ke sejumlah partisipan.

5. Bagian 5

   Di bagian ini terdapat dua tombol yaitu tombol "*Reset*" dan tombol "Bagikan *Password*". Tombol "*Reset*" berfungsi untuk mengosongkan semua *text field* dan *text area* pada antarmuka dan mengembalikan tampilan antarmuka ke tampilan awal. Bila pengguna menekan tombol "Bagikan *Password*" maka perangkat lunak akan membagikan *password* dengan menggunakan metode *secret sharing* yang sudah dipilih pada Bagian 1.

6. Bagian 6

   Pada bagian ini perangkat lunak akan menampilkan *share* yang diperoleh setiap partisipan sesuai dengan metode *secret sharing* yang dipilih oleh pengguna.

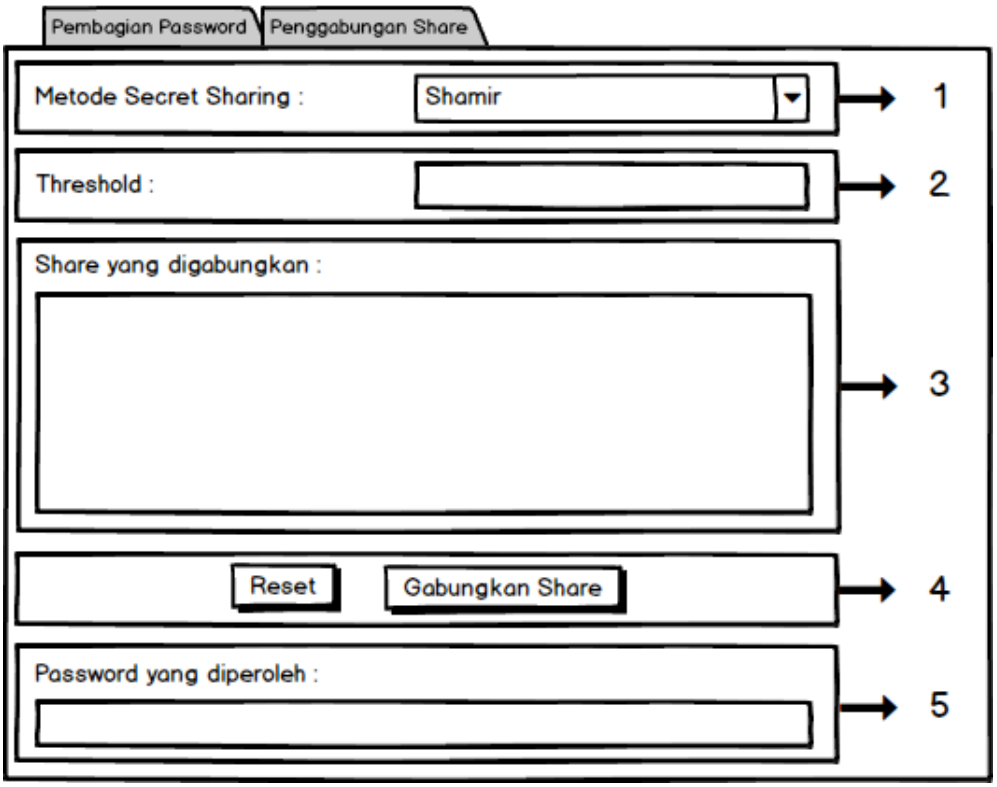
Rancangan antarmuka untuk halaman Penggabungan *Share* dapat dilihat pada Gambar 4.2. Pada halaman ini *password* akan direkonstruksi berdasarkan masukan metode *secret sharing* dan banyak *threshold*. Berikut adalah penjelasan untuk masing-masing bagian dari tampilan antarmuka pada halaman Penggabungan *Share* :

1. Bagian 1

   Pada bagian ini pengguna dapat memilih salah satu dari dua buah pilihan metode *secret sharing* yang tersedia, yaitu metode *secret sharing* Shamir dan metode *secret sharing* Blakley.

2. Bagian 2

   Di bagian ini pengguna dapat memasukkan banyak *threshold* sesuai dengan banyak *threshold* di saat proses pembagian *password*. Banyak *threshold* akan secara otomatis ditetapkan menjadi 3 apabila pada Bagian 1 metode *secret sharing* yang dipilih oleh pengguna adalah metode *secret sharing* Blakley.



Gambar 4.2: Rancangan antarmuka proses penggabungan *share*

3. Bagian 3

   Pada bagian ini pengguna dapat memasukkan *share* yang ingin digabungkan pada *text area* yang disediakan untuk merekonstruksi *password*

4. Bagian 4

   Di bagian ini terdapat dua buah tombol yaitu tombol "*Reset*" dan tombol "Gabungkan *share*". Tombol "*Reset*" berfungsi untuk mengembalikan tampilan antarmuka seperti semula dengan mengosongkan semua *text field* dan *text area* pada halaman Penggabungan *Share*. Tombol "Gabungkan *Share*" berfungsi untuk menggabungkan *share* yang telah dimasukkan pengguna pada Bagian 3.

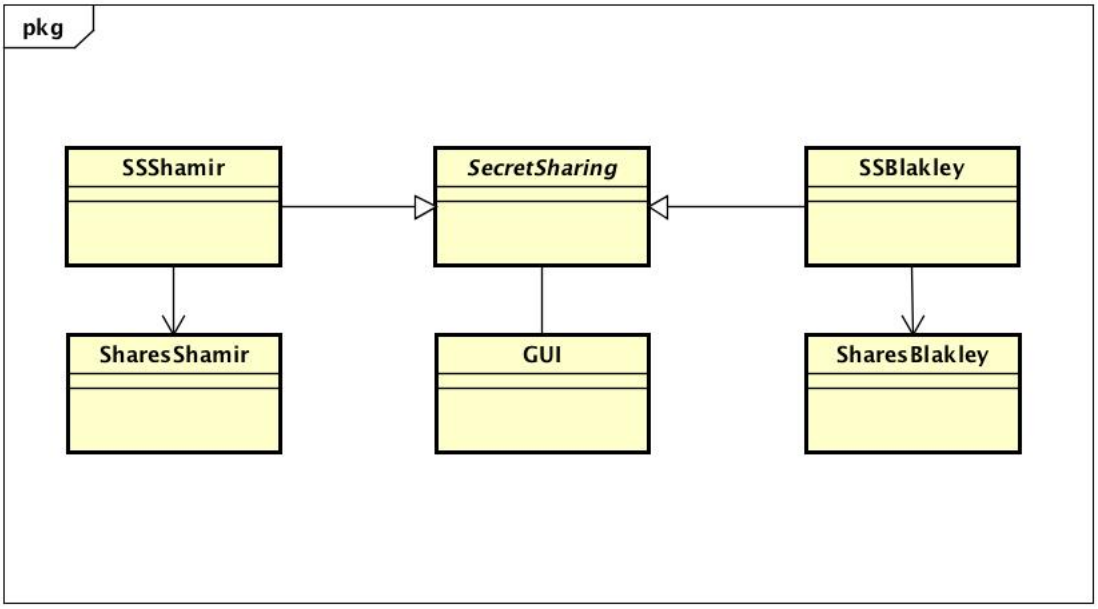
5. Bagian 5

   Pada bagian ini perangkat lunak akan menampilkan hasil rekonstruksi *password* sesuai dengan masukan metode *secret sharing*, *threshold*, dan *share* yang telah dimasukkan oleh pengguna pada Bagian 1, Bagian 2, dan Bagian 3.

4.3 Diagram Kelas Rinci

Subbab ini berisi penjelasan mengenai diagram kelas rinci yang dapat dilihat pada Gambar 4.3. Diagram kelas rinci yang dibahas pada subbab ini menjelaskan kelas-kelas yang digunakan untuk membangun perangkat lunak yang mengimplementasikan penggunaan *secret sharing* untuk berbagi *password* beserta relasi antar kelas-kelas tersebut.

Pada diagram kelas yang dibahas pada subbab ini terdapat beberapa penambahan kelas dari kelas-kelas yang ada pada Diagram Kelas Awal pada Gambar 3.3. Terdapat 6 kelas yang digunakan untuk membangun perangkat lunak. Kelas-kelas tersebut adalah kelas SecretSharing, kelas SSShamir, kelas SSBlakley, kelas SharesShamir, kelas SharesBlakley, dan kelas GUI. Berikut adalah penjelasan untuk setiap kelas yang dibutuhkan untuk membangun perangkat lunak :



Gambar 4.3: Diagram kelas

1. Kelas SecretSharing

   Kelas SecretSharing adalah kelas utama perangkat lunak yang bertipe abstrak dan merupakan kelas *parent* dari kelas SSShamir dan kelas SSBlakley. Kelas ini merupakan generalisasi dari metode-metode *secret sharing* yang digunakan pada perangkat lunak yang dibangun.

2. Kelas SSShamir

   Kelas SSShamir merupakan kelas turunan/*child* dari kelas SecretSharing. Kelas ini mengimplementasikan pembagian *password* dan rekonstruksi *password* menggunakan metode *secret sharing* Shamir.

3. Kelas SSBlakley

   Kelas SSBlakley merupakan kelas turunan/*child* dari kelas SecretSharing. Kelas ini mengimplementasikan pembagian *password* dan rekonstruksi *password* menggunakan metode *secret sharing* Blakley.

4. Kelas SharesShamir

   Kelas SharesShamir adalah kelas yang merepresentasikan *share* yang diperoleh setiap partisipan apabila *password* dibagikan dengan menggunakan metode *secret sharing* Shamir.

5. Kelas SharesBlakley

Kelas SharesShamir adalah kelas yang merepresentasikan *share* yang diperoleh setiap partisipan apabila *password* dibagikan dengan menggunakan metode *secret sharing* Shamir.

6. Kelas GUI

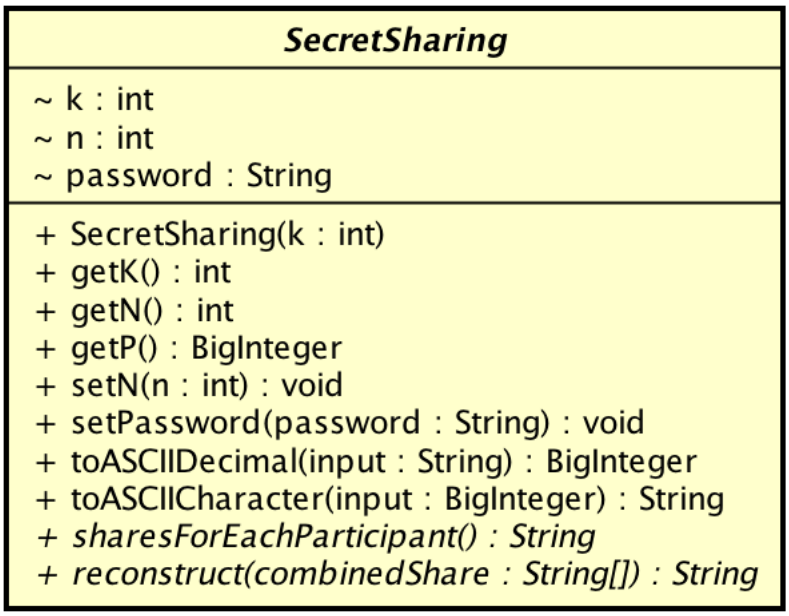
   Kelas GUI merupakan kelas yang menampilkan antarmuka perangkat lunak. Pada kelas GUI ini pengguna dapat memasukkan *input* yang dibutuhkan untuk membagikan *password*. Kelas GUI juga berfungsi untuk menampilkan hasil pembagian *password* yang berupa *share* kepada pengguna.

4.4 Rincian Metode

Pada bagian ini akan dijelaskan mengenai rincian metode yang terdapat pada kelas-kelas yang digunakan pada perangkat lunak yang dibangun.

4.4.1 Kelas SecretSharing

Kelas SecretSharing dapat dilihat pada Gambar 4.4. Kelas ini memiliki 9 buah metode. Penjelasan untuk masing-masing metode pada kelas SecretSharing adalah seperti berikut :



Gambar 4.4: Kelas SecretSharing

- **int getK()**
  Metode ini digunakan untuk memperoleh banyak *threshold*.
  Input : -
  Output : banyak *threshold*

- **int getN()**
  Metode ini digunakan untuk memperoleh banyak partisipan.

Input : -
Output : banyak partisipan.

- **BigInteger getP()**
  Metode ini digunakan untuk memperoleh bilangan prima p.
  Input : -
  Output : bilangan prima p.

- **void setN(int n)**
  Metode ini digunakan untuk mengubah banyak partisipan.
  Input : banyak partisipan yang diinginkan.
  Output : -

- **void setPassword(String password)**
  Metode ini digunakan untuk mengubah *password*.
  Input : *password* yang diinginkan.
  Output : -

- **BigInteger toASCIIDecimal(String password)**
  Metode ini mengubah *password* ke bilangan desimal BigInteger dengan menggunakan format ASCII.
  Input : masukan *password* yang ingin diubah ke bentuk desimal.
  Output : *password* yang telah diubah ke bentuk desimal.
  Algoritma untuk metode ini dapat dilihat pada Algoritma 1.

Algoritma 1 BigInteger toASCIIDecimal(String password)

```
result ← untuk menampung hasil karakter yang diubah ke bilangan desimal
charToInt ← 0
temp ← untuk menampung hasil String sementara
for i ← 0 to password.length − 1 do
    character ← password[i] ← huruf ke-i pada masukan password
    charToInt ← merubah huruf ke-i pada password menjadi Integer sesuai dengan kode ASCII
    temp ← charToInt
    if charToInt/100 < 1 then
        if i = 0 then
            result ← result + temp
        else
            result ← result + ("0" + temp)
        end if
    else
        result ← result + temp
    end if
end for
return result
```

- **String toASCIICharacter(BigInteger passwordDecimal)**
  Metode ini mengubah bilangan desimal BigInteger ke tipe data String dengan menggunakan format ASCII.

Input : bilangan desimal BigInteger yang ingin diubah ke tipe data String.
Output : bilangan desimal yang telah diubah ke tipe data String.
Algoritma untuk metode ini dapat dilihat pada Algoritma 2.

Algoritma 2 String toASCIICharacter(BigInteger passwordDecimal)

```
result ← untuk menampung hasil bilangan desimal yang diubah menjadi String
j ← passwordDecimal.length
temp ← 0
if passwordDecimal.length < 3 then
    result ← password yang diubah ke bentuk String
end if
for i ← passwordDecimal.length − 3 to 0 do
    temp ← passwordDecimal.substring(i, j)
    if i = 2 then
        result ← temp to char + result
        temp ← passwordDecimal.substring(0, 2))
        result ← temp to char + result
    else
        result ← temp to char + result
    end if
end for
return result
```

- **String sharesForEachParticipant()**
  Metode ini adalah metode abstrak yang digunakan untuk membagikan *share* kepada setiap partisipan.
  Input : -
  Output : *share* yang diperoleh setiap partisipan.

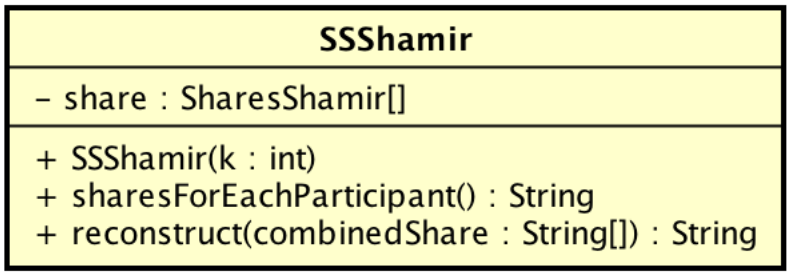
- **String reconstruct(String[] combinedShare)**
  Metode ini adalah metode abstrak yang digunakan untuk merekonstruksi *password*.
  Input : *share* yang ingin digabungkan.
  Output : *password* yang berhasil direkonstruksi.

4.4.2 Kelas SSShamir

Berikut dilampirkan gambar kelas SSShamir beserta penjelasan dari setiap metode yang dimiliki kelas SSShamir :

- **SSShamir(int k)**
  Metode ini merupakan *constructor* dari kelas SSShamir.
  Input : banyak *threshold*.
  Output : -

- **String sharesForEachParticipant()**
  Metode ini berfungsi untuk membagikan *password* ke partisipan dengan menggunakan metode *secret sharing* Shamir.
  Input : -



Gambar 4.5: Kelas SSShamir

Output : *share* yang diperoleh partisipan.

Algoritma untuk metode ini dapat dilihat pada Algoritma 3.

Algoritma 3 String sharesForEachParticipant()

1: $result \leftarrow$ untuk menampung *share* yang akan ditampilkan
2: $a \leftarrow$ konstanta acak
3: $share \leftarrow$ *array share* untuk setiap partisipan
4: $p \leftarrow$ bilangan prima acak
5: **for** $i \leftarrow 0$ **to** $n - 1$ **do**
6: $\quad share[i].X \leftarrow (i + 1)$
7: $\quad share[i].Y \leftarrow passwordDecimal$
8: **end for**
9: **for** $i \leftarrow 0$ **to** $share.length - 1$ **do**
10: $\quad$ **for** $j \leftarrow 0$ **to** $k - 2$ **do**
11: $\quad\quad share[i].Y \leftarrow share[i].Y + ((a + j) \times share[i].X^{(j+1)})$
12: $\quad$ **end for**
13: $\quad share[i].Y \leftarrow share[i].Y \ mod \ p$
14: **end for**
15: $pEncrypt \leftarrow$ untuk menyembunyikan nilai dari bilangan prima pada *share*
16: **for** $i \leftarrow 0$ **to** $share.length - 1$ **do**
17: $\quad pEncrypt[i] \leftarrow p - (share[i].Y + share[i].X)$
18: $\quad result \leftarrow result \ + (share[i].X + " - " + share[i].Y + " - " + pEncrypt[i] + " \hookleftarrow ") \leftarrow$ *share* yang diperoleh setiap partisipan
19: $\quad share[i].X \leftarrow 0 \leftarrow$ *reset* nilai X pada setiap *share*
20: $\quad share[i].Y \leftarrow 0 \leftarrow$ *reset* nilai Y pada setiap *share*
21: **end for**
22: $p \leftarrow 0 \leftarrow$ *reset* bilangan prima
23: $password \leftarrow null \leftarrow$ *reset password*
24: **return** *result*

- **String reconstruct(String[] combinedShare)**

  Metode ini berfungsi untuk membagikan *password* ke partisipan dengan menggunakan metode *secret sharing* Blakley.

  Input : *share* yang ingin digabungkan.

  Output : *password* yang berhasil direkonstruksi.

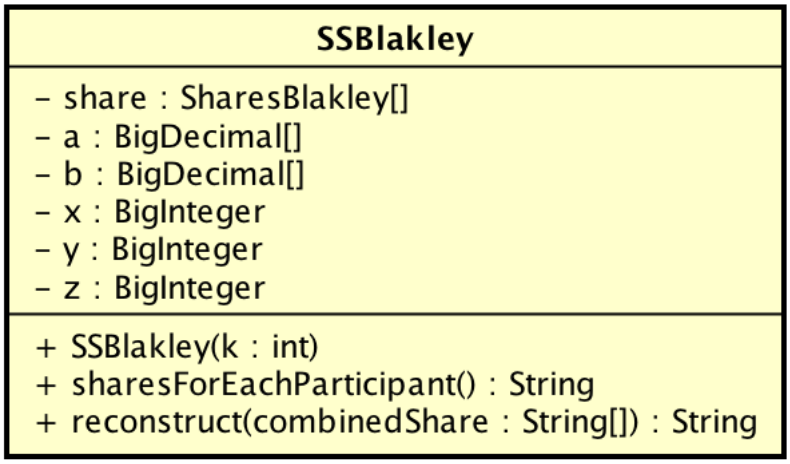
  Algoritma untuk metode ini dapat dilihat pada Algoritma 4.

Algoritma 4 String reconstruct(String[] combinedShare)

1: $result \leftarrow 0$
2: $splitShare \leftarrow combinedShare[0].split(" - ")$ ← memisahkan *share* berdasarkan "-"
3: $p \leftarrow splitShare[2] + splitShare[1] - splitShare[0]$ ← memperoleh bilangan prima dari *share* pertama
4: **for** $i \leftarrow 0$ **to** $k - 1$ **do**
5: $splitShare = combinedShare[i].split(" - ")$
6: $share[i].X \leftarrow splitShare[0]$
7: $share[i].Y \leftarrow splitShare[1]$
8: **end for**
9: **for** $i \leftarrow 0$ **to** $k - 1$ **do**
10: $numerator \leftarrow 0$ ← pembilang
11: $denominator \leftarrow 0$ ← penyebut
12: **for** $j \leftarrow 0$ **to** $k - 1$ **do**
13: **if** $i! = j$ **then**
14: $start \leftarrow share[i].X$
15: $next \leftarrow share[i].X$
16: $numerator \leftarrow numerator \times ((-next) \ mod \ p)$
17: $denominator \leftarrow (denominator \times (start - next)) \ mod \ p$
18: **end if**
19: **end for**
20: $value \leftarrow share[i].Y$
21: $temp \leftarrow value \times numerator \times (denominator \ mod \ inverse \ p)$
22: $result \leftarrow (p + result + temp) \ mod \ p$
23: **end for**
24: **return** *result*

4.4.3 Kelas SSBlakley

Kelas SSBlakley dapat dilihat pada Gambar 4.6. Berikut adalah penjelasan dari setiap metode yang dimiliki kelas SSBlakley :



Gambar 4.6: Kelas SSBlakley

- **SSBlakley(int k)**
  Metode ini merupakan *constructor* dari kelas SSBlakley.
  Input : banyak *threshold*.
  Output : -

- **String sharesForEachParticipant()**
  Metode ini berfungsi untuk membagikan *password* ke partisipan dengan menggunakan metode *secret sharing* Blakley.
  Input : -
  Output : *share* yang diperoleh partisipan.
  Algoritma untuk metode ini dapat dilihat pada Algoritma 5.

- **String reconstruct(String[] combinedShare)**
  Metode ini berfungsi untuk membagikan *password* ke partisipan dengan menggunakan metode *secret sharing* Blakley.
  Input : *share* yang ingin digabungkan.
  Output : *password* yang berhasil direkonstruksi.
  Algoritma untuk metode ini dapat dilihat pada Algoritma 6.

4.4.4 Kelas SharesShamir

Berikut dilampirkan gambar kelas SharesShamir beserta penjelasan dari metode-metode yang digunakan pada kelas SharesShamir :

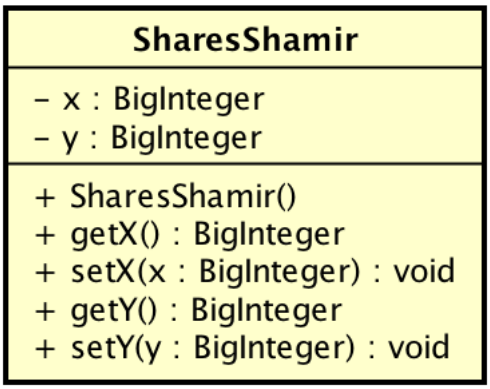
- **SharesShamir()**
  Metode ini merupakan *constructor* dari kelas SharesShamir.

Algoritma 5 String sharesForEachParticipant()

result ← untuk menampung *share* yang akan ditampilkan
share ← *array share* untuk setiap partisipan
$a_0 \leftarrow 1$
$b_0 \leftarrow 2$
$a_1 \leftarrow 2$
$b_1 \leftarrow 3$
$a_2 \leftarrow 5$
$b_2 \leftarrow 7$
passwordDecimal ← *password* dalam bilangan desimal sesuai dengan kode ASCII
counter ← untuk menyimpan panjang *password* dalam bilangan desimal
if $passwordDecimal.length < 3$ **then**
  $x \leftarrow 0$
  $z \leftarrow passwordDecimal \ mod \ 10$
  $y \leftarrow (passwordDecimal - z)/10$
else
  if $passwordDecimal.length \ mod \ 3 = 0$ **then**
    $counter \leftarrow passwordDecimal.length/3$
  else
    $counter \leftarrow passwordDecimal.length/3$ ← hasil operasi ini dibulatkan ke atas
  end if
  $modd \leftarrow 10^{(counter)}$
  $z \leftarrow passwordDecimal \ mod \ modd$
  $y \leftarrow ((passwordDecimal - z)/modd) \ mod \ modd$
  $x \leftarrow (((passwordDecimal - z)/modd) - y)/modd$
end if
$arrayXYZ \leftarrow x, y, z$
$largest \leftarrow 0$
for $i \leftarrow 0$ **to** 2 **do**
  if $arrayXYZ[i] > largest$ **then**
    $largest \leftarrow arrayXYZ[i]$
  end if
end for
$p \leftarrow$ menentukan bilangan prima yang panjang bit-nya lebih dari *largest*
$temp \leftarrow 1$
for $i \leftarrow 0$ **to** $share.length - 1$ **do**
  $share[i].A \leftarrow a_i$
  $share[i].B \leftarrow b_i$
  $share[i].C \leftarrow (z - (share[i].A \times x) - (share[i].B \times y)) \ mod \ p$
end for
pEncrypt ← untuk menyembunyikan nilai dari bilangan prima pada *share*
for $i \leftarrow 0$ **to** $share.length - 1$ **do**
  $pEncrypt[i] \leftarrow p + share[i].B - share[i].C$
  $result \leftarrow result + share[i].C + " - " + pEncrypt[i] + " - " + (counter + i) + " \hookleftarrow "$
  $share[i].A \leftarrow 0 \leftarrow$ *reset* nilai A pada setiap *share*
  $share[i].B \leftarrow 0 \leftarrow$ *reset* nilai B pada setiap *share*
  $share[i].C \leftarrow 0 \leftarrow$ *reset* nilai C pada setiap *share*
end for
$p \leftarrow 0 \leftarrow$ *reset* bilangan prima
password ← *null* ← *reset password*
return *result*

Algoritma 6 String reconstruct(String[] combinedShare)

$result \leftarrow$ untuk menyimpan hasil rekonstruksi *password*
$MatrixA \leftarrow Matrix(3, 3)$
$MatrixB \leftarrow Matrix(3, 1)$
$MatrixSecret \leftarrow Matrix(3, 1)$
$share[0].A \leftarrow 1$
$share[0].B \leftarrow 2$
$share[1].A \leftarrow 2$
$share[1].B \leftarrow 3$
$share[2].A \leftarrow 5$
$share[2].B \leftarrow 7$
$splitShare \leftarrow combinedShare[0].split(" - ") \leftarrow$ memisahkan *share* berdasarkan "-"
$p \leftarrow splitShare[1] + splitShare[0] - share[0].B \leftarrow$ memperoleh bilangan prima dari *share* pertama
for $i \leftarrow 0$ **to** $k - 1$ **do**
  $splitShare \leftarrow combinedShare[i].split(" - ")$
  $share[i].C \leftarrow splitShare[0]$
end for
for $i \leftarrow 0$ **to** 2 **do**
  for $j \leftarrow 0$ **to** 2 **do**
    if $j = 2$ **then**
      $MatrixA[i][j] \leftarrow (-1)$
    else if $j = 1$ **then**
      $MatrixA[i][j] \leftarrow share[i].B$
    else
      $MatrixA[i][j] \leftarrow share[i].A$
    end if
  end for
end for
for $i \leftarrow 0$ **to** 2 **do**
  $MatrixB[i][0] \leftarrow (-share[i].C)$
end for
$MatrixSecret \leftarrow MatrixA.solve(MatrixB) \leftarrow$ melakukan operasi baris elementer untuk memperoleh pesan rahasia
$x \leftarrow MatrixSecret[0][0] \ mod \ p$
$y \leftarrow MatrixSecret[1][0] \ mod \ p$
$z \leftarrow MatrixSecret[2][0] \ mod \ p$
$counter \leftarrow splitShare[2] - 2$
if $z.length < counter$ **then**
  $c \leftarrow counter - z.length$
  for $i \leftarrow 0$ **to** $c - 1$ **do**
    $z \leftarrow$ "0" $+ z$
  end for
end if
if $y.length < counter$ **then**
  $c \leftarrow counter - y.length$
  for $i \leftarrow 0$ **to** $c - 1$ **do**
    $y \leftarrow$ "0" $+ y$
  end for
end if
$result \leftarrow x + y + z \leftarrow$ menyatukan x, y, dan z
return *result*



Gambar 4.7: Kelas SharesShamir

Input : -
Output : -

- **BigInteger getX()**
  Metode ini berfungsi untuk memperoleh nilai dari variabel x pada *share*.
  Input : -
  Output : nilai dari variabel x.

- **void setX(BigInteger x)**
  Metode ini digunakan untuk mengubah nilai dari variabel x.
  Input : nilai variabel x yang baru.
  Output : -

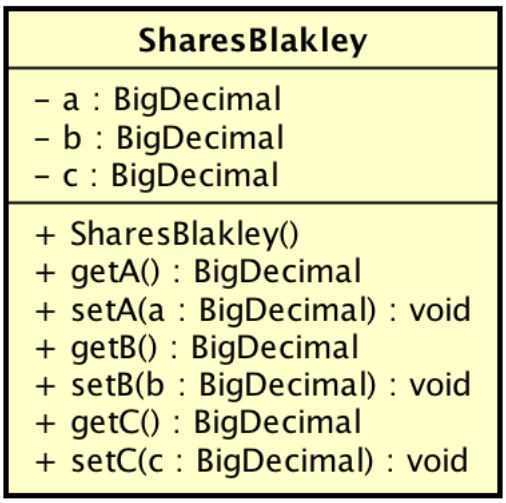
- **BigInteger getY()**
  Metode ini berfungsi untuk memperoleh nilai dari variabel y pada *share*.
  Input : -
  Output : nilai dari variabel y.

- **void setY(BigInteger y)**
  Metode ini digunakan untuk mengubah nilai dari variabel y pada *share*.
  Input : nilai variabel y yang baru.
  Output : -

4.4.5 Kelas SharesBlakley

Berikut dilampirkan gambar kelas SharesBlakley beserta penjelasan dari metode-metode yang digunakan pada kelas SharesBlakley :

- **SharesBlakley()**
  Metode ini merupakan *constructor* dari kelas SharesBlakley.
  Input : -
  Output : -



Gambar 4.8: Kelas SharesBlakley

- **BigDecimal getA()**
  Metode ini berfungsi untuk memperoleh nilai dari variabel *a* pada *share*.
  Input : -
  Output : nilai dari variabel *a*.

- **void setA(BigDecimal a)**
  Metode ini digunakan untuk merubah nilai dari variabel *a*.
  Input : nilai baru dari variabel *a*.
  Output : -

- **BigDecimal getB()**
  Metode ini berfungsi untuk memperoleh nilai dari variabel *ab* pada *share*.
  Input : -
  Output : nilai dari variabel *b*.

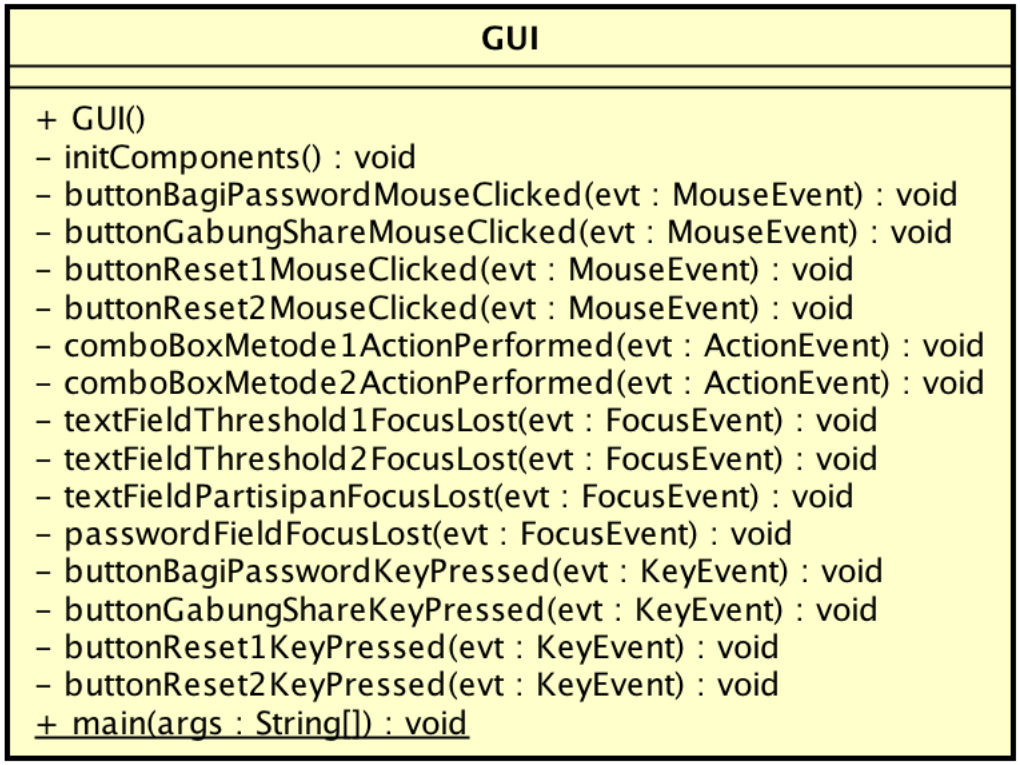
- **void setB(BigDecimal b)**
  Metode ini digunakan untuk merubah nilai dari variabel *b*.
  Input : nilai baru dari variabel *b*.
  Output : -

- **BigDecimal getC()**
  Metode ini berfungsi untuk memperoleh nilai dari variabel *c* pada *share*.
  Input : -
  Output : nilai dari variabel *c*.

- **void setC(BigDecimal c)**
  Metode ini digunakan untuk merubah nilai dari variabel *c*.
  Input : nilai baru dari variabel *c*.
  Output : -

4.4.6 Kelas GUI

Kelas GUI adalah kelas yang digunakan untuk menampilkan antarmuka yang dapat berinteraksi langsung dengan pengguna. Pada kelas ini perangkat lunak yang dibangun dapat menerima masukan dari pengguna. Kelas ini juga menampilkan keluaran yang dihasilkan perangkat lunak. Kelas GUI dapat dilihat pada Gambar 4.9.



Gambar 4.9: Kelas GUI

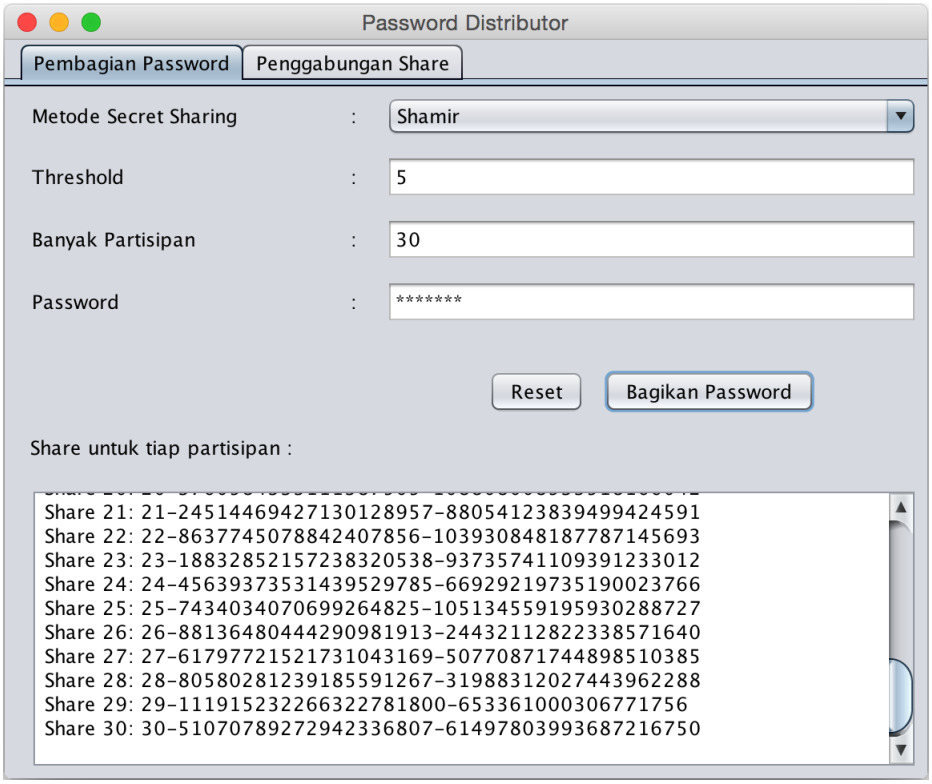
BAB 5

IMPLEMENTASI DAN PENGUJIAN

Bab ini akan membahas implementasi antarmuka perangkat lunak yang dibangun beserta pengujian yang dilakukan pada perangkat lunak. Pengujian perangkat lunak yang dilakukan pada skripsi ini adalah pengujian fungsional dan pengujian eksperimental.

5.1 Implementasi Antarmuka

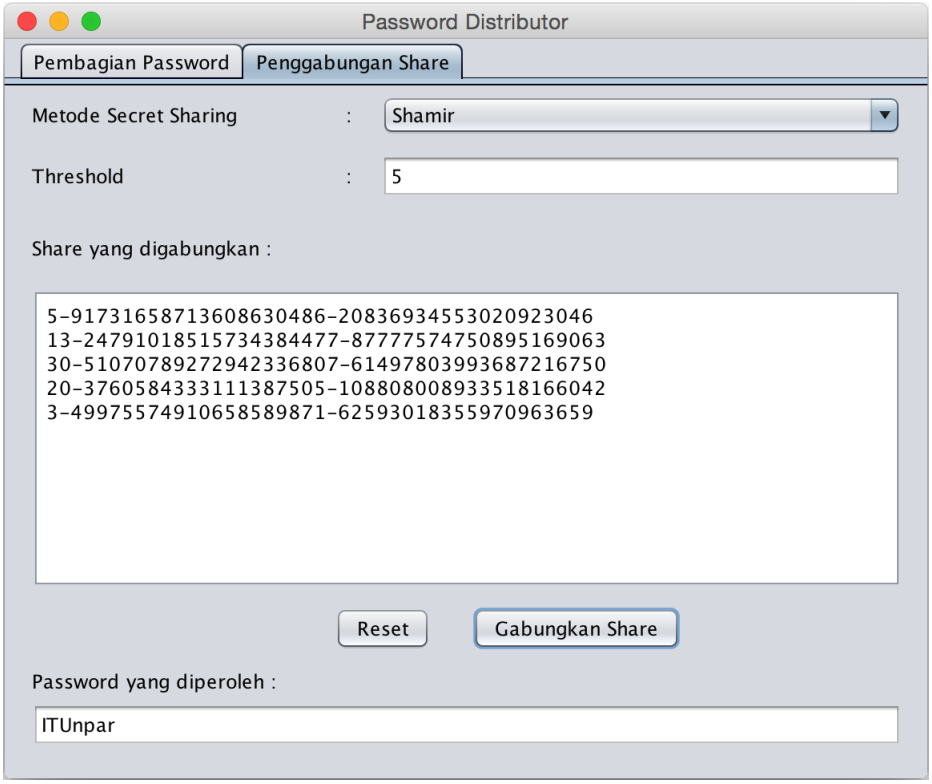
Pada bagian ini akan ditunjukkan hasil implementasi perangkat lunak penggunaan *secret sharing* untuk berbagi *password*. Perangkat lunak yang dibangun akan menampilkan dua buah tampilan antarmuka, yaitu tampilan antarmuka untuk pembagian *password* dan antarmuka untuk penggabungan *share*.



Gambar 5.1: Antarmuka untuk proses pembagian *password*

Gambar 5.1 menunjukkan tampilan antarmuka untuk proses pembagian *password*. Pengguna akan diminta untuk memilih metode *secret sharing* yang digunakan, setelah itu pengguna dapat

memasukkan *input* berupa *threshold*, banyak partisipan, dan *password* yang ingin dibagikan. Tombol "Bagikan *Password*" digunakan untuk membagikan masukan *password* ke partisipan sejumlah banyak partisipan yang telah ditentukan oleh pengguna. *Password* akan dibagikan dengan menggunakan metode *secret sharing* yang dipilih. Perangkat lunak akan menampilkan *share-share* yang diperoleh setiap partisipan pada *text area* "*Share* untuk tiap partisipan". Tombol "*Reset*" pada antarmuka digunakan untuk menampilkan tampilan awal perangkat lunak dengan mengosongkan semua *text field* dan *text area* pada tampilan antarmuka.



Gambar 5.2: Antarmuka untuk proses penggabungan *share*

Gambar 5.2 menunjukkan tampilan antarmuka untuk proses penggabungan *share*. Pada tampilan antarmuka ini pengguna dapat menggabungkan *share-share* yang telah ditampilkan di antarmuka pembagian *password* pada Gambar 5.1. Pengguna akan diminta untuk memilih metode *secret sharing* yang digunakan, lalu memasukkan *threshold* dan *share-share* yang ingin digabungkan. Pengguna dapat merekonstruksi *password* dengan menekan tombol "Gabungkan *Share*". *Password* yang berhasil direkonstruksi akan ditampilkan pada *text field* "*Password* yang diperoleh". Tombol "*Reset*" pada antarmuka ini digunakan untuk menampilkan tampilan awal antarmuka proses penggabungan *share*.

5.2 Pengujian Fungsional

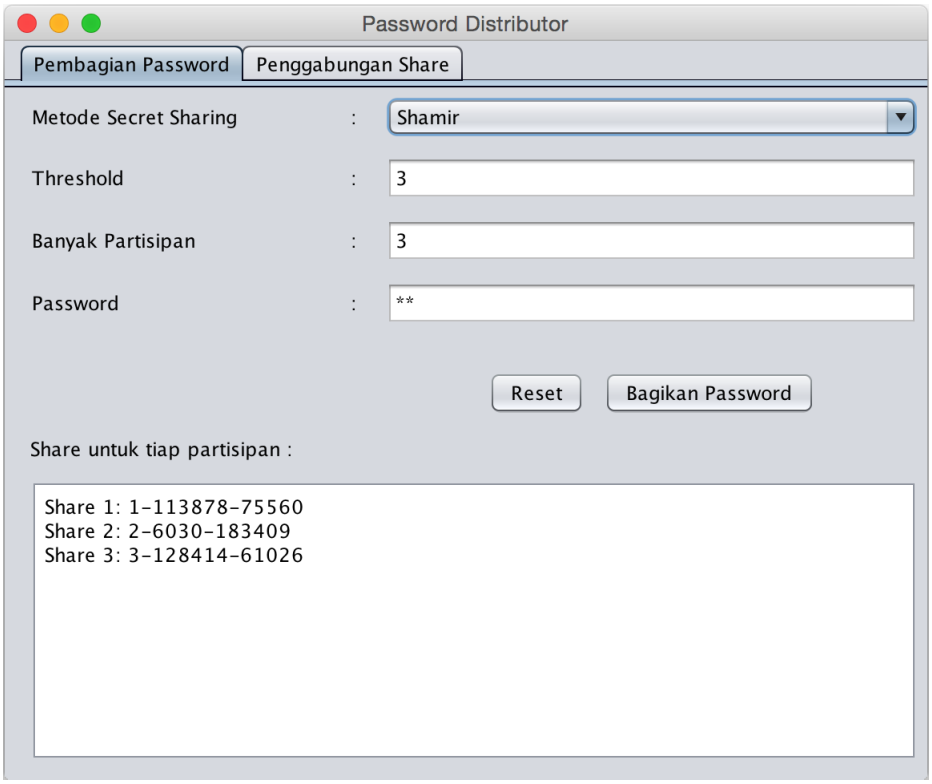
Subbab ini akan membahas mengenai pengujian fungsional yang dilakukan pada perangkat lunak yang dibangun. Pengujian fungsional yang dilakukan akan menguji implementasi metode *secret sharing* Shamir dan metode *secret sharing* Blakley pada perangkat lunak. Pengujian ini akan melihat keluaran yang dihasilkan oleh perangkat lunak yang dibangun berdasarkan masukan yang diterima.

Pengujian fungsional yang dilakukan akan menunjukkan apakah perangkat lunak yang dibangun dapat membagikan *password* dengan menggunakan metode *secret sharing* Shamir dan *secret sharing* Blakley. Pada pengujian ini juga dilihat hasil rekonstruksi *password* dengan menggabungkan *share* yang dihasilkan, apakah sudah sesuai dengan masukan *password*.

Pada pengujian fungsional yang dilakukan, *Threshold* yang digunakan adalah 3, dan banyak partisipan yang digunakan untuk kedua metode adalah 3. Dalam pengujian ini, *password* yang akan dibagikan menggunakan metode *secret sharing* Shamir dan metode *secret sharing* Blakley adalah "IT".

5.2.1 Pengujian Fungsional Metode Secret Sharing Shamir

Bagian ini akan menjelaskan pengujian fungsional untuk metode *secret sharing* Shamir. Pada Gambar 5.3 dapat dilihat pengujian fungsional perangkat lunak untuk metode *secret sharing* Shamir menggunakan masukan *threshold* sebanyak 3, banyak partisipan 3, dan masukan *password* "IT". Perhitungan manual untuk proses pembagian *password* dan proses penggabungan *share* pada metode *secret sharing* Shamir dapat dilihat di Bab 3 Analisis pada Subbab 3.2.1.



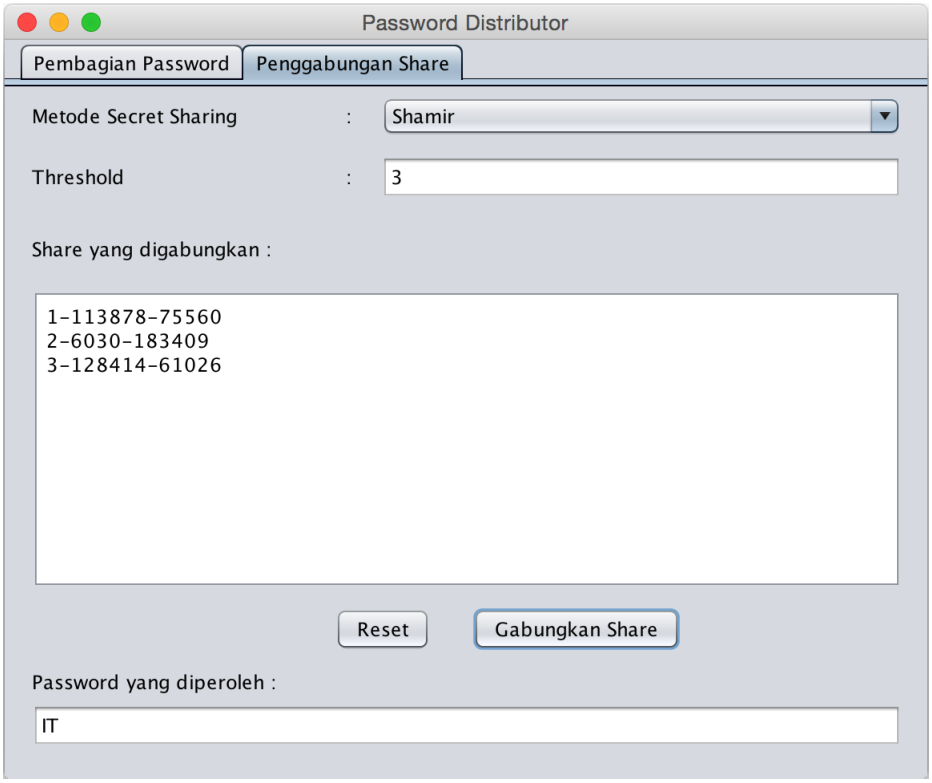
Gambar 5.3: Pengujian fungsional metode *secret sharing* Shamir untuk proses pembagian *password*

Gambar 5.3 menunjukkan hasil pembagian *password* menggunakan metode *secret sharing* Shamir untuk masukan *password* "IT" dengan *threshold* 3 dan banyak partisipan 3. Pada gambar tersebut dapat dilihat bahwa perangkat lunak berhasil membagikan *password* kepada partisipan dengan menggunakan metode *secret sharing* Shamir. *Share* yang diperoleh partisipan dapat dilihat pada *text area* "*Share* untuk tiap partisipan". *Password* dapat direkonstruksi dengan menggabungkan *share-share* tersebut pada tampilan antarmuka "Penggabungan *Share*".

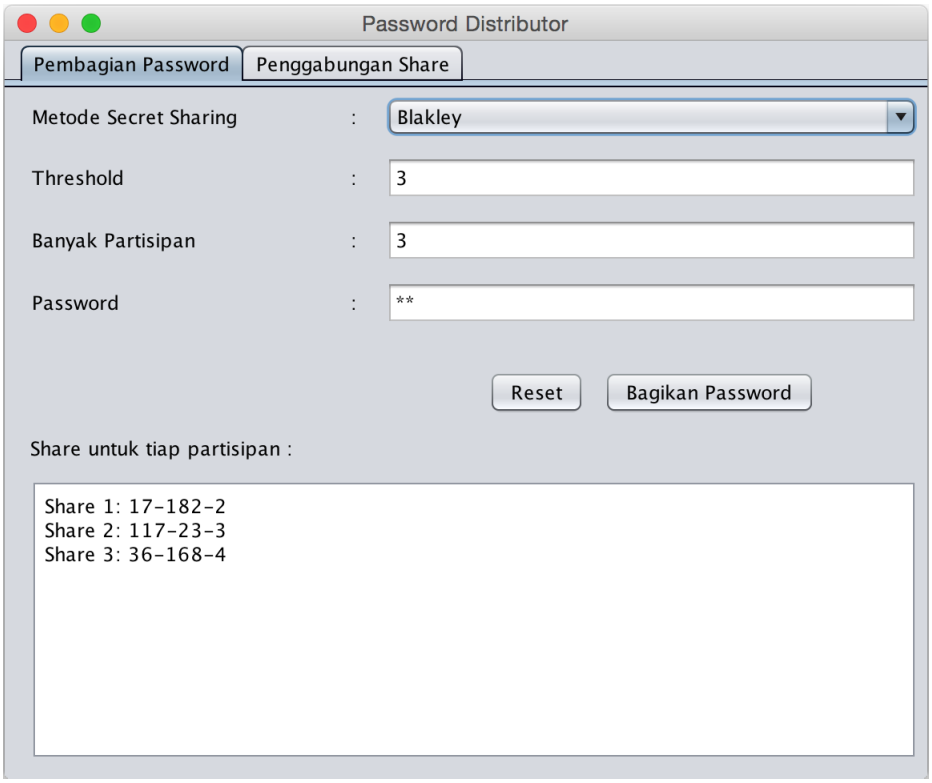
Gambar 5.4 menunjukkan hasil penggabungan *share-share* yang dihasilkan dari proses pembagian *password*. Pada gambar tersebut dapat dilihat bahwa perangkat lunak berhasil merekonstruksi *password* sesuai dengan masukkan *password* yang dimasukkan pengguna pada tampilan antarmuka proses pembagian *password*. *Password* yang berhasil direkonstruksi dapat dilihat pada *text field* "*Password* yang diperoleh".

5.2.2 Pengujian Fungsional Metode Secret Sharing Blakley

Bagian ini akan menjelaskan pengujian fungsional untuk metode *secret sharing* Blakley. Pada Gambar 5.5 dapat dilihat pengujian fungsional perangkat lunak untuk metode *secret sharing* Blakley menggunakan masukan *threshold* sebanyak 3, banyak partisipan 3, dan masukan *password* "IT". Perhitungan manual untuk proses pembagian *password* dan proses penggabungan *share* pada metode *secret sharing* Blakley dapat dilihat di Bab 3 Analisis pada Subbab 3.2.2.



Gambar 5.4: Pengujian fungsional metode *secret sharing* Shamir untuk proses penggabungan *share*



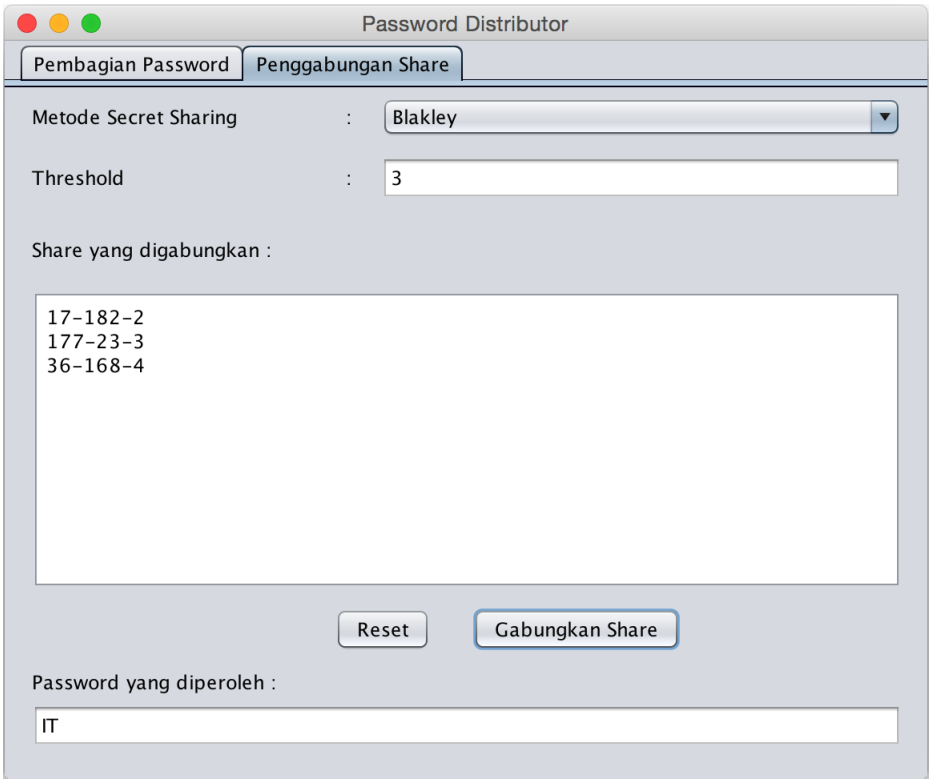
Gambar 5.5: Pengujian fungsional metode *secret sharing* Blakley untuk proses pembagian *password*

Gambar 5.5 menunjukkan hasil pembagian *password* menggunakan metode *secret sharing* Blakley untuk masukan *password* "IT" dengan *threshold* 3 dan banyak partisipan 3. Pada gambar tersebut dapat dilihat bahwa perangkat lunak berhasil membagikan *password* kepada partisipan dengan menggunakan metode *secret sharing* Blakley. *Share* yang diperoleh dapat dilihat pada *text area* "*Share* untuk tiap partisipan". *Password* dapat direkonstruksi dengan menggabungkan *share-share* tersebut pada tampilan antarmuka "Penggabungan *Share*".

Gambar 5.6 menunjukkan hasil penggabungan *share-share* yang dihasilkan dari proses pembagian *password*. Pada gambar tersebut dapat dilihat bahwa perangkat lunak berhasil merekonstruksi *password* sesuai dengan masukan *password* yang telah dimasukkan pengguna pada tampilan antarmuka "Pembagian *Password*". *Password* yang berhasil direkonstruksi dapat dilihat pada *text field* "*Password* yang diperoleh".

5.2.3 Kesimpulan Pengujian Fungsional

Berdasarkan pengujian fungsional yang telah dilakukan pada metode *secret sharing* Shamir dan metode *secret sharing* Blakley, dapat disimpulkan bahwa perangkat lunak sudah dapat mengimplementasikan metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Perangkat lunak berhasil mengeluarkan hasil rekonstruksi *password* yang sesuai dengan masukan *password* yang ingin dibagikan. Jumlah *share* yang ditampilkan oleh perangkat lunak juga sudah sesuai dengan jumlah partisipan yang dimasukkan oleh pengguna.



Gambar 5.6: Pengujian fungsional metode *secret sharing* Blakley untuk proses penggabungan *share*

5.3 Pengujian Eksperimental

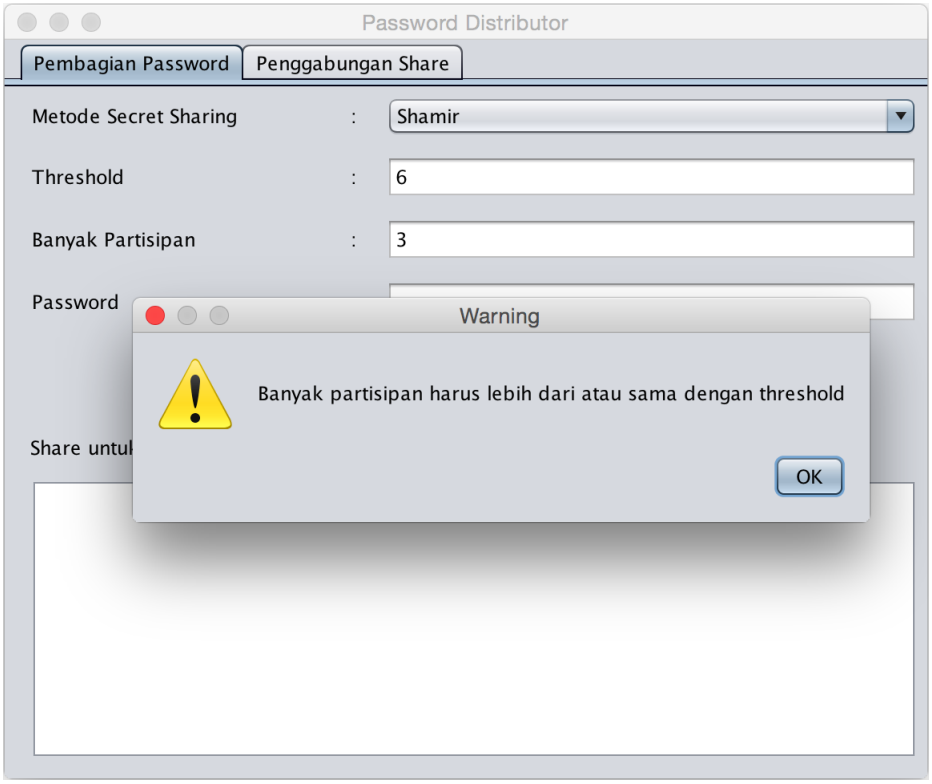
Pada subbab ini akan dibahas mengenai pengujian eksperimental yang dilakukan pada perangkat lunak yang dibangun. Perangkat lunak yang mengimplementasikan metode *secret sharing* Shamir dan metode *secret sharing* Blakley akan diuji dengan berbagai eksperimen masukan. Pengujian yang dilakukan akan melihat respon dari perangkat lunak terhadap berbagai macam percobaan masukan. Pengujian fungsional untuk metode *secret sharing* Shamir dan metode *secret sharing* Blakley akan dibahas pada Subbab 5.3.1 dan Subbab 5.3.2.

Sebelum dilakukan pengujian untuk metode *secret sharing* Shamir dan metode *secret sharing* Blakley, terlebih dulu akan dilakukan pengujian eksperimental perangkat lunak secara umum. Beberapa percobaan masukan akan diuji untuk melihat respon perangkat lunak bila terdapat kesalahan masukan.

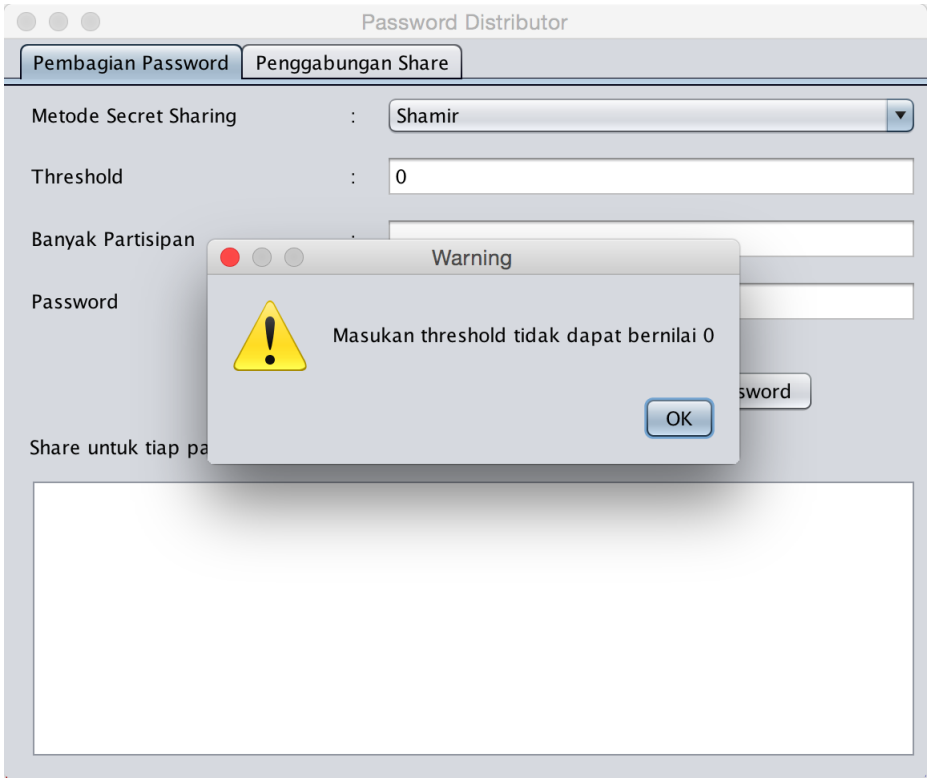
Seperti yang sudah dibahas pada Subbab 2.3 mengenai dasar teori *secret sharing*, dalam metode *secret sharing* banyak partisipan harus lebih besar atau sama dengan *threshold*. Dapat dilihat pada Gambar 5.7 bahwa perangkat lunak akan menampilkan *pop up* berupa peringatan apabila banyak partisipan yang dimasukkan oleh pengguna lebih sedikit dari banyak *threshold*.

Banyak *threshold* dan banyak partisipan yang dimasukkan oleh pengguna juga tidak dapat bernilai 0. Perangkat lunak dapat menangani kesalahan masukan ini pada Gambar 5.8 dan Gambar 5.9.

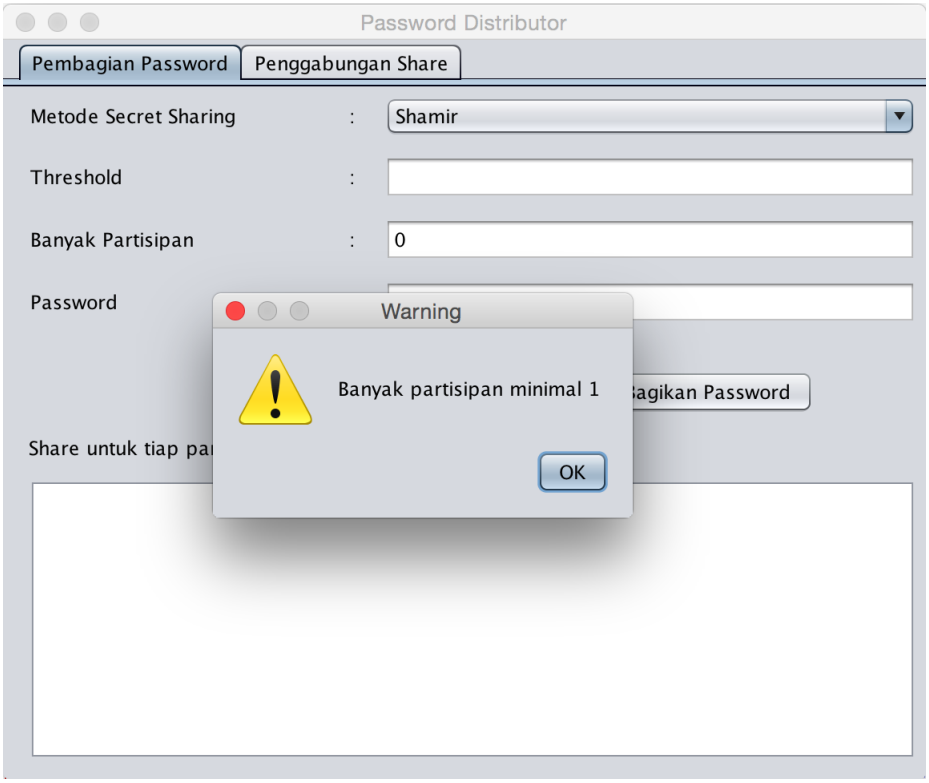
Dapat dilihat pada Gambar 5.10 dan Gambar 5.11, perangkat lunak akan menangani apabila masukan *threshold* dan banyak partisipan yang dimasukkan oleh pengguna bukan berupa bilangan bulat.



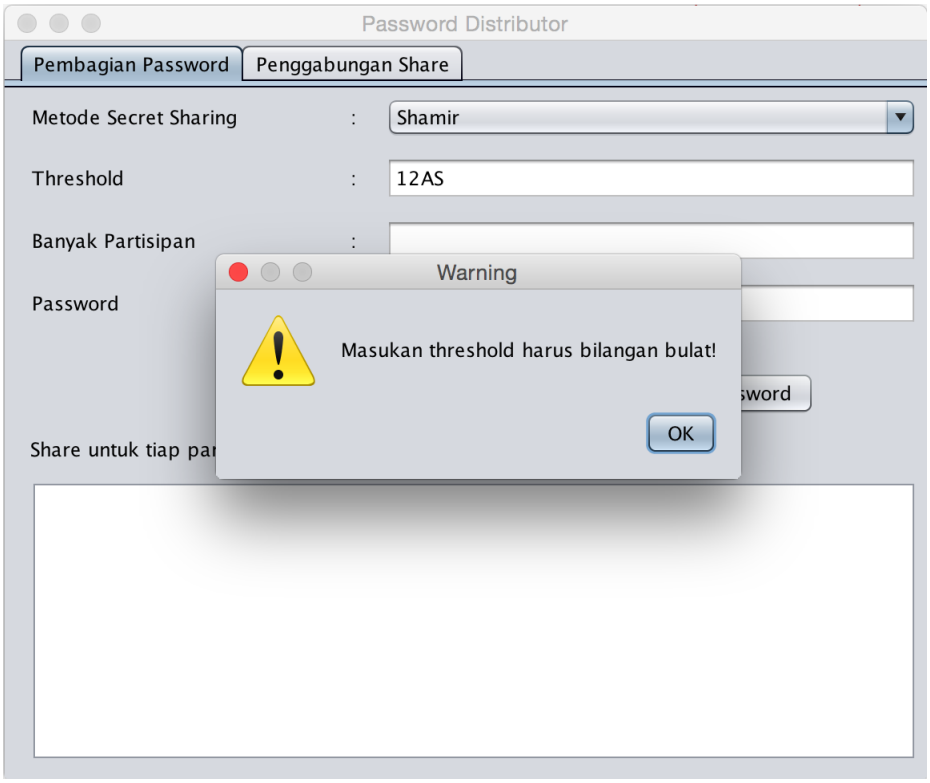
Gambar 5.7: Pengujian eksperimental dengan masukan banyak partisipan lebih sedikit dari *threshold*



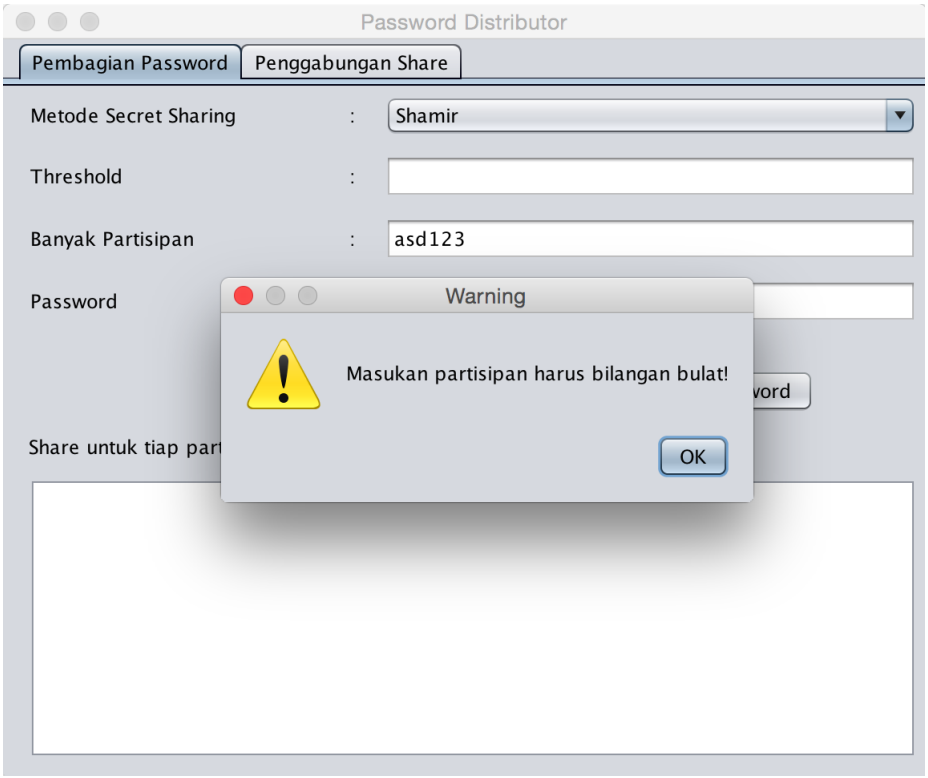
Gambar 5.8: Pengujian eksperimental dengan masukan *threshold* 0



Gambar 5.9: Pengujian eksperimental dengan banyak partisipan 0



Gambar 5.10: Pengujian eksperimental dengan masukan *threshold* bukan bilangan bulat



Gambar 5.11: Pengujian eksperimental dengan masukan banyak partisipan bukan bilangan bulat

5.3.1 Pengujian Eksperimental Metode Secret Sharing Shamir

Bagian ini akan membahas mengenai pengujian eksperimental yang dilakukan pada perangkat lunak dengan menggunakan metode *secret sharing* Shamir. Pada pengujian ini akan diuji implementasi dari metode *secret sharing* Shamir yang menggunakan skema $(3, 3)$ dengan berbagai percobaan masukan *password* dengan panjang karakter yang berbeda-beda. Eksperimen pertama akan mencoba implementasi metode *secret sharing* Shamir pada perangkat lunak menggunakan masukan *password* dengan panjang karakter 5 karakter. Eksperimen kedua menggunakan panjang *password* 10 karakter, dan eksperimen ketiga menggunakan panjang *password* 20 karakter. Pada eksperimen pertama *password* yang digunakan adalah "Unpar", eksperimen kedua menggunakan *password* "ITUnpar123", dan eksperimen ketiga menggunakan *password* "InformatikaUnpar2012".

Hasil pengujian yang diharapkan adalah *password* yang direkonstruksi menggunakan metode *secret sharing* Shamir sesuai dengan masukan *password* sebelum dibagikan. Hasil dari pengujian ini akan menampilkan *share* yang dihasilkan beserta hasil rekonstruksi *password* dengan panjang karakter 5, 10, dan 20 karakter. Hasil pengujian dapat dilihat pada Tabel 5.1, Tabel 5.2, dan Tabel 5.3.

Tabel 5.1: Tabel eksperimen pertama pengujian fungsional metode *secret sharing* Shamir

Share yang Diperoleh	Hasil Penggabungan *Share* dalam Bilangan Desimal	Hasil Rekonstruksi *Password*
Share 1 : 1-146520075412591-94250309849371 *Share* 2 : 2-28569616781585-212200768480378 *Share* 3 : 3-212799506728018-27970878533946	85110112097114	Unpar

Tabel 5.2: Tabel eksperimen kedua pengujian fungsional metode *secret sharing* Shamir

Share yang Diperoleh	Hasil Penggabungan *Share* dalam Bilangan Desimal	Hasil Rekonstruksi *Password*
Share 1 : 1-388878198255495718298910656 78-566162761149874764476060726 26 *Share* 2 : 2-659993851969615695390722352 36-295047107435754787384249030 69 *Share* 3 : 3-589146852838110419640954204 22-365894106567260063134017178 84	730840851101120 97114049050051	ITUnpar123

Tabel 5.3: Tabel eksperimen ketiga pengujian fungsional metode *secret sharing* Shamir

Share yang Diperoleh	Hasil Penggabungan *Share* dalam Bilangan Desimal	Hasil Rekonstruksi *Password*
Share 1 : 1-172826471353033982822970041836 62059132962232443511773203976 -156898517749488911384571622903 4 43921750816585517600418128786 *Share* 2 : 2-798089020684742863195274294238 97988062441321063547414846590 -943722628163180233473411976632 0 79928213374968975647764861 73 *Share* 3 : 3-865077020258344635419387537 40 69889101477054501304478164413 1 -876734628589578461249298733464070 89869008272948067409688633	731101021111141 0909711610510709 708511011209711 4050048049050	InformatikaUnpar2012

5.3.2 Pengujian Eksperimental Metode Secret Sharing Blakley

Pada bagian ini akan dijelaskan mengenai pengujian eksperimental pada perangkat lunak yang dibangun. Bagian ini akan menguji implementasi perangkat lunak menggunakan metode *secret sharing* Blakley. Pada pengujian ini skema *secret sharing* yang digunakan adalah skema $(3, 3)$ dengan berbagai percobaan masukan *password* dengan panjang karakter yang berbeda-beda. Eksperimen pertama akan mencoba implementasi metode *secret sharing* Blakley pada perangkat lunak menggunakan masukan *password* dengan panjang karakter 5 karakter. Eksperimen kedua menggunakan panjang *password* 10 karakter, dan eksperimen ketiga menggunakan panjang *password* 20 karakter. Pada eksperimen pertama *password* yang digunakan adalah "Unpar", eksperimen kedua menggunakan *password* "ITUnpar123", dan eksperimen ketiga menggunakan *password* "InformatikaUnpar2012".

Hasil pengujian yang diharapkan adalah *password* yang direkonstruksi menggunakan metode *secret sharing* Blakley sesuai dengan masukan *password* sebelum dibagikan. Hasil dari pengujian ini akan menampilkan *share* yang dihasilkan beserta hasil rekonstruksi *password* dengan panjang karakter 5, 10, dan 20 karakter. Hasil pengujian dapat dilihat pada Tabel 5.4, Tabel 5.5, dan Tabel 5.6.

Tabel 5.4: Tabel eksperimen pertama pengujian fungsional metode *secret sharing* Blakley

Share yang Diperoleh	Hasil Penggabungan *Share* dalam Bilangan Desimal	Hasil Rekonstruksi *Password*
Share 1 : 86363-47470-5 *Share* 2 : 76732-57102-6 *Share* 3 : 46719-87119-7	85110112097114	Unpar

Tabel 5.5: Tabel eksperimen kedua pengujian fungsional metode *secret sharing* Blakley

Share yang Diperoleh	Hasil Penggabungan *Share* dalam Bilangan Desimal	Hasil Rekonstruksi *Password*
Share 1 : 1295789778-5991876663-10 *Share* 2 : 6841405655-446260787-11 *Share* 3 : 604044258-6683622188-12	730840851101120 97114049050051	ITUnpar123

Tabel 5.6: Tabel eksperimen ketiga pengujian fungsional metode *secret sharing* Blakley

Share yang Diperoleh	Hasil Penggabungan *Share* dalam Bilangan Desimal	Hasil Rekonstruksi *Password*
Share 1 : 97865178798653858693 -19835409752175602686-20 *Share* 2 : 19393117516571596683 -983074710342578646 97-21 *Share* 3 : 65917648251842343683 -517829402989871177 01-22	731101021111141 0909711610510709 708511011209711 4050048049050	Ã§ÅtlÇİÃŸÈĄÏŐÊŋ ÈźlÇİÃŸÈĄÏŐÊŋÈź

5.3.3 Kesimpulan Pengujian Eksperimental

Berdasarkan pengujian eksperimental yang telah dilakukan pada perangkat lunak, dapat disimpulkan bahwa perangkat lunak yang dibangun dapat mengimplementasikan pembagian *password* dan rekonstruksi *password* dengan menggunakan metode *secret sharing* Shamir. Hasil rekonstruksi *password* yang diperoleh pada pengujian eksperimental metode *secret sharing* Shamir membuktikan bahwa perangkat lunak dapat menangani masukan *password* dengan panjang karakter sampai 20 karakter. Sementara pada metode *secret sharing* Blakley, untuk masukan *password* dengan panjang 20, hasil rekonstruksi *password* tidak sesuai dengan masukan *password* yang ditentukan pada awal pengujian. Dapat disimpulkan bahwa perangkat lunak tidak dapat mengatasi implementasi penggunaan metode *secret sharing* Blakley apabila panjang karakter dari masukan *password* lebih dari 15 karakter.

BAB 6

KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai kesimpulan dan saran dari hasil penyusunan skripsi.

6.1 Kesimpulan

Subbab ini akan membahas mengenai kesimpulan dari hasil penyusunan skripsi yang telah dikerjakan. Kesimpulan dari skripsi ini diperoleh setelah melakukan beberapa langkah-langkah pengerjaan skripsi. Berikut adalah langkah-langkah pengerjaan skripsi yang telah dilakukan :

1. Melakukan studi literatur mengenai dasar-dasar kriptografi.
2. Melakukan studi literatur mengenai *secret sharing*, seperti metode-metode *secret sharing* dan cara kerjanya, beserta cara mengimplementasikan metode-metode *secret sharing*.
3. Melakukan perancangan kelas yang akan digunakan untuk mengimplementasikan *secret sharing* Shamir dan *secret sharing* Blakley.
4. Mengimplementasikan hasil perancangan kelas ke dalam bahasa pemrograman *Java*.
5. Melakukan pengujian terhadap perangkat lunak yang telah mengimplementasikan metode *secret sharing* Shamir dan *secret sharing* Blakley.
6. Menarik kesimpulan berdasarkan hasil pengujian.

Langkah awal yang dilakukan pada pengerjaan skripsi ini adalah melakukan studi literatur mengenai dasar-dasar kriptografi. Pada langkah ini dilakukan studi literatur pada buku-buku referensi yang membahas mengenai kriptografi. Setelah langkah ini selesai dilakukan, penulis dapat memahami dasar-dasar dari ilmu kriptografi yang digunakan sebagai dasar ilmu pada proses pengerjaan skripsi ini. Langkah berikutnya adalah mempelajari metode *secret sharing* yang dicapai dengan melakukan studi literatur pada bahan referensi yang membahas mengenai metode-metode *secret sharing* seperti metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Hasil yang diperoleh dari pengerjaan langkah ini adalah penulis dapat memahami cara kerja dari metode *secret sharing* Shamir dan metode *secret sharing* Blakley, sehingga pada langkah selanjutnya dapat dilakukan perancangan kelas-kelas yang diperlukan untuk mengimplementasikan metode *secret sharing* Shamir dan metode *secret sharing* Blakley ke dalam bahasa pemrograman *Java*. Perangkat lunak yang dibangun pada skripsi ini merupakan implementasi dari perancangan kelas-kelas yang telah dilakukan pada langkah sebelumnya.

Setelah melakukan langkah-langkah tersebut, penulis dapat memperoleh kesimpulan berdasarkan hasil pengujian fungsional dan pengujian eksperimental pada perangkat lunak yang dibangun. Pada Bab 5 Implementasi dan Pengujian dapat dilihat rincian dari hasil pengujian fungsional dan pengujian eksperimental yang telah dilakukan. Pada paragraf-paragraf berikutnya akan dijelaskan kesimpulan yang dapat ditarik berdasarkan hasil pengujian tersebut.

Berdasarkan pengujian fungsional yang telah dilakukan, dapat disimpulkan bahwa perangkat lunak yang dibangun telah berhasil mengimplementasikan penggunaan metode *secret sharing* Shamir dan metode *secret sharing* Blakley. Pada Subbab 5.2, dapat dilihat bahwa perangkat lunak berhasil menampung masukan dari pengguna dan melakukan proses pembagian *password* dan penggabungan *share* dengan menggunakan metode *secret sharing* Shamir dan metode *secret sharing* Blakley.

Berdasarkan pengujian yang dilakukan pada Subbab 5.3, dapat disimpulkan bahwa perangkat lunak dapat mengimplementasikan penggunaan metode *secret sharing* Shamir untuk membagikan *password* tanpa membatasi panjang karakter dari masukan *password*. Berdasarkan pengujian tersebut juga dapat disimpulkan bahwa panjang karakter *password* yang ingin dibagikan dengan menggunakan metode *secret sharing* Blakley maksimal 15 karakter. Dengan ini dapat disimpulkan bahwa metode *secret sharing* Shamir lebih baik daripada metode *secret sharing* Blakley dalam penggunaannya untuk membagikan *password* menggunakan perangkat lunak.

6.2 Saran

Berikut adalah beberapa saran untuk pengembangan skripsi lebih lanjut :

- Pada skripsi ini, *share* yang dihasilkan perangkat lunak berupa deretan angka. Pada pengembangan selanjutnya *share* yang dihasilkan dapat dikembangkan menjadi kombinasi dari angka dan huruf alfabet.
- Metode *secret sharing* Blakley yang diimplementasikan pada skripsi ini menggunakan skema *secret sharing* $(3, 3)$. Metode ini dapat dikembangkan sehingga dapat mengimplementasikan pembagian *password* dengan menggunakan skema (k, n) dengan nilai k dan n yang dapat ditentukan dengan bebas.
- Perangkat lunak yang dibangun pada skripsi ini hanya dapat mengimplementasikan metode *secret sharing* Blakley dengan masukan *password* yang memiliki panjang maksimal 15 karakter. Untuk pengembangan selanjutnya, dapat dibangun perangkat lunak yang dapat mengimplementasikan metode *secret sharing* Blakley tanpa membatasi panjang masukan *password*.

DAFTAR REFERENSI

[1] Stamp, M. (2006) *Information Security: Principles and Practice*, 1st edition. John Wiley and Sons, Hoboken.

[2] Forouzan, B. A. (2008) *Cryptography and Network Security*. McGraw-Hill, London.

[3] van Tilborg, H. C. A. (1999) *Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial*. Kluwer Academic Publisher, Eindhoven.

[4] Mao, K. H. (2004) Secret sharing schemes: A cryptographic application of finite projective geometry. Skripsi. National University of Singapore, Singapore.

[5] Munir, R. (2006) *Kriptografi*, 1st edition. Informatika Bandung, Bandung.

[6] Bozkurt, I. N., Kaya, K., Selcuk, A. A., dan Guloglu, A. M. (2008) Threshold cryptography based on blakley secret sharing. *3rd Information Security and Cryptology Conference With International Participation*, Ankara, Turkey, 25-27 December, pp. 183–186. Bildiriler KitabyÌĄ, Turkey.

[7] Ragucci, J. (2008) Shared secret cryptography. http://www.demoivre.org/courses/CIS628/chapter15.pdf. 25 October 2017.

[8] Mackenzie, C. E. (1980) *Coded Character Sets: History and Development*, 1st edition. Addison-Wesley Publishing Company, Boston.

LAMPIRAN A

KODE PROGRAM

Listing A.1: SecretSharing.java

```
1  package secretsharing;
2
3  import java.math.BigInteger;
4  import java.security.SecureRandom;
5  import java.util.Random;
6  import java.util.Scanner;
7
8  /*
9   * To change this license header, choose License Headers in Project Properties.
10  * To change this template file, choose Tools | Templates
11  * and open the template in the editor.
12  */
13 /**
14  *
15  * @author abrahamwahono
16  */
17 public abstract class SecretSharing {
18     int k = 0;
19     int n = 0;
20     String password = new String();
21     SecureRandom random = new SecureRandom();
22     BigInteger p;
23
24     public SecretSharing(int k) {
25         this.k = k;
26         this.n = n;
27         this.password = password;
28     }
29
30     public int getK() {
31         return this.k;
32     }
33
34     public int getN() {
35         return this.n;
36     }
37
38     public BigInteger getP() {
39         return this.p;
40     }
41
42     public void setN(int n) {
43         this.n = n;
44     }
45
46     public void setPassword(String password) {
47         this.password = password;
48     }
49
50
51     public BigInteger toASCIIDecimal(String input) {
52         String result = new String();
53         int charToInt = 0;
54         String temp = new String();
55         for (int i = 0; i < input.length(); i++) {
56             char character = input.charAt(i);
57             charToInt = (int) character;
58             temp = Integer.toString(charToInt);
59             if (charToInt/100 < 1) {
60                 if (i == 0) {
61                     result += temp;
62                 }
63                 else{
64                     result += ("0" + temp);
65                 }
66             }
67             else{
68                 result += temp;
69             }
70         }
71         BigInteger resultToBigInt = new BigInteger(result);
72         return resultToBigInt;
73     }
74
75     public String toASCIICharacter(BigInteger input){
```

```
76          String toString = input.toString();
77          String result = new String();
78          int j = toString.length();
79          int temp = 0;
80
81          if (toString.length() < 3) {
82              result += (char) Integer.parseInt(toString);
83          }
84
85          for (int i = toString.length()-3; i >= 0; i-=3, j-=3) {
86              temp = Integer.parseInt(toString.substring(i, j));
87              if (i == 2) {
88                  temp = Integer.parseInt(toString.substring(i, j));
89                  result = (char) temp + result;
90                  temp = Integer.parseInt(toString.substring(0, 2));
91                  result = (char) temp + result;
92              }
93              else{
94                  result = (char) temp + result;
95              }
96          }
97          return result;
98      }
99
100     public abstract String sharesForEachParticipant();
101
102     public abstract String reconstruct(String[] combinedShare);
103 }
```

Listing A.2: SSShamir.java

```
1  package secretsharing;
2
3  import java.math.BigInteger;
4
5  /*
6   * To change this license header, choose License Headers in Project Properties.
7   * To change this template file, choose Tools | Templates
8   * and open the template in the editor.
9   */
10 /**
11  *
12  * @author abrahamwahono
13  */
14 public class SSShamir extends SecretSharing {
15
16     SharesShamir[] share;
17
18     public SSShamir(int k) {
19         super(k);
20     }
21
22     @Override
23     public String sharesForEachParticipant() {
24         String result = new String();
25         BigInteger a = new BigInteger(this.toASCIIDecimal(password).bitLength(), random);
26         share = new SharesShamir[n];
27         p = BigInteger.probablePrime(this.toASCIIDecimal(password).bitLength()+1, random);
28
29         for (int i = 0; i < n; i++) {
30             share[i] = new SharesShamir();
31             share[i].setX(new BigInteger(String.valueOf(i+1)));
32             share[i].setY(this.toASCIIDecimal(password));
33         }
34
35         for (int i = 0; i < share.length; i++) {
36             for (int j = 0; j < k - 1; j++) {
37                 share[i].setY(share[i].getY().add(a.add(new BigInteger(String.valueOf(j))).multiply(share[i].getX().pow(j+1))));
38             }
39             share[i].setY(share[i].getY().mod(p));
40         }
41         //a ditambahkan dengan j agar nilai a selalu berubah
42
43         String[] pEncrypt = new String[share.length];
44
45         for (int i = 0; i < share.length; i++) {
46             pEncrypt[i] = p.subtract(share[i].getY()).add(share[i].getX()).toString();
47             result += "Share " + (i + 1) + ": " + share[i].getX() + "-" +share[i].getY() + "-" + pEncrypt[i] + "\n";
48
49             //RESET x DAN y
50             share[i].setX(BigInteger.ZERO);
51             share[i].setY(BigInteger.ZERO);
52         }
53
54         //RESET BILANGAN PRIMA DAN PASSWORD
55         p = BigInteger.ZERO;
56         password = null;
57
58         return result;
59     }
60
61     @Override
62     public String reconstruct(String[] combinedShare){
63         BigInteger result = BigInteger.ZERO;
64         String resultToStr;
65         share = new SharesShamir[k];
66         BigInteger numerator;
67         BigInteger denominator;
```

```
        BigInteger start;
        BigInteger next;

        String[] splitShare = combinedShare[0].split("-");
        p = new BigInteger(splitShare[2]).add(new BigInteger(splitShare[1])).subtract(new BigInteger(splitShare[0]));

        for (int i = 0; i < k; i++) {
            splitShare = combinedShare[i].split("-");
            share[i] = new SharesShamir();
            share[i].setX(new BigInteger(splitShare[0]));
            share[i].setY(new BigInteger(splitShare[1]));
        }

        for (int i = 0; i < k; i++) {
            numerator = BigInteger.ONE;
            denominator = BigInteger.ONE;

            for (int j = 0; j < k; j++) {
                if (i != j) {
                    start = share[i].getX();
                    next = share[j].getX();
                    numerator = numerator.multiply((next.negate()).mod(p));
                    denominator = denominator.multiply(start.subtract(next)).mod(p);
                }
            }

            BigInteger value = share[i].getY();
            BigInteger temp = value.multiply(numerator).multiply(denominator.modInverse(p));
            result = p.add(result).add(temp).mod(p);
        }

        resultToStr = this.toASCIICharacter(result);
        return resultToStr;
    }
}
```

Listing A.3: SSBlakley.java

```
package secretsharing;

import Jama.Matrix;
import java.math.BigDecimal;
import java.math.BigInteger;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author abrahamwahono
 */
public class SSBlakley extends SecretSharing {
    SharesBlakley[] share;
    BigDecimal[] a, b;
    BigInteger x, y, z; //Merepresentasikan rahasia/secret

    public SSBlakley(int k) {
        super(k);
    }

    @Override
    public String sharesForEachParticipant() {
        String result = new String();
        share = new SharesBlakley[n];
        a = new BigDecimal[n];
        b = new BigDecimal[n];

        a[0] = new BigDecimal(1);
        b[0] = new BigDecimal(2);
        a[1] = new BigDecimal(2);
        b[1] = new BigDecimal(3);
        a[2] = new BigDecimal(5);
        b[2] = new BigDecimal(7);

        String passwordASCIIToString = toASCIIDecimal(password).toString();
        int counter = 1;

        if (passwordASCIIToString.length() < 3) {
            x = BigInteger.ZERO;
            z = toASCIIDecimal(password).mod(BigInteger.TEN);
            y = toASCIIDecimal(password).subtract(z).divide(BigInteger.TEN);
        }
        else {
            if (passwordASCIIToString.length() % 3 == 0) {
                counter = passwordASCIIToString.length()/3;
            }
            else {
                double c = Math.ceil(passwordASCIIToString.length()/3.0);
                counter = (int) c;
            }

            BigInteger modd = BigInteger.TEN.pow(counter);
            z = toASCIIDecimal(password).mod(modd);
            y = toASCIIDecimal(password).subtract(z).divide(modd).mod(modd);
            x = toASCIIDecimal(password).subtract(z).divide(modd).subtract(y).divide(modd);
```

```
        }

        //Menentukan bilangan acak prima sesuai dengan panjang rahasia
        BigInteger[] arrayXYZ = {x, y, z};
        BigInteger largest = BigInteger.ZERO;

        for (int i = 0; i < arrayXYZ.length; i++) {
            if (arrayXYZ[i].compareTo(largest) == 1) {
                largest = arrayXYZ[i];
            }
        }

        p = BigInteger.probablePrime(largest.bitLength()+1, random);

        BigInteger temp = BigInteger.ONE;
        for (int i = 0; i < share.length; i++) {
           share[i] = new SharesBlakley();
           share[i].setA(a[i]);
           share[i].setB(b[i]);
           share[i].setC(new BigDecimal(z));
           share[i].setC(share[i].getC().subtract(share[i].getA().multiply(new BigDecimal(x))));
           share[i].setC(share[i].getC().subtract(share[i].getB().multiply(new BigDecimal(y))));

           temp = share[i].getC().toBigInteger().mod(p);
           share[i].setC(new BigDecimal(temp));
        }

        String[] pEncrypt = new String[share.length];

        for (int i = 0; i < share.length; i++) {
            pEncrypt[i] = p.add(share[i].getB().toBigInteger()).subtract(share[i].getC().toBigInteger()).toString();
            result += "Share " + (i+1) + ": " + share[i].getC() + "-" + pEncrypt[i] + "-" + String.valueOf(counter+i) + "\n";

            //RESET nilai a, b, c
            share[i].setA(BigDecimal.ZERO);
            share[i].setB(BigDecimal.ZERO);
            share[i].setC(BigDecimal.ZERO);
        }

        //RESET p, password
        p = BigInteger.ZERO;
        password = null;

        return result;
    }

    @Override
    public String reconstruct(String[] combinedShare) {
        String result = new String();
        Matrix A = new Matrix(3, 3);
        Matrix B = new Matrix(3, 1);
        Matrix secret = new Matrix(3, 1);
        share = new SharesBlakley[k];

        share[0] = new SharesBlakley();
        share[0].setA(new BigDecimal(1));
        share[0].setB(new BigDecimal(2));
        share[1] = new SharesBlakley();
        share[1].setA(new BigDecimal(2));
        share[1].setB(new BigDecimal(3));
        share[2] = new SharesBlakley();
        share[2].setA(new BigDecimal(5));
        share[2].setB(new BigDecimal(7));

        String[] splitShare = new String[k];
        splitShare = combinedShare[0].split("-");
        p = new BigInteger(splitShare[1]).add(new BigInteger(splitShare[0])).subtract(share[0].getB().toBigInteger());

        for (int i = 0; i < k; i++) {
            splitShare = combinedShare[i].split("-");
            share[i].setC(new BigDecimal(splitShare[0]));
        }

        //SET MATRIX A
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                switch (j) {
                    case 2:
                        A.set(i, j, -1.0);
                        break;
                    case 1:
                        A.set(i, j, share[i].getB().doubleValue());
                        break;
                    default:
                        A.set(i, j, share[i].getA().doubleValue());
                        break;
                }
            }
        }

        //SET MATRIX B
        for (int i = 0; i < 3; i++) {
            B.set(i, 0, share[i].getC().multiply(new BigDecimal(-1)).doubleValue());
        }

        secret = A.solve(B);

        x = new BigDecimal(Math.round(secret.get(0, 0))).toBigInteger();
        y = new BigDecimal(Math.round(secret.get(1, 0))).toBigInteger();
```

```
        z = new BigDecimal(Math.round(secret.get(2, 0))).toBigInteger();

        x = x.mod(p);
        y = y.mod(p);
        z = z.mod(p);

        int counter = Integer.parseInt(splitShare[2])-2;
        int c;
        String zString = z.toString();
        String yString = y.toString();
        String xString = x.toString();
        if (zString.length() < counter) {
            c = counter - zString.length();
            for (int i = 0; i < c; i++) {
                zString = "0" + zString;
            }
        }
        if (yString.length() < counter) {
            c = counter - yString.length();
            for (int i = 0; i < c; i++) {
                yString = "0" + yString;
            }
        }

        result = xString + yString + zString;
        result = toASCIICharacter(new BigInteger(result));

        return result;
    }
}
```

Listing A.4: SharesShamir.java

```
package secretsharing;

import java.math.BigInteger;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author abrahamwahono
 */
public class SharesShamir {
    BigInteger x;
    BigInteger y;

    public SharesShamir(){
        BigInteger x = BigInteger.ZERO;
        BigInteger y = BigInteger.ZERO;
    }

    public BigInteger getX() {
        return x;
    }

    public void setX(BigInteger x) {
        this.x = x;
    }

    public BigInteger getY() {
        return y;
    }

    public void setY(BigInteger y) {
        this.y = y;
    }
}
```

Listing A.5: SharesBlakley.java

```
package secretsharing;


import java.math.BigDecimal;
import java.math.BigInteger;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author abrahamwahono
 */
public class SharesBlakley {
    BigDecimal a, b, c;

    public SharesBlakley() {
        this.a = BigDecimal.ZERO;
        this.b = BigDecimal.ZERO;
```

```
23          this.c = BigDecimal.ZERO;
24      }
25
26      public BigDecimal getA() {
27          return a;
28      }
29
30      public void setA(BigDecimal a) {
31          this.a = a;
32      }
33
34      public BigDecimal getB() {
35          return b;
36      }
37
38      public void setB(BigDecimal b) {
39          this.b = b;
40      }
41
42      public BigDecimal getC() {
43          return c;
44      }
45
46      public void setC(BigDecimal c) {
47          this.c = c;
48      }
49
50
51  }
```

Listing A.6: GUI.java

```
1   package secretsharing;
2
3   import java.io.IOException;
4   import java.math.BigInteger;
5   import javafx.scene.control.Alert;
6   import javax.swing.JOptionPane;
7
8   /*
9    * To change this license header, choose License Headers in Project Properties.
10   * To change this template file, choose Tools | Templates
11   * and open the template in the editor.
12   */
13  /**
14   *
15   * @author abrahamwahono
16   */
17  public class GUI extends javax.swing.JFrame {
18
19      /**
20       * Creates new form GUIHome
21       */
22      public GUI() {
23          initComponents();
24
25      }
26
27      /**
28       * This method is called from within the constructor to initialize the form.
29       * WARNING: Do NOT modify this code. The content of this method is always
30       * regenerated by the Form Editor.
31       */
32      @SuppressWarnings("unchecked")
33      // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
34      private void initComponents() {
35
36          jLabel16 = new javax.swing.JLabel();
37          jTabbedPane1 = new javax.swing.JTabbedPane();
38          jPanel1 = new javax.swing.JPanel();
39          jLabel1 = new javax.swing.JLabel();
40          jLabel2 = new javax.swing.JLabel();
41          jLabel3 = new javax.swing.JLabel();
42          jLabel4 = new javax.swing.JLabel();
43          jLabel5 = new javax.swing.JLabel();
44          jLabel6 = new javax.swing.JLabel();
45          jLabel7 = new javax.swing.JLabel();
46          jLabel8 = new javax.swing.JLabel();
47          comboBoxMetode1 = new javax.swing.JComboBox<>();
48          textFieldThreshold1 = new javax.swing.JTextField();
49          textFieldPartisipan = new javax.swing.JTextField();
50          passwordField = new javax.swing.JPasswordField();
51          buttonBagiPassword = new javax.swing.JButton();
52          jLabel9 = new javax.swing.JLabel();
53          jScrollPane1 = new javax.swing.JScrollPane();
54          textAreaShares = new javax.swing.JTextArea();
55          buttonReset1 = new javax.swing.JButton();
56          jPanel2 = new javax.swing.JPanel();
57          jLabel10 = new javax.swing.JLabel();
58          jScrollPane2 = new javax.swing.JScrollPane();
59          textAreaGabungShare = new javax.swing.JTextArea();
60          buttonGabungShare = new javax.swing.JButton();
61          jLabel11 = new javax.swing.JLabel();
62          textFieldPasswordDiperoleh = new javax.swing.JTextField();
63          jLabel12 = new javax.swing.JLabel();
64          jLabel13 = new javax.swing.JLabel();
65          jLabel14 = new javax.swing.JLabel();
66          jLabel15 = new javax.swing.JLabel();
```

```
67          comboBoxMetode2 = new javax.swing.JComboBox<>();
68          textFieldThreshold2 = new javax.swing.JTextField();
69          buttonReset2 = new javax.swing.JButton();
70
71          jLabel16.setText("jLabel16");
72
73          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
74          setTitle("Password Distributor");
75
76          jLabel1.setText("Metode Secret Sharing");
77
78          jLabel2.setText("Threshold");
79
80          jLabel3.setText("Banyak Partisipan");
81
82          jLabel4.setText("Password");
83
84          jLabel5.setText(":");
85
86          jLabel6.setText(":");
87
88          jLabel7.setText(":");
89
90          jLabel8.setText(":");
91
92          comboBoxMetode1.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Shamir", "Blakley" }));
93          comboBoxMetode1.addActionListener(new java.awt.event.ActionListener() {
94              public void actionPerformed(java.awt.event.ActionEvent evt) {
95                  comboBoxMetode1ActionPerformed(evt);
96              }
97          });
98
99          textFieldThreshold1.addFocusListener(new java.awt.event.FocusAdapter() {
100             public void focusLost(java.awt.event.FocusEvent evt) {
101                 textFieldThreshold1FocusLost(evt);
102             }
103         });
104
105         textFieldPartisipan.addFocusListener(new java.awt.event.FocusAdapter() {
106             public void focusLost(java.awt.event.FocusEvent evt) {
107                 textFieldPartisipanFocusLost(evt);
108             }
109         });
110
111         passwordField.addFocusListener(new java.awt.event.FocusAdapter() {
112             public void focusLost(java.awt.event.FocusEvent evt) {
113                 passwordFieldFocusLost(evt);
114             }
115         });
116
117         buttonBagiPassword.setText("Bagikan Password");
118         buttonBagiPassword.addMouseListener(new java.awt.event.MouseAdapter() {
119             public void mouseClicked(java.awt.event.MouseEvent evt) {
120                 buttonBagiPasswordMouseClicked(evt);
121             }
122         });
123         buttonBagiPassword.addKeyListener(new java.awt.event.KeyAdapter() {
124             public void keyPressed(java.awt.event.KeyEvent evt) {
125                 buttonBagiPasswordKeyPressed(evt);
126             }
127         });
128
129         jLabel9.setText("Share untuk tiap partisipan :");
130
131         textAreaShares.setEditable(false);
132         textAreaShares.setColumns(20);
133         textAreaShares.setRows(5);
134         jScrollPane1.setViewportView(textAreaShares);
135
136         buttonReset1.setText("Reset");
137         buttonReset1.addMouseListener(new java.awt.event.MouseAdapter() {
138             public void mouseClicked(java.awt.event.MouseEvent evt) {
139                 buttonReset1MouseClicked(evt);
140             }
141         });
142         buttonReset1.addKeyListener(new java.awt.event.KeyAdapter() {
143             public void keyPressed(java.awt.event.KeyEvent evt) {
144                 buttonReset1KeyPressed(evt);
145             }
146         });
147
148         javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
149         jPanel1.setLayout(jPanel1Layout);
150         jPanel1Layout.setHorizontalGroup(
151             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
152             .addGroup(jPanel1Layout.createSequentialGroup()
153                 .addGap(16, 16, 16)
154                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
155                     .addGroup(jPanel1Layout.createSequentialGroup()
156                         .addComponent(jLabel9)
157                         .addGap(0, 396, Short.MAX_VALUE))
158                     .addGroup(jPanel1Layout.createSequentialGroup()
159                         .addComponent(jScrollPane1)
160                         .addContainerGap())))
161             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()
162                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
163                 .addComponent(buttonReset1)
164                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
165                 .addComponent(buttonBagiPassword)
```

```
166                 .addGap(70, 70, 70))
167             .addGroup(jPanel1Layout.createSequentialGroup()
168                 .addGap(17, 17, 17)
169                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
170                     .addComponent(jLabel4)
171                     .addComponent(jLabel1)
172                     .addComponent(jLabel3)
173                     .addComponent(jLabel2))
174                 .addGap(71, 71, 71)
175                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
176                     .addGroup(jPanel1Layout.createSequentialGroup()
177                         .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 16, javax.swing.GroupLayout.PREFERRED_SIZE)
178                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
179                         .addComponent(textFieldPartisipan))
180                     .addGroup(jPanel1Layout.createSequentialGroup()
181                         .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 16, javax.swing.GroupLayout.PREFERRED_SIZE)
182                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
183                         .addComponent(passwordField))
184                     .addGroup(jPanel1Layout.createSequentialGroup()
185                         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
186                             .addComponent(jLabel5)
187                             .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 16, javax.swing.GroupLayout.
                                    PREFERRED_SIZE))
188                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
189                         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190                             .addComponent(textFieldThreshold1)
191                             .addComponent(comboBoxMetode1, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
192                 .addContainerGap())
193         );
194         jPanel1Layout.setVerticalGroup(
195             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
196             .addGroup(jPanel1Layout.createSequentialGroup()
197                 .addContainerGap()
198                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
199                     .addComponent(jLabel1)
200                     .addComponent(jLabel5)
201                     .addComponent(comboBoxMetode1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
202                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
203                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
204                     .addComponent(jLabel2)
205                     .addComponent(jLabel6)
206                     .addComponent(textFieldThreshold1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
                            DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
207                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
208                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
209                     .addComponent(jLabel3)
210                     .addComponent(jLabel7)
211                     .addComponent(textFieldPartisipan, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
                            DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
212                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
213                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
214                     .addComponent(jLabel4)
215                     .addComponent(jLabel8)
216                     .addComponent(passwordField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
217                 .addGap(29, 29, 29)
218                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
219                     .addComponent(buttonBagiPassword)
220                     .addComponent(buttonReset1))
221                 .addGap(14, 14, 14)
222                 .addComponent(jLabel9)
223                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
224                 .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 175, Short.MAX_VALUE)
225                 .addGap(12, 12, 12))
226         );
227 
228         jTabbedPane1.addTab("Pembagian Password", jPanel1);
229 
230         jLabel10.setText("Share yang digabungkan :");
231 
232         textAreaGabungShare.setColumns(20);
233         textAreaGabungShare.setRows(5);
234         textAreaGabungShare.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
235         jScrollPane2.setViewportView(textAreaGabungShare);
236 
237         buttonGabungShare.setText("Gabungkan Share");
238         buttonGabungShare.addMouseListener(new java.awt.event.MouseAdapter() {
239             public void mouseClicked(java.awt.event.MouseEvent evt) {
240                 buttonGabungShareMouseClicked(evt);
241             }
242         });
243         buttonGabungShare.addKeyListener(new java.awt.event.KeyAdapter() {
244             public void keyPressed(java.awt.event.KeyEvent evt) {
245                 buttonGabungShareKeyPressed(evt);
246             }
247         });
248 
249         jLabel11.setText("Password yang diperoleh :");
250 
251         textFieldPasswordDiperoleh.setEditable(false);
252 
253         jLabel12.setText("Metode Secret Sharing");
254 
255         jLabel13.setText("Threshold");
256 
257         jLabel14.setText(":");
258 
259         jLabel15.setText(":");
```

```
260
261         comboBoxMetode2.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Shamir", "Blakley" }));
262         comboBoxMetode2.addActionListener(new java.awt.event.ActionListener() {
263             public void actionPerformed(java.awt.event.ActionEvent evt) {
264                 comboBoxMetode2ActionPerformed(evt);
265             }
266         });
267
268         textFieldThreshold2.addFocusListener(new java.awt.event.FocusAdapter() {
269             public void focusLost(java.awt.event.FocusEvent evt) {
270                 textFieldThreshold2FocusLost(evt);
271             }
272         });
273
274         buttonReset2.setText("Reset");
275         buttonReset2.addMouseListener(new java.awt.event.MouseAdapter() {
276             public void mouseClicked(java.awt.event.MouseEvent evt) {
277                 buttonReset2MouseClicked(evt);
278             }
279         });
280         buttonReset2.addKeyListener(new java.awt.event.KeyAdapter() {
281             public void keyPressed(java.awt.event.KeyEvent evt) {
282                 buttonReset2KeyPressed(evt);
283             }
284         });
285
286         javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
287         jPanel2.setLayout(jPanel2Layout);
288         jPanel2Layout.setHorizontalGroup(
289             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
290             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
291                 .addContainerGap(179, Short.MAX_VALUE)
292                 .addComponent(buttonReset2)
293                 .addGap(26, 26, 26)
294                 .addComponent(buttonGabungShare)
295                 .addGap(154, 154, 154))
296             .addGroup(jPanel2Layout.createSequentialGroup()
297                 .addGap(17, 17, 17)
298                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
299                     .addComponent(jLabel11, javax.swing.GroupLayout.PREFERRED_SIZE, 172, javax.swing.GroupLayout.PREFERRED_SIZE)
300                     .addComponent(jLabel10)
301                     .addComponent(textFieldPasswordDiperoleh)
302                     .addComponent(jScrollPane2)
303                     .addGroup(jPanel2Layout.createSequentialGroup()
304                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
305                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
306                                 .addComponent(jLabel13)
307                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.
                                         DEFAULT_SIZE, Short.MAX_VALUE)
308                                 .addComponent(jLabel15, javax.swing.GroupLayout.PREFERRED_SIZE, 16, javax.swing.GroupLayout.
                                         PREFERRED_SIZE))
309                             .addGroup(jPanel2Layout.createSequentialGroup()
310                                 .addComponent(jLabel12)
311                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.
                                         DEFAULT_SIZE, Short.MAX_VALUE)
312                                 .addComponent(jLabel14, javax.swing.GroupLayout.PREFERRED_SIZE, 16, javax.swing.GroupLayout.
                                         PREFERRED_SIZE)))
313                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
314                         .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
315                             .addComponent(textFieldThreshold2)
316                             .addComponent(comboBoxMetode2, 0, 325, Short.MAX_VALUE))))
317                 .addGap(16, 16, 16))
318         );
319         jPanel2Layout.setVerticalGroup(
320             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
321             .addGroup(jPanel2Layout.createSequentialGroup()
322                 .addContainerGap()
323                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
324                     .addComponent(jLabel12)
325                     .addComponent(jLabel14)
326                     .addComponent(comboBoxMetode2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                             javax.swing.GroupLayout.PREFERRED_SIZE))
327                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
328                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
329                     .addComponent(jLabel13)
330                     .addComponent(jLabel15)
331                     .addComponent(textFieldThreshold2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
                             DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
332                 .addGap(24, 24, 24)
333                 .addComponent(jLabel10)
334                 .addGap(18, 18, 18)
335                 .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 184, Short.MAX_VALUE)
336                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
337                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
338                     .addComponent(buttonReset2, javax.swing.GroupLayout.Alignment.TRAILING)
339                     .addComponent(buttonGabungShare))
340                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
341                 .addComponent(jLabel11)
342                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
343                 .addComponent(textFieldPasswordDiperoleh, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
                         DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
344                 .addGap(20, 20, 20))
345         );
346
347         jTabbedPane1.addTab("Penggabungan Share", jPanel2);
348
349         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
350         getContentPane().setLayout(layout);
351         layout.setHorizontalGroup(
```

```
352             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
353             .addComponent(jTabbedPane1)
354         );
355         layout.setVerticalGroup(
356             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
357             .addComponent(jTabbedPane1)
358         );
359 
360         pack();
361     }// </editor-fold>//GEN-END:initComponents
362 
363     private void buttonBagiPasswordMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_buttonBagiPasswordMouseClicked
364         String getK = textFieldThreshold1.getText();
365         String getN = textFieldPartisipan.getText();
366         String getPass = passwordField.getText();
367 
368         try {
369             if (getK.isEmpty()) {
370                 JOptionPane.showMessageDialog(null, "Masukkan threshold!", "Error", JOptionPane.ERROR_MESSAGE);
371             }
372             else{
373                 for (int i = 0; i < getK.length(); i++) {
374                     if (Character.isAlphabetic(getK.charAt(i))) {
375                         JOptionPane.showMessageDialog(null, "Masukan threshold harus bilangan bulat!", "Error", JOptionPane.
                                ERROR_MESSAGE);
376                         break;
377                     }
378                 }
379             }
380 
381             if (getN.isEmpty()) {
382                 JOptionPane.showMessageDialog(null, "Masukkan banyak partisipan!", "Error", JOptionPane.ERROR_MESSAGE);
383             }
384             else{
385                 for (int i = 0; i < getN.length(); i++) {
386                     if (Character.isAlphabetic(getN.charAt(i))) {
387                         JOptionPane.showMessageDialog(null, "Masukan partisipan harus bilangan bulat!", "Error", JOptionPane.
                                ERROR_MESSAGE);
388                         break;
389                     }
390                 }
391             }
392 
393             if (getPass.isEmpty()) {
394                 JOptionPane.showMessageDialog(null, "Masukkan password!", "Error", JOptionPane.ERROR_MESSAGE);
395             }
396 
397             int k = Integer.parseInt(getK);
398             int n = Integer.parseInt(getN);
399             String password = getPass;
400 
401             //SHAMIR
402             if (comboBoxMetode1.getSelectedIndex() == 0) {
403                 if (n < k) {
404                     JOptionPane.showMessageDialog(null, "Banyak partisipan harus lebih dari atau sama dengan threshold.", "Error",
                            JOptionPane.ERROR_MESSAGE);
405                     textFieldPartisipan.setText(getK);
406                 }
407                 else{
408                     SSShamir shamir = new SSShamir(k);
409                     shamir.setN(n);
410                     shamir.setPassword(password);
411                     textAreaShares.setText(shamir.sharesForEachParticipant());
412                 }
413             }
414 
415             //BLAKLEY
416             else {
417                 if (getPass.length() > 15) {
418                     JOptionPane.showMessageDialog(null, "Password maksimal 15 karakter.", "Error", JOptionPane.ERROR_MESSAGE);
419                 }
420                 else {
421                     SSBlakley blakley = new SSBlakley(k);
422                     blakley.setN(n);
423                     blakley.setPassword(password);
424                     textAreaShares.setText(blakley.sharesForEachParticipant());
425                 }
426             }
427         }
428         catch(NumberFormatException e){
429             System.out.println("error");
430         }
431     }//GEN-LAST:event_buttonBagiPasswordMouseClicked
432 
433     private void buttonGabungShareMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_buttonGabungShareMouseClicked
434         String getK = textFieldThreshold2.getText();
435         String getTextArea = textAreaGabungShare.getText();
436 
437         try {
438             if (getK.isEmpty()) {
439                 JOptionPane.showMessageDialog(null, "Masukkan threshold!", "Error", JOptionPane.ERROR_MESSAGE);
440             }
441 
442             if (getTextArea.isEmpty()) {
443                 JOptionPane.showMessageDialog(null, "Masukkan share yang ingin digabungkan!", "Error", JOptionPane.ERROR_MESSAGE);
444             }
445 
446             int k = Integer.parseInt(getK);
447 
```

```
448             //SHAMIR
449             if (this.comboBoxMetode2.getSelectedIndex() == 0) {
450                 SSShamir shamir = new SSShamir(k);
451                 String[] splitInput = textAreaGabungShare.getText().split("\n");
452                 String[] combinedShare = new String[k];
453                 for (int i = 0; i < k; i++) {
454                     combinedShare[i] = splitInput[i];
455                 }
456                 textFieldPasswordDiperoleh.setText(shamir.reconstruct(combinedShare));
457             }
458 
459             //BLAKLEY
460             else {
461                 SSBlakley blakley = new SSBlakley(k);
462                 String[] splitInput = textAreaGabungShare.getText().split("\n");
463                 String[] combinedShare = new String[k];
464                 for (int i = 0; i < k; i++) {
465                     combinedShare[i] = splitInput[i];
466                 }
467                 textFieldPasswordDiperoleh.setText(blakley.reconstruct(combinedShare));
468             }
469         }
470         catch(NumberFormatException e){
471             JOptionPane.showMessageDialog(null, "Masukan threshold harus bilangan bulat!", "Error", JOptionPane.ERROR_MESSAGE);
472 
473         }
474         catch(ArrayIndexOutOfBoundsException e){
475             JOptionPane.showMessageDialog(null, "Password gagal diperoleh.", "Error", JOptionPane.ERROR_MESSAGE);
476         }
477     }//GEN-LAST:event_buttonGabungShareMouseClicked
478 
479     private void buttonReset1MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_buttonReset1MouseClicked
480         comboBoxMetode1.setSelectedIndex(0);
481 
482         textFieldThreshold1.setText(null);
483         textFieldThreshold1.setEditable(true);
484 
485         textFieldPartisipan.setText(null);
486         textFieldPartisipan.setEditable(true);
487 
488         passwordField.setText(null);
489 
490         textAreaShares.setText(null);
491     }//GEN-LAST:event_buttonReset1MouseClicked
492 
493     private void buttonReset2MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_buttonReset2MouseClicked
494         comboBoxMetode2.setSelectedIndex(0);
495 
496         textFieldThreshold2.setText(null);
497         textFieldThreshold2.setEditable(true);
498 
499         textAreaGabungShare.setText(null);
500 
501         textFieldPasswordDiperoleh.setText(null);
502     }//GEN-LAST:event_buttonReset2MouseClicked
503 
504     private void comboBoxMetode1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_comboBoxMetode1ActionPerformed
505         if (comboBoxMetode1.getSelectedIndex() == 1) {
506             textFieldThreshold1.setText("3");
507             textFieldThreshold1.setEditable(false);
508 
509             textFieldPartisipan.setText("3");
510             textFieldPartisipan.setEditable(false);
511 
512             JOptionPane.showMessageDialog(null, "Untuk metode Blakley panjang password maksimal 15 karakter", "Warning",
                    JOptionPane.WARNING_MESSAGE);
513         }
514     }//GEN-LAST:event_comboBoxMetode1ActionPerformed
515 
516     private void comboBoxMetode2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_comboBoxMetode2ActionPerformed
517         if (comboBoxMetode2.getSelectedIndex() == 1) {
518             textFieldThreshold2.setText("3");
519             textFieldThreshold2.setEditable(false);
520             JOptionPane.showMessageDialog(null, "Masukkan share mulai dari share ke-1 sampai ke-3 secara berurutan!", "Warning",
                    JOptionPane.WARNING_MESSAGE);
521         }
522     }//GEN-LAST:event_comboBoxMetode2ActionPerformed
523 
524     private void textFieldThreshold1FocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_textFieldThreshold1FocusLost
525         String getK = textFieldThreshold1.getText();
526         for (int i = 0; i < getK.length(); i++) {
527             if (Character.isAlphabetic(getK.charAt(i))) {
528                 JOptionPane.showMessageDialog(null, "Masukan threshold harus bilangan bulat!", "Warning", JOptionPane.
                        WARNING_MESSAGE);
529                 break;
530             }
531         }
532         if (Integer.parseInt(getK) == 0) {
533             JOptionPane.showMessageDialog(null, "Masukan threshold tidak dapat bernilai 0", "Warning", JOptionPane.WARNING_MESSAGE
                    );
534             textFieldThreshold1.setText("1");
535         }
536     }//GEN-LAST:event_textFieldThreshold1FocusLost
537 
538     private void textFieldThreshold2FocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_textFieldThreshold2FocusLost
539         String getK = textFieldThreshold2.getText();
540         for (int i = 0; i < getK.length(); i++) {
541             if (Character.isAlphabetic(getK.charAt(i))) {
542                 JOptionPane.showMessageDialog(null, "Masukan threshold harus bilangan bulat!", "Warning", JOptionPane.
```

```
                    WARNING_MESSAGE);
543                 break;
544             }
545         }
546         if (Integer.parseInt(getK) == 0) {
547             JOptionPane.showMessageDialog(null, "Masukan threshold tidak dapat bernilai 0", "Warning", JOptionPane.WARNING_MESSAGE
                    );
548             textFieldThreshold2.setText("1");
549         }
550     }//GEN-LAST:event_textFieldThreshold2FocusLost
551
552     private void textFieldPartisipanFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_textFieldPartisipanFocusLost
553         String getK = textFieldThreshold1.getText();
554         String getN = textFieldPartisipan.getText();
555         for (int i = 0; i < getN.length(); i++) {
556             if (Character.isAlphabetic(getN.charAt(i))) {
557                 JOptionPane.showMessageDialog(null, "Masukan partisipan harus bilangan bulat!", "Warning", JOptionPane.
                        WARNING_MESSAGE);
558                 break;
559             }
560         }
561         if (Integer.parseInt(getN) == 0) {
562             JOptionPane.showMessageDialog(null, "Banyak partisipan minimal 1", "Warning", JOptionPane.WARNING_MESSAGE);
563         }
564         if (Integer.parseInt(getN) < Integer.parseInt(getK)) {
565             JOptionPane.showMessageDialog(null, "Banyak partisipan harus lebih dari atau sama dengan threshold", "Warning",
                    JOptionPane.WARNING_MESSAGE);
566             textFieldPartisipan.setText(getK);
567         }
568     }//GEN-LAST:event_textFieldPartisipanFocusLost
569
570     private void passwordFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_passwordFieldFocusLost
571         String getPass = passwordField.getText();
572         if (comboBoxMetode1.getSelectedIndex() == 1) {
573             if (getPass.length() > 15) {
574                 JOptionPane.showMessageDialog(null, "Password maksimal 15 karakter", "Warning", JOptionPane.WARNING_MESSAGE);
575             }
576         }
577     }//GEN-LAST:event_passwordFieldFocusLost
578
579     private void buttonBagiPasswordKeyPressed(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_buttonBagiPasswordKeyPressed
580         String getK = textFieldThreshold1.getText();
581         String getN = textFieldPartisipan.getText();
582         String getPass = passwordField.getText();
583
584         try {
585             if (getK.isEmpty()) {
586                 JOptionPane.showMessageDialog(null, "Masukkan threshold!", "Error", JOptionPane.ERROR_MESSAGE);
587             }
588             else{
589                 for (int i = 0; i < getK.length(); i++) {
590                     if (Character.isAlphabetic(getK.charAt(i))) {
591                         JOptionPane.showMessageDialog(null, "Masukan threshold harus bilangan bulat!", "Error", JOptionPane.
                                ERROR_MESSAGE);
592                         break;
593                     }
594                 }
595             }
596
597             if (getN.isEmpty()) {
598                 JOptionPane.showMessageDialog(null, "Masukkan banyak partisipan!", "Error", JOptionPane.ERROR_MESSAGE);
599             }
600             else{
601                 for (int i = 0; i < getN.length(); i++) {
602                     if (Character.isAlphabetic(getN.charAt(i))) {
603                         JOptionPane.showMessageDialog(null, "Masukan partisipan harus bilangan bulat!", "Error", JOptionPane.
                                ERROR_MESSAGE);
604                         break;
605                     }
606                 }
607             }
608
609             if (getPass.isEmpty()) {
610                 JOptionPane.showMessageDialog(null, "Masukkan password!", "Error", JOptionPane.ERROR_MESSAGE);
611             }
612
613             int k = Integer.parseInt(getK);
614             int n = Integer.parseInt(getN);
615             String password = getPass;
616
617             //SHAMIR
618             if (comboBoxMetode1.getSelectedIndex() == 0) {
619                 if (n < k) {
620                     JOptionPane.showMessageDialog(null, "Banyak partisipan harus lebih dari atau sama dengan threshold.", "Error",
                            JOptionPane.ERROR_MESSAGE);
621                     textFieldPartisipan.setText(getK);
622                 }
623                 else{
624                     SSShamir shamir = new SSShamir(k);
625                     shamir.setN(n);
626                     shamir.setPassword(password);
627                     textAreaShares.setText(shamir.sharesForEachParticipant());
628                 }
629             }
630
631             //BLAKLEY
632             else {
633                 if (getPass.length() > 15) {
634                     JOptionPane.showMessageDialog(null, "Password maksimal 15 karakter.", "Error", JOptionPane.ERROR_MESSAGE);
```

```
635                 }
636                 else {
637                     SSBlakley blakley = new SSBlakley(k);
638                     blakley.setN(n);
639                     blakley.setPassword(password);
640                     textAreaShares.setText(blakley.sharesForEachParticipant());
641                 }
642             }
643         }
644         catch(NumberFormatException e){
645             System.out.println("error");
646         }
647     }//GEN-LAST:event_buttonBagiPasswordKeyPressed
648 
649     private void buttonGabungShareKeyPressed(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_buttonGabungShareKeyPressed
650 
651     }//GEN-LAST:event_buttonGabungShareKeyPressed
652 
653     private void buttonReset1KeyPressed(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_buttonReset1KeyPressed
654         comboBoxMetode1.setSelectedIndex(0);
655 
656         textFieldThreshold1.setText(null);
657         textFieldThreshold1.setEditable(true);
658 
659         textFieldPartisipan.setText(null);
660         textFieldPartisipan.setEditable(true);
661 
662         passwordField.setText(null);
663 
664         textAreaShares.setText(null);
665     }//GEN-LAST:event_buttonReset1KeyPressed
666 
667     private void buttonReset2KeyPressed(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_buttonReset2KeyPressed
668 
669     }//GEN-LAST:event_buttonReset2KeyPressed
670 
671     /**
672      * @param args the command line arguments
673      */
674     public static void main(String args[]) {
675         /* Set the Nimbus look and feel */
676         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
677         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
678          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
679          */
680         try {
681             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
682                 if ("Nimbus".equals(info.getName())) {
683                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
684                     break;
685                 }
686             }
687         } catch (ClassNotFoundException ex) {
688             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
689         } catch (InstantiationException ex) {
690             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
691         } catch (IllegalAccessException ex) {
692             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
693         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
694             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
695         }
696         //</editor-fold>
697         //</editor-fold>
698 
699         /* Create and display the form */
700         java.awt.EventQueue.invokeLater(new Runnable() {
701             public void run() {
702                 new GUI().setVisible(true);
703             }
704         });
705     }
706 
707     // Variables declaration - do not modify//GEN-BEGIN:variables
708     private javax.swing.JButton buttonBagiPassword;
709     private javax.swing.JButton buttonGabungShare;
710     private javax.swing.JButton buttonReset1;
711     private javax.swing.JButton buttonReset2;
712     private javax.swing.JComboBox<String> comboBoxMetode1;
713     private javax.swing.JComboBox<String> comboBoxMetode2;
714     private javax.swing.JLabel jLabel1;
715     private javax.swing.JLabel jLabel10;
716     private javax.swing.JLabel jLabel11;
717     private javax.swing.JLabel jLabel12;
718     private javax.swing.JLabel jLabel13;
719     private javax.swing.JLabel jLabel14;
720     private javax.swing.JLabel jLabel15;
721     private javax.swing.JLabel jLabel16;
722     private javax.swing.JLabel jLabel2;
723     private javax.swing.JLabel jLabel3;
724     private javax.swing.JLabel jLabel4;
725     private javax.swing.JLabel jLabel5;
726     private javax.swing.JLabel jLabel6;
727     private javax.swing.JLabel jLabel7;
728     private javax.swing.JLabel jLabel8;
729     private javax.swing.JLabel jLabel9;
730     private javax.swing.JPanel jPanel1;
731     private javax.swing.JPanel jPanel2;
732     private javax.swing.JScrollPane jScrollPane1;
733     private javax.swing.JScrollPane jScrollPane2;
```

```
734     private javax.swing.JTabbedPane jTabbedPane1;
735     private javax.swing.JPasswordField passwordField;
736     private javax.swing.JTextArea textAreaGabungShare;
737     private javax.swing.JTextArea textAreaShares;
738     private javax.swing.JTextField textFieldPartisipan;
739     private javax.swing.JTextField textFieldPasswordDiperoleh;
740     private javax.swing.JTextField textFieldThreshold1;
741     private javax.swing.JTextField textFieldThreshold2;
742     // End of variables declaration//GEN-END:variables
743 }
```