

SKRIPSI

PERLINDUNGAN *PASSWORD* DENGAN ENTROPI
PERSONAL



SAMUEL CHRISTIAN

NPM: 2011730002

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2015

UNDERGRADUATE THESIS

PROTECTING PASSWORD WITH PERSONAL ENTROPY



SAMUEL CHRISTIAN

NPM: 2011730002

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2015**

DAFTAR ISI

DAFTAR ISI	v
DAFTAR GAMBAR	vii
DAFTAR TABEL	ix
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan	1
1.4 Batasan Masalah	1
1.5 Metodologi Penelitian	1
1.6 Sistematika Pembahasan	2
2 DASAR TEORI	3
2.1 Kriptografi	3
2.1.1 Sejarah Kriptografi	3
2.1.2 Pengertian Kriptografi	3
2.2 Kerahasiaan (<i>Confidentiality</i>)	4
2.3 <i>Data Encryption Standard</i> (DES)	4
2.3.1 Sejarah	4
2.3.2 Struktur DES	5
2.3.3 Permutasi Awal	5
2.3.4 Pembangunan Kunci Putaran	6
2.3.5 Putaran	7
2.3.6 Permutasi Akhir	11
2.4 Fungsi <i>Hash</i>	12
2.5 <i>Secure Hashing Algorithm</i> 512 (SHA-512)	12
2.5.1 <i>Message Padding</i>	12
2.5.2 Inisialisasi Konstanta Awal	13
2.5.3 Ekspansi Blok Message	13
2.5.4 Fungsi Kompresi dan Putaran	14
2.6 Otentikasi	17
2.6.1 <i>Password</i>	18
2.7 Eliminasi Gauss-Jordan	20
2.7.1 Proses Reduksi Matriks	21
2.7.2 Proses Substitusi Balik	22
2.8 <i>Secret Sharing</i> Shamir	22
2.8.1 Sejarah Singkat	22
2.8.2 Pembahasan <i>Secret Sharing</i> Shamir	23
2.9 Probabilitas	24
2.9.1 Distribusi Binom	24
2.10 Entropi	25

2.10.1	Sejarah Singkat	25
2.10.2	Pembahasan	25
3	ANALISIS	27
3.1	Studi Kasus	27
3.1.1	Pengenalan Kasus	27
3.1.2	Proses Penyimpanan <i>Password</i>	27
3.1.3	Proses Pengembalian <i>Password</i>	31
3.2	Pemilihan n dan k	35
3.2.1	Pemilihan k	35
3.2.2	Pemilihan n	35
3.3	Perancangan Perangkat Lunak	35
3.3.1	Alur Proses	36
3.3.2	Diagram <i>Use Case</i>	37
3.3.3	Diagram Aktivitas	38
3.3.4	Diagram Kelas	39
3.3.5	Arsitektur Perangkat Lunak	40
4	PERANCANGAN	43
4.1	Diagram Kelas Rinci	43
4.2	Deskripsi Kelas dan Fungsi	43
4.2.1	Kelas <i>SHA512</i>	43
4.2.2	Kelas <i>Function</i>	44
4.2.3	Kelas <i>EquationSolver</i>	45
4.2.4	Kelas <i>SecretSharing</i>	46
4.2.5	Kelas <i>DESEncryption</i>	46
4.2.6	Kelas <i>DESDecryption</i>	46
4.2.7	Kelas <i>DataReader</i>	46
4.2.8	Kelas <i>DataWriter</i>	47
4.3	Perancangan Antarmuka	47
5	IMPLEMENTASI DAN PENGUJIAN	51
5.1	Implementasi Perangkat Lunak	51
5.1.1	Tampilan Antarmuka Perangkat Lunak	51
5.2	Pengujian Perangkat Lunak	55
5.2.1	Metode Pengujian	55
5.2.2	Hasil Pengujian Fungsional	55
5.2.3	Hasil Pengujian Survei	57
5.2.4	Analisis Pengujian	60
5.2.5	Kesimpulan Pengujian	61
6	KESIMPULAN DAN SARAN	63
6.1	Kesimpulan	63
6.2	Saran	63
	DAFTAR REFERENSI	65
	A THE PROGRAM	67

DAFTAR GAMBAR

2.1	Proses Enkripsi	5
2.2	Proses Permutasi	6
2.3	Putaran dalam DES	7
2.4	Jaringan Feistel	8
2.5	Struktur Putaran dalam SHA-512	15
2.6	Masukan dan Keluaran dalam 1 Putaran SHA-512	15
2.7	Fungsi Khusus dalam 1 Putaran SHA-512	16
2.8	Proses Keseluruhan dari SHA-512	17
2.9	<i>Username</i> dan <i>Password</i>	19
2.10	<i>Password hashing</i>	19
2.11	<i>Password salting</i>	20
3.1	Proses pembangunan <i>share</i> dari <i>password</i>	36
3.2	Proses pembangunan kembali atau rekonstruksi <i>password</i>	37
3.3	Diagram <i>use case</i> dari perangkat lunak	37
3.4	Diagram aktivitas untuk menyimpan <i>password</i>	38
3.5	Diagram aktivitas untuk mengembalikan <i>password</i>	39
3.6	Diagram kelas <i>engine</i>	40
3.7	Arsitektur perangkat lunak	41
4.2	Kelas SHA512	43
4.3	Kelas <i>Function</i>	45
4.4	Kelas <i>EquationSolver</i>	45
4.5	Kelas <i>DataReader</i>	46
4.6	Kelas <i>DataWriter</i>	47
4.7	Perancangan Tampilan Awal	47
4.8	Perancangan Tampilan Menyimpan <i>Password</i>	48
4.9	Perancangan Tampilan Mengembalikan <i>Password</i>	48
4.10	Perancangan Tampilan Mengembalikan <i>Password</i>	49
4.1	Diagram Kelas Rinci	50
5.1	Tampilan antarmuka awal	51
5.2	Tampilan antarmuka untuk menyimpan <i>password</i>	52
5.3	Tampilan antarmuka untuk menyimpan <i>password</i>	52
5.4	Tampilan antarmuka untuk menyimpan <i>password</i>	53
5.5	Tampilan antarmuka untuk menyimpan <i>password</i>	53
5.6	Tampilan antarmuka untuk mengembalikan <i>password</i>	54
5.7	Tampilan antarmuka untuk mengembalikan <i>password</i>	54
5.8	Tampilan antarmuka untuk mengembalikan <i>password</i>	55
5.9	Langkah menyimpan <i>password</i>	56
5.10	Langkah menjawab pertanyaan keamanan	56
5.11	<i>Password</i> berhasil dikembalikan	56
5.12	Langkah menjawab pertanyaan keamanan	57

5.13 <i>Password</i> tidak berhasil dikembalikan	57
5.14 Pengujian survei kasus 1	58
5.15 Pengujian survei kasus 1	59
5.16 Pengujian survei kasus 1	60

DAFTAR TABEL

2.1	Matriks Permutasi Awal	5
2.2	Matriks Permutasi untuk <i>Parity Drop</i>	6
2.3	Matriks kompresi <i>P-box</i>	7
2.4	<i>P-box</i>	9
2.5	<i>S-box</i> 1	9
2.6	<i>S-box</i> 2	10
2.7	<i>S-box</i> 3	10
2.8	<i>S-box</i> 4	10
2.9	<i>S-box</i> 5	10
2.10	<i>S-box</i> 6	10
2.11	<i>S-box</i> 7	11
2.12	<i>S-box</i> 8	11
2.13	Matriks Permutasi m	11
2.14	Matriks Permutasi Akhir	12
2.15	Konstanta Awal	13
3.1	Nilai x untuk masing-masing $f(x)$	29
3.2	Hasil Enkripsi setiap <i>Share</i> untuk <i>Password</i> Pertama	30
3.3	Hasil Dekripsi <i>Share</i>	32

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Otentikasi adalah proses untuk menentukan keaslian identitas dari sebuah entitas saat akan mengakses sumber daya sebuah sistem. Proses otentikasi menentukan apakah sebuah entitas berhak atau tidak untuk mengakses sumber daya sistem tersebut.

Salah satu dari metode otentikasi adalah dengan menggunakan *password*. *Password* adalah sekumpulan huruf, angka, dan simbol yang sifatnya rahasia. Umumnya, *password* digunakan bersamaan dengan *username* untuk mengakses sebuah akun, email, dan sebagainya. Entitas yang memiliki *password* dan *username* diijinkan untuk mengakses akun.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dibuat, maka permasalahan yang akan dibahas dalam penelitian ini adalah:

- Bagaimana mengembalikan *password* untuk banyak akun dengan metode *secret sharing* Shamir?
- Bagaimana cara membangun perangkat lunak pengingat *password* yang mengimplementasikan metode *secret sharing* Shamir?

1.3 Tujuan

Berdasarkan rumusan masalah yang sudah ditetapkan, maka tujuan dari penelitian ini adalah:

- Mempelajari bagaimana metode *secret sharing* Shamir dapat mengembalikan *password* untuk banyak akun.
- Membangun perangkat lunak pengingat *password* yang mengimplementasikan metode *secret sharing* Shamir.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah setiap pertanyaan keamanan dijawab dengan jawaban yang relevan.

1.5 Metodologi Penelitian

Metodologi dalam penelitian ini berupa:

- Melakukan studi literatur untuk mempelajari hal-hal yang diperlukan dalam penggunaan dan implementasi metode *secret sharing* Shamir.

- 1 • Membangun perangkat lunak yang mengimplementasikan metode *secret sharing* Sha-
2 mir.
- 3 • Melakukan pengujian pada perangkat lunak yang sudah dibangun.

4 1.6 Sistematika Pembahasan

5 Sistematika pembahasan dalam penelitian ini berupa:

- 6 • Bab Pendahuluan
7 Bab 1 berisi latar belakang, rumusan masalah, tujuan penelitian, batasan masalah,
8 metodologi penelitian, dan sistematika pembahasan.
- 9 • Bab Dasar Teori
10 Bab 2 berisi mengenai teori-teori dasar, antara lain kriptografi, algoritma enkripsi,
11 algoritma fungsi *hash*, otentikasi, *secret sharing*, probabilitas, dan entropi.
- 12 • Bab Analisis
13 Bab 3 berisi analisis meliputi perhitungan dan proses, *flow chart*, *use case*, dan ran-
14 cangan awal diagram kelas.
- 15 • Bab Perancangan
16 Bab 4 berisi tahapan penjelasan rancangan perangkat lunak meliputi algoritma, dia-
17 gram kelas lengkap, dan rancangan tampilan perangkat lunak.
- 18 • Bab Implementasi dan Pengujian
19 Bab 5 berisi tahapan implementasi pada perangkat lunak meliputi tampilan antarmuka
20 perangkat lunak, pengujian perangkat lunak, dan kesimpulan.
- 21 • Bab Kesimpulan dan Saran
22 Bab 6 berisi kesimpulan serta beberapa saran untuk pengembangan lebih lanjut dari
23 penelitian yang dilakukan dan perangkat lunak yang dibangun.

BAB 2

DASAR TEORI

Pada bab ini akan dibahas dasar-dasar teori yang diperlukan dalam proses penulisan penelitian mengenai perlindungan *password* dengan entropi personal. Terdapat beberapa hal yang dibahas pada bab ini, yaitu mengenai kriptografi, *Data Encryption Standard*, *Secure Hash Algorithm 512*, otentikasi, eliminasi Gauss-Jordan, *secret sharing*, probabilitas, dan entropi.

2.1 Kriptografi

Pada bagian ini akan dijelaskan mengenai kriptografi dimulai dari sejarah kriptografi, dan pengertian kriptografi.

2.1.1 Sejarah Kriptografi

Kriptografi berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu, *kripto* dan *graphia*, *kripto* berarti rahasia dan *graphia* berarti tulisan. Jadi, kriptografi berarti teknik atau metode untuk merahasiakan tulisan.

Kemunculan kriptografi ini diawali karena kebutuhan manusia untuk merahasiakan informasi berupa pesan atau tulisan. Pada zaman dahulu kala, kriptografi digunakan untuk merahasiakan tulisan-tulisan mengenai pesan rahasia, strategi perang, dan masih banyak lagi. Salah satu bentuk penggunaan kriptografi pada zaman dahulu kala adalah alat yang dinamakan *scytale*. *Scytale* digunakan oleh tentara Sparta di Yunani untuk mengirimkan pesan rahasia[1].

2.1.2 Pengertian Kriptografi

Zaman sekarang ini, kerahasiaan informasi menjadi hal yang penting. Informasi yang berharga perlu dirahasiakan sehingga tidak diketahui oleh orang yang tidak berhak. Kriptografi berperan dalam merahasiakan informasi berharga tersebut. Jadi, kriptografi adalah ilmu atau seni untuk menjaga kerahasiaan informasi.

Kriptografi memiliki 4 layanan utama:

1. Kerahasiaan (*confidentiality*)

Layanan ini menjamin bahwa informasi yang dikirimkan tidak diketahui oleh pihak yang tidak berhak melihat atau membacanya.

2. Integritas (*integrity*)

Layanan ini menjamin keaslian dari informasi yang dikirimkan dan menjamin bahwa informasi yang dikirimkan tidak diubah tanpa seijin pengirim informasi.

3. Otentikasi (*authentication*)

Layanan ini menjamin keaslian identitas dari pengirim dan penerima informasi.

4. Non-repudiasi (*nonrepudiation*)

Layanan ini menjamin pengirim dan penerima informasi tidak dapat menyangkal aktivitas yang sudah dilakukan.

2.2 Kerahasiaan (*Confidentiality*)

Kerahasiaan adalah layanan yang menjamin bahwa informasi yang dikirimkan tidak dapat dibaca oleh orang atau pihak yang tidak berhak. Dalam kriptografi, informasi yang bisa dibaca dan dimengerti disebut *plaintext*. Informasi yang sudah dirahasiakan sehingga tidak bisa dibaca dan dimengerti disebut *ciphertext*. Untuk merahasiakan *plaintext*, maka *plaintext* harus diubah menjadi *ciphertext*. Kemudian, untuk bisa membaca kembali informasi yang sudah dirahasiakan, *ciphertext* harus diubah kembali menjadi *plaintext*.

Proses untuk mengubah *plaintext* menjadi *ciphertext* dinamakan enkripsi. Sebaliknya, proses untuk mengubah *ciphertext* menjadi *plaintext* dinamakan dekripsi. Proses enkripsi dan dekripsi ini menggunakan kunci. Kunci adalah sekumpulan huruf, angka, atau simbol. Kunci sifatnya rahasia dan hanya boleh diketahui oleh pemilik informasi.

Dalam proses enkripsi, *plaintext* dipetakan dengan fungsi enkripsi E menjadi *ciphertext* menggunakan kunci k , seperti pada persamaan 2.1.

$$E(\textit{plaintext}) = \textit{ciphertext} \quad (2.1)$$

Sementara itu, dalam proses dekripsi, *ciphertext* dipetakan dengan fungsi dekripsi D menjadi *plaintext* menggunakan kunci k seperti pada persamaan 2.2.

$$D(\textit{ciphertext}) = \textit{plaintext} \quad (2.2)$$

Proses enkripsi dan dekripsi ini menggunakan sekumpulan fungsi matematika untuk mengubah *plaintext* menjadi *ciphertext* dan sebaliknya. Sekumpulan fungsi matematika yang digunakan dalam proses enkripsi dan dekripsi dinamakan algoritma kriptografi. Menurut penggunaan kuncinya algoritma kriptografi dibagi menjadi 2 jenis, yaitu algoritma kriptografi kunci simetris dan algoritma kriptografi kunci asimetris.

Algoritma kunci simetris menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. Pemilik informasi melakukan proses enkripsi dan dekripsi dengan kunci yang sama sehingga kunci harus dirahasiakan. Contoh dari algoritma kriptografi kunci simetris antara lain, *Data Encryption Standard* (DES), *Advanced Encryption Standard* (AES), *Twofish*, dan *Blowfish*.

Algoritma kunci asimetris menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsi. Pemilik informasi melakukan proses enkripsi menggunakan kunci yang dinamakan kunci publik dan melakukan proses dekripsi menggunakan kunci yang dinamakan kunci pribadi. Kunci publik sifatnya tidak rahasia dan kunci pribadi sifatnya rahasia. Contoh dari algoritma kriptografi kunci asimetris antara lain, Rivest-Shamir-Adleman (RSA), ElGamal, Diffie-Helman, *Digital Signature Algorithm*, dan *Elliptic Curve Digital Signature Algorithm* (ECDSA).

2.3 Data Encryption Standard (DES)

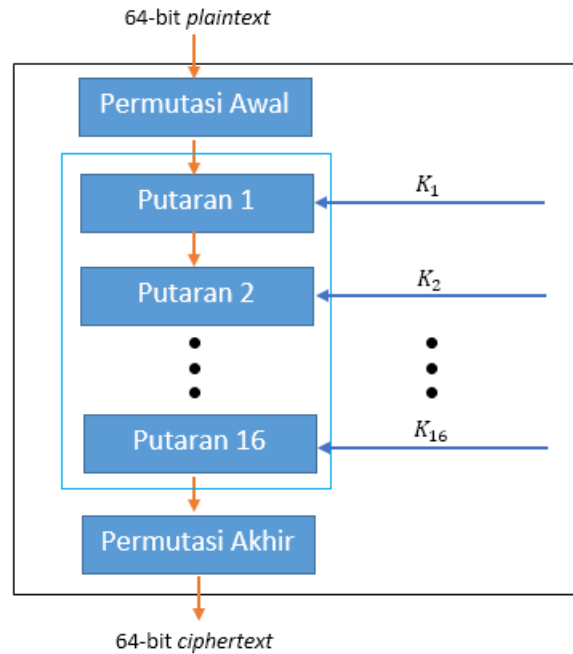
Pada bagian ini akan dijelaskan hal-hal mengenai data encryption standard dimulai dari sejarah data encryption standard dan algoritma dari data encryption standard (?).

2.3.1 Sejarah

Data encryption standard atau disingkat DES adalah algoritma kriptografi kunci simetris. DES pertama kali dipublikasikan oleh *National Institute of Standards and Technology* (NIST) pada tahun 1973. DES merupakan algoritma enkripsi pertama yang disetujui oleh pemerintah Amerika Serikat untuk digunakan secara luas. Pada bulan Maret 1975, NIST memublikasikan DES sebagai standar enkripsi untuk data pemerintahan atau *Federal Information Processing Standard* (FIPS).

2.3.2 Struktur DES

Masukan dari DES berupa 64-bit *plaintext*. Keluaran dari DES berupa 64-bit *ciphertext*. DES menggunakan kunci yang sama pada proses enkripsi dan dekripsi. Panjang kunci dari DES adalah 64-bit. Proses enkripsi terdiri dari permutasi awal, putaran dan permutasi akhir. Gambar 2.1 menunjukkan proses enkripsi dari DES. Pada bagian selanjutnya akan dijelaskan mengenai setiap bagian dari proses enkripsi.



Gambar 2.1: Proses Enkripsi

2.3.3 Permutasi Awal

Permutasi awal dalam DES menggunakan matriks permutasi mp . Masukan dari matriks permutasi mp adalah *plaintext*. Tabel 2.1 menunjukkan matriks permutasi mp .

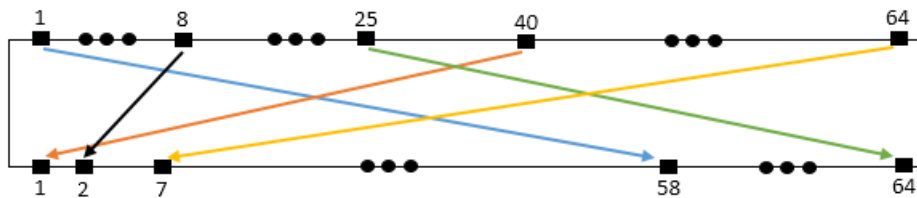
Tabel 2.1: Matriks Permutasi Awal

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Cara kerja dari proses permutasi adalah sebagai berikut. Angka yang ditunjukkan pada posisi ke- i matriks mp merupakan posisi *bit* dari masukan, sedangkan i menunjukkan posisi *bit* dari keluaran. Proses permutasi ditunjukkan oleh persamaan 2.3.

$$keluaran_i = masukan_{p_i} \quad (2.3)$$

Sebagai contoh, posisi ke-1 dari matriks mp menunjukkan angka 58. Maka, *bit* ke-58 dari masukan akan menjadi *bit* ke-1 dari keluaran. Gambar 2.2 menunjukkan ilustrasi dari proses permutasi yang sudah dijelaskan.



Gambar 2.2: Proses Permutasi

2.3.4 Pembangunan Kunci Putaran

DES menggunakan kunci dengan panjang 64-bit. Kunci ini perlu diubah menjadi kunci untuk setiap putaran DES dengan panjang masing-masing 48-bit. Proses pembangunan kunci putaran terdiri dari *parity drop*, *shift left*, dan permutasi *P-box*. Pada bagian ini akan dijelaskan proses pengubahan kunci 64-bit menjadi kunci putaran dengan panjang 48-bit.

Parity Drop

Pada proses ini, *parity bit* akan dihilangkan dari kunci masukan. *Bit* yang dihilangkan adalah *bit* posisi kelipatan 8, yaitu posisi ke-8, posisi ke-16, posisi ke-24, dan seterusnya sampai posisi ke-64. Proses penghilangan *parity bit* ini menggunakan matriks permutasi *p* seperti ditunjukkan pada Tabel 2.2. Cara kerja proses permutasi sama dengan cara kerja proses permutasi pada tahap permutasi awal (Subbab 2.3.3).

Tabel 2.2: Matriks Permutasi untuk *Parity Drop*

57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4

Hasil akhir dari proses ini kunci dengan panjang 56-bit.

Shift Left

Pada proses ini, kunci hasil proses *parity drop* dibagi menjadi 2 bagian dengan panjang masing-masing 28-bit, yaitu bagian kiri (*L*) dan bagian kanan (*R*). *L* dan *R* akan digeser ke arah kiri secara sirkular sebanyak 1 atau 2 bit tergantung dari urutan putaran. Ketentuan banyak *bit* yang digeser adalah sebagai berikut.

- Untuk putaran ke-1, 2, 9, dan 16 maka *L* dan *R* akan digeser ke arah kiri secara sirkular sebanyak 1 bit.
- Untuk putaran ke-3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, dan 15, *L* dan *R* akan digeser ke arah kiri secara sirkular sebanyak 2 bit.

Sebagai contoh, diasumsikan *L* dan *R* pada persamaan 2.4 dan 2.5.

$$L = 1001\ 1010\ 1000\ 0110\ 0110\ 1111\ 1101 \quad (2.4)$$

$$R = 0001\ 0100\ 0111\ 1110\ 1010\ 0101\ 1011 \quad (2.5)$$

1 Untuk putaran ke-1, 2, 9, dan 16 maka hasil dari L dan R akan seperti yang ditunjukkan
 2 pada persamaan 2.6 dan 2.7.

$$L = 0011\ 0101\ 0000\ 1100\ 1101\ 1111\ 1011 \quad (2.6)$$

$$R = 0010\ 1000\ 1111\ 1101\ 0100\ 1011\ 0110 \quad (2.7)$$

3 Sementara itu, jika untuk putaran ke-3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, dan 15 akan
 4 seperti yang ditunjukkan pada persamaan 2.8 dan 2.9.

$$L = 0110\ 1010\ 0001\ 1001\ 1011\ 1111\ 0110 \quad (2.8)$$

$$R = 0101\ 0001\ 1111\ 1010\ 1001\ 0110\ 1100 \quad (2.9)$$

5 Kemudian, L dan R akan disatukan kembali sehingga panjangnya menjadi 56-bit.

6 Permutasi P -box

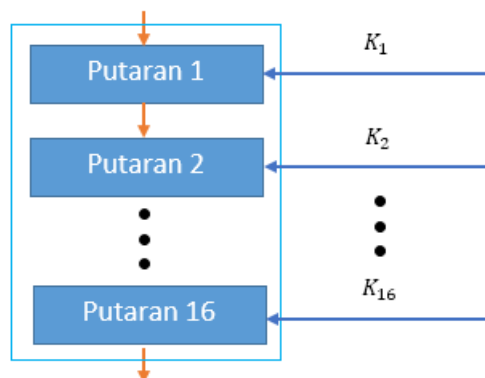
7 Tahap ini adalah proses permutasi untuk mengubah kunci dari proses *Shift Left* dengan pan-
 8 jang 56-bit menjadi kunci putaran dengan panjang 48-bit. Tabel 2.3 menunjukkan matriks
 9 permutasi P -box yang digunakan untuk proses ini.

Tabel 2.3: Matriks kompresi P -box

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
32	29	36	50	42	46	53	34

10 2.3.5 Putaran

11 DES terdiri dari 16 putaran. Setiap putaran adalah jaringan Feistel yang akan dijelaskan
 12 pada bagian selanjutnya. Gambar 2.3 menunjukkan ilustrasi dari 16 putaran dari DES.



Gambar 2.3: Putaran dalam DES

13 Jaringan Feistel

14 Pada bagian ini akan dijelaskan mengenai sejarah singkat dari jaringan Feistel dan pemba-
 15 hasan jaringan Fesitel.

1 Sejarah Singkat

- 2 Jaringan Feistel diciptakan oleh ilmuwan asal Jerman bernama Horst Feistel. Horst Feistel
- 3 mempublikasikan jaringan ini pada tahun 1973. Jaringan Feistel banyak digunakan dalam
- 4 berbagai skema enkripsi khususnya digunakan dalam DES.

5 Pembahasan

- 6 Masukan dari jaringan Feistel adalah *plaintext* dengan panjang 64-bit dan keluaran dari
- 7 jaringan Feistel adalah *ciphertext* dengan panjang 64-bit. Jaringan Feistel menggunakan
- 8 kunci K dan fungsi enkripsi f dalam pemrosesan *plaintext*. Selanjutnya akan dijelaskan
- 9 langkah-langkah pemrosesan *plaintext* pada jaringan Feistel.

- 10 1. *Plaintext* dibagi menjadi 2 bagian sama panjang, yaitu bagian kiri (L_{i-1}) dan bagian
- 11 kanan (R_{i-1}). Huruf i menunjukkan urutan dari putaran. Panjang masing-masing
- 12 bagian adalah 32-bit.

2. Bagian kanan (R_{i-1}) pada *plaintext* akan menjadi bagian kiri (L_i) dari *ciphertext*. Persamaan 2.10 menunjukkan langkah yang sudah dijelaskan.

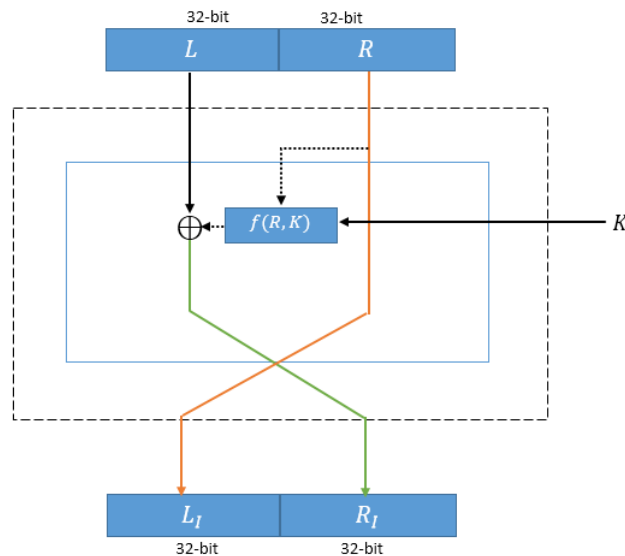
$$L_i = R_{i-1} \quad (2.10)$$

3. Untuk memperoleh bagian kanan dari *ciphertext* (R_i), bagian kanan dari *plaintext* (R_{i-1}) dan kunci putaran K_i dipetakan dengan fungsi f . Kemudian, hasil pemetaan dengan fungsi f akan di *exclusive-or* (XOR) dengan bagian kiri dari *plaintext* (L). Persamaan 2.11 menunjukkan langkah yang sudah dijelaskan.

$$R_i = L \oplus f(R_{i-1}, K_i) \quad (2.11)$$

- 13 4. Hasil akhirnya berupa *ciphertext* dengan 2 bagian sama panjang, yaitu bagian kiri (L_i)
- 14 dan bagian kanan (R_i).

- 15 Gambar 2.4 menunjukkan ilustrasi dari langkah-langkah yang sudah dijelaskan.



Gambar 2.4: Jaringan Feistel

1 Fungsi DES

2 Fungsi DES adalah fungsi f yang digunakan dalam jaringan Feistel pada Gambar 2.4. Fungsi
 3 DES terdiri dari 4 bagian, yaitu ekspansi P -box, operasi XOR, substitusi S -box, dan permutasi.
 4 Pada bagian selanjutnya akan dijelaskan masing-masing bagian dari fungsi DES.

5 Ekspansi P -box

6 Pada bagian ini, masukan berupa blok bagian kanan dari *plaintext* (R) dengan panjang
 7 32-bit. Ekspansi P -box menggunakan matriks permutasi p yang ditunjukkan pada tabel 2.4.

Tabel 2.4: P -box

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

8 Hasil keluaran dari ekspansi P -box adalah blok dengan panjang 48-bit.

9 Operasi XOR

10 Setelah ekspansi P -box, dilakukan operasi XOR antara R dengan kunci putaran ke- i , K_i .
 11 Kunci putaran hanya digunakan pada bagian ini saja.

12 Substitusi S -box

13 Pada bagian ini, akan dilakukan substitusi pada R dengan menggunakan S -box. Masukan
 14 dari S -box adalah R dengan panjang 48-bit dan keluarannya adalah R dengan panjang 32-bit.
 15 R akan dibagi menjadi 8 blok dengan panjang masing-masing 6-bit. Setiap blok memiliki S -
 16 box masing-masing. Blok pertama menggunakan S -box pertama, blok kedua menggunakan
 17 S -box kedua, dan seterusnya. Berikut masing-masing dari S -box ditunjukkan pada Tabel 2.5
 18 sampai Tabel 2.12.

Tabel 2.5: S -box 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	10	3	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabel 2.6: *S-box 2*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	12	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tabel 2.7: *S-box 3*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tabel 2.8: *S-box 4*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tabel 2.9: *S-box 5*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tabel 2.10: *S-box 6*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tabel 2.11: *S-box* 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tabel 2.12: *S-box* 8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

1 Proses substitusi terjadi sebagai berikut. Kombinasi *bit* ke-1 dan *bit* ke-6 pada blok
2 akan menunjukkan posisi baris pada *S-box*. Kemudian, kombinasi dari *bit* ke-2 sampai ke-5
3 menunjukkan posisi kolom pada *S-box*. Setelah itu, angka yang ditunjuk oleh baris dan
4 kolom pada *S-box* ini akan menjadi blok keluaran.

5 Sebagai contoh, diasumsikan masukan dari *S-box* pertama adalah 110011. Maka, kom-
6 binasi *bit*nya adalah 11 untuk baris dan 1001 untuk kolom. Jadi, baris yang dipilih adalah
7 baris ke-3 dan kolom yang dipilih adalah kolom ke-9. Angka yang ditunjuk oleh *S-box* per-
8 tama pada baris ke-3 dan kolom ke-9 adalah 11. Maka, blok keluaran untuk *S-box* pertama
9 adalah 1011. Lalu, setelah seluruh blok masukan diproses dengan *S-box* masing-masing,
10 seluruh blok keluaran digabungkan menjadi blok dengan panjang 32-*bit*.

11 Permutasi

12 Bagian ini adalah bagian terakhir dari fungsi DES. Masukan dari bagian ini adalah blok
13 keluaran dari proses substitusi *S-box*, yaitu blok dengan panjang 32-*bit*. Proses permutasi
14 dilakukan dengan menggunakan matriks m yang ditunjukkan oleh Tabel 2.13. Hasil keluaran
15 dari bagian ini adalah blok dengan panjang 32-*bit*.

Tabel 2.13: Matriks Permutasi m

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

16 Setelah proses permutasi ini, hasil dari proses permutasi akan di exclusive-or (XOR)
17 dengan L_{i-1} seperti yang sudah dijelaskan pada bagian Jaringan Feistel. Hasil XOR adalah
18 bagian kanan dari ciphertext (R_i). Setelah itu, L_i dan R_i akan digabungkan kemudian
19 dijadikan sebagai masukan untuk putaran selanjutnya.

20 2.3.6 Permutasi Akhir

21 Setelah dilakukan 16 putaran, tahap terakhir dari enkripsi DES adalah permutasi akhir.
22 Proses permutasi akhir menggunakan matriks yang ditunjukkan pada Tabel 2.14. Hasil dari

- 1 proses permutasi akhir adalah 64-bit *ciphertext*.

Tabel 2.14: Matriks Permutasi Akhir

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

2 2.4 Fungsi Hash

- 3 Fungsi *hash* adalah fungsi yang memiliki masukan berupa *string* dengan panjang sembarang
 4 dan menghasilkan keluaran berupa *string* dengan panjang yang tetap. Masukan dari fungsi
 5 *hash* dinamakan *message*. Hasil keluaran dari fungsi *hash* dinamakan *digest*. *Message m*
 6 akan dipetakan dengan fungsi *hash H* menghasilkan *digest h*. Persamaan 2.12 menunjukkan
 7 pemetaan *m* dengan *H* yang menghasilkan *h*.

$$h = H(m) \quad (2.12)$$

- 8 Fungsi *hash* harus memiliki 3 kriteria sebagai berikut[2].

- 9 1. Preimage Resistance
 10 Untuk setiap $h = H(m)$ yang dihasilkan, tidak mungkin dikembalikan m sedemikian
 11 rupa sehingga $H(m) = h$. Dalam proses pembuatan *digest*, fungsi *hash* menghilangkan
 12 beberapa bagian dari m (*lossy*). Maka dari itu, *digest* tidak bisa dikembalikan menjadi
 13 *message*. Itulah sebabnya fungsi *hash* disebut fungsi satu arah.
- 14 2. Second Preimage Resistance
 15 Untuk setiap m yang diberikan, tidak mungkin mencari $m' \neq m$ sedemikian rupa
 16 sehingga $H(m') = H(m)$.
- 17 3. Collision Resistance
 18 Tidak mungkin mencari pasangan m dan m' sedemikian rupa sehingga $h = H(m)$
 19 sama dengan $h' = H(m')$. Untuk 2 *message* yang berbeda tidak mungkin menghasilkan
 20 *digest* yang sama.

- 21 Contoh fungsi *hash* antara lain MD-2, MD-4, MD-5, SHA-0, SHA-1, SHA-256, dan
 22 SHA-512.

23 2.5 Secure Hashing Algorithm 512 (SHA-512)

- 24 *Secure hashing algorithm* 512 atau SHA-512 adalah algoritma fungsi *hash* yang menghasilkan
 25 *digest* dengan panjang 512-bit. Proses dari SHA-512 terdiri dari *message padding*, inisialisasi
 26 konstanta awal, ekspansi blok *message*, fungsi kompresi, dan putaran. Bagian selanjutnya
 27 akan menjelaskan masing-masing proses dari SHA-512.

28 2.5.1 Message Padding

- 29 Sebelum *digest* dibuat, *message* akan dipadding terlebih dahulu. Pertama-tama, blok *mes-*
 30 *sage M* akan dipadding dengan blok *L*. Blok *L* berisi informasi mengenai panjang dari *M*.
 31 Panjang dari blok *L* adalah 128-bit. Kemudian, gabungan dari blok *M* dan *L* akan dipadding

1 lagi dengan blok *padding* P sampai panjang dari gabungan blok M , L , dan P mencapai ke-
 2 lipatan 1024-bit. Panjang dari blok *padding* P bervariasi. Persamaan 2.13 menunjukkan
 3 rumus untuk menghitung panjang dari blok *padding* P .

$$(M + P + 128) = 0 \bmod 1024 \quad \Rightarrow \quad P = (-M - 128) \bmod 1024 \quad (2.13)$$

4 Isi dari blok *padding* P adalah angka 1 diikuti dengan angka 0. Sebagai contoh, jika
 5 panjang dari *message* (M) adalah 2590 bit, maka panjang dari blok *padding* P ditunjukkan
 6 pada persamaan 2.14.

$$\begin{aligned} P &= (-2590 - 128) \bmod 1024 \\ &= -2718 \bmod 1024 \\ &= 354 \end{aligned} \quad (2.14)$$

7 Maka, dari persamaan 2.14, panjang dari blok P adalah 354 bit. Isi dari blok P adalah
 8 1 bit angka 1 diikuti dengan 353 bit angka 0.

9 2.5.2 Inisialisasi Konstanta Awal

10 Setelah proses *message padding*, proses selanjutnya adalah inisialisasi konstanta awal. Ada 8
 11 konstanta awal yang akan dibentuk. Delapan konstanta awal ini akan diberi nama $A_0, B_0, C_0,$
 12 $D_0, E_0, F_0, G_0,$ dan H_0 . Panjang masing-masing konstanta awal ini adalah 64-bit. Setiap
 13 nilai konstanta awal diperoleh dari nilai di belakang koma dari akar kuadrat bilangan prima.
 14 Kemudian, nilai di belakang koma ini akan diubah menjadi heksadesimal. Bilangan prima
 15 yang digunakan untuk masing-masing konstanta awal adalah bilangan prima awal secara
 16 berurutan, yaitu 2, 3, 5, 7, 11, 13, 17, dan 19.

17 Sebagai contoh, misalkan akan dicari nilai untuk A_0 . A_0 merupakan konstanta awal
 18 pertama maka bilangan prima yang digunakan adalah bilangan prima urutan pertama, yaitu
 19 2. Setelah itu, akan dihitung akar kuadrat dari 2. Kemudian, angka di belakang koma dari
 20 akar kuadrat 2 akan diubah menjadi heksadesimal. Nilai heksadesimal inilah yang menjadi
 21 nilai dari A_0 . Persamaan 2.15 menunjukkan langkah yang sudah dijelaskan.

$$\begin{aligned} A_0 &= \sqrt{2} \\ &= 1.4142135623730950 \\ &= (1.6A09E667F3BCC908)_{16} \\ &= 6A09E667F3BCC908 \end{aligned} \quad (2.15)$$

22 Tabel 2.15 menunjukkan nilai masing-masing konstanta.

Tabel 2.15: Konstanta Awal

Konstanta	Nilai	Konstanta	Nilai
A_0	6A09E667F3BCC908	E_0	510E527FADE682D1
B_0	BB67AE8584CAA73B	F_0	9B05688C2B3E6C1F
C_0	3C6EF372FE94F828	G_0	1F83D9ABFB41BD6B
D_0	A54FF53A5F1D36F1	H_0	5BE0CD19137E2179

23 2.5.3 Ekspansi Blok Message

24 Setelah inisialisasi konstanta awal, proses berikutnya adalah ekspansi blok *message*. Sesudah
 25 blok *message* dipadding, blok *message* akan dibagi menjadi beberapa blok yang panjangnya

1 masing-masing 1024-*bit*. Kemudian, setelah dibagi menjadi beberapa blok, masing-masing
 2 dari blok akan dibagi lagi menjadi blok-blok dengan panjang 64-*bit*. Blok dengan panjang
 3 64-*bit* ini dinamakan *word*.

4 Satu blok 1024-bit terdiri dari 16 *word*. Proses ekspansi blok *message* akan mengekspansi
 5 dari 16 *word* menjadi 80 *word*. Masing-masing *word* ini akan diberi nama W_0 sampai W_{79} .
 6 Untuk W_0 sampai W_{15} berisi dari 16 *word* pertama dari blok 1024-*bit*. Sementara itu, W_{16}
 7 sampai W_{79} diperoleh dengan rumus dasar yang ditunjukkan oleh persamaan 2.16.

$$W_i = W_{i-16} \oplus RotShift_{1-8-7}(W_{i-15}) \oplus W_{i-7} \oplus RotShift_{19-61-6}(W_{i-2}) \quad (2.16)$$

8 Sebagai contoh untuk memperoleh nilai dari W_{60} , maka rumus dasarnya adalah seperti
 9 yang ditunjukkan pada persamaan 2.17.

$$W_{60} = W_{44} \oplus RotShift_{1-8-7}(W_{45}) \oplus W_{53} \oplus RotShift_{19-61-6}(W_{58}) \quad (2.17)$$

10 *RotShift* pada persamaan 2.16 dan 2.17 adalah hasil *exclusive-or* (XOR) dari operasi
 11 rotasi ke kanan dan *shift left*. Rumus untuk rotasi ke kanan dan *shift left* ditunjukkan pada
 12 persamaan 2.18.

$$RotShift_{l-m-n}(x) = RotR_l(x) \oplus RotR_m(x) \oplus ShL_n(x) \quad (2.18)$$

13 $RotR_i(x)$ pada persamaan 2.18 adalah rotasi ke kanan x sebanyak i *bit*. Sebagai contoh,
 14 diasumsikan $i = 2$ dan $x = 1001$, maka hasil dari $RotR_2(1001)$ ditunjukkan pada persamaan
 15 2.19.

$$\begin{aligned} i = 1 & \Rightarrow x = 1100 \\ i = 2 & \Rightarrow x = 0110 \\ RotR_2(1001) &= 0110 \end{aligned} \quad (2.19)$$

16 Sementara itu, $ShL_i(x)$ pada persamaan 2.18 adalah operasi *shift left* x sebanyak i *bit*
 17 *dipadding* dengan angka 0. Sebagai contoh, diasumsikan $i = 2$ dan $x = 1011$, maka hasil
 18 dari $ShL_2(1011)$ ditunjukkan pada persamaan 2.20.

$$\begin{aligned} i = 1 & \Rightarrow x = 0110 \\ i = 2 & \Rightarrow x = 1100 \\ ShL_2(1011) &= 1100 \end{aligned} \quad (2.20)$$

19 Setelah ekspansi blok *message* menjadi 80 *word* untuk setiap blok *message*, proses selan-
 20 jutnya adalah putaran dari SHA-512. Proses putaran SHA-512 akan dijelaskan pada bagian
 21 selanjutnya.

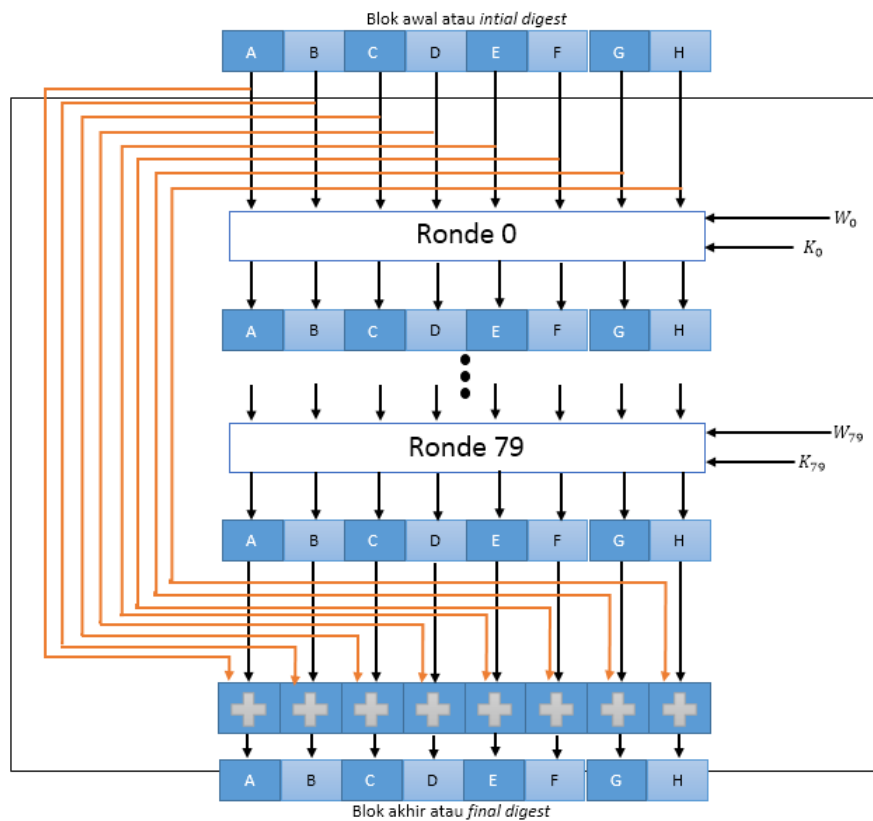
22 2.5.4 Fungsi Kompresi dan Putaran

23 Fungsi kompresi adalah proses yang mengkompresi blok 512-*bit* dan blok *message* yang
 24 berukuran 1024-*bit* menjadi blok keluaran dengan panjang 512-*bit*. Fungsi kompresi ini
 25 terdiri dari 80 putaran SHA-512.

26 Struktur Putaran

27 Masukan dari putaran SHA-512 adalah blok dengan panjang 512-*bit* terdiri dari 8 *word* (A ,
 28 B , C , D , E , F , G , dan H). Untuk putaran pertama, blok 512-*bit* diperoleh dari konstanta
 29 awal (A_0 sampai H_0) sedangkan untuk putaran kedua dan selanjutnya blok 512-*bit* diperoleh

- 1 dari hasil dari putaran sebelumnya. Gambar 2.5 menunjukkan ilustrasi proses yang sudah
 2 dijelaskan.

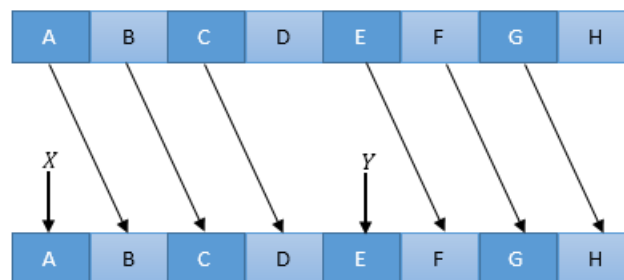


Gambar 2.5: Struktur Putaran dalam SHA-512

- 3 Dalam 1 putaran SHA-512, *word* keluaran diperoleh dari salinan *word masukan*, berikut
 4 menunjukkan masukan dan keluaran dari masing-masing *word*.

- 5 • *Word* keluaran *B* diperoleh dari *word* masukan *A*
- 6 • *Word* keluaran *C* diperoleh dari *word* masukan *B*
- 7 • *Word* keluaran *D* diperoleh dari *word* masukan *C*
- 8 • *Word* keluaran *F* diperoleh dari *word* masukan *E*
- 9 • *Word* keluaran *G* diperoleh dari *word* masukan *F*
- 10 • *Word* keluaran *H* diperoleh dari *word* masukan *G*

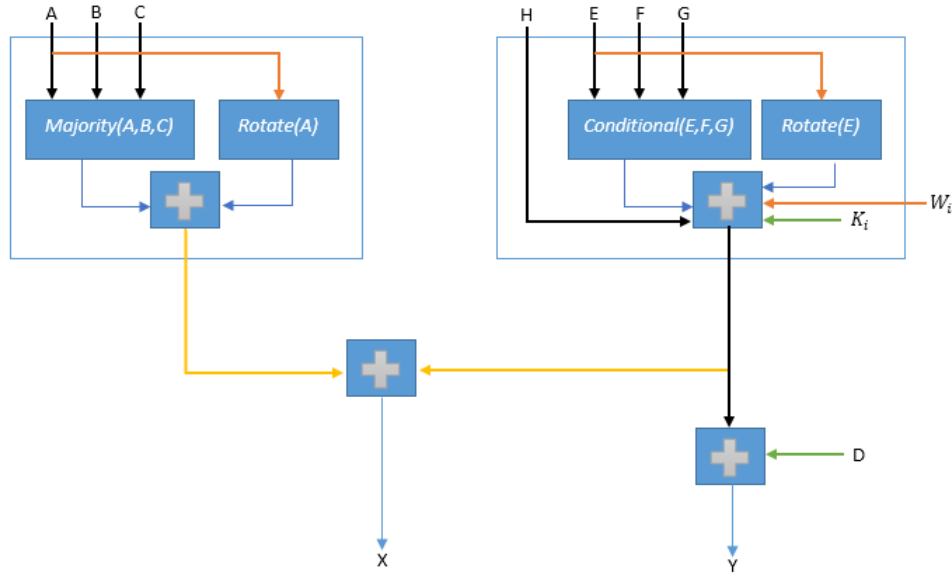
- 11 Gambar 2.6 menunjukkan ilustrasi dari masukan dan keluaran dalam 1 putaran SHA-512
 12 untuk setiap *word*.



Gambar 2.6: Masukan dan Keluaran dalam 1 Putaran SHA-512

- 1 Untuk nilai *word* keluaran A dan E diperoleh dari *word* X dan Y . *Word* X dan Y ini
- 2 diperoleh dari sebuah fungsi khusus. Gambar 2.7 menunjukkan struktur dari fungsi khusus.
- 3 Berikut akan dijelaskan struktur dari fungsi khusus.

4 Struktur Fungsi Khusus



Gambar 2.7: Fungsi Khusus dalam 1 Putaran SHA-512

- 5 *Word* Y pada gambar 2.7 diperoleh dari proses persamaan 2.21.

$$Y = D + (Conditional(E, F, G) + Rotate(E) + W_i + K_i + H) \quad (2.21)$$

- 6 Nilai W_i diperoleh dari proses Ekspansi Blok *Message* (Subbab ??), dimana i menun-
- 7 jukkan urutan dari putaran. Nilai K_i pada persamaan 2.21 diperoleh dari nilai belakang
- 8 koma akar kubik bilangan prima ke- $(i + 1)$. Kemudian, nilai belakang koma ini akan dikon-
- 9 versi menjadi heksadesimal.

- 10 Bilangan prima yang digunakan untuk menghitung nilai K_i dimulai dari 2 untuk K_0 ,
- 11 3 untuk K_1 , dan seterusnya secara berurutan sampai 409 untuk K_{79} . Persamaan 2.22
- 12 menunjukkan cara untuk menghitung salah satu dari nilai K_i .

$$\begin{aligned} K_{79} &= \sqrt[3]{409} \\ &= 7.4229141204362155 \\ &= (7.6C44198C4A475817)_{16} \\ &= 6C44198C4A475817 \end{aligned} \quad (2.22)$$

- 13 Sementara itu, untuk operasi *Conditional* pada persamaan 2.21 adalah operasi *AND*,
- 14 *OR* dan *XOR* dari *bit-bit* setiap *word*. Rumus dari *Conditional* ditunjukkan oleh persamaan
- 15 2.23.

$$Conditional(x, y, z) = (x \text{ AND } y) \oplus (NOT \ x \text{ AND } z) \quad (2.23)$$

- 16 Operasi *Rotate* pada persamaan 2.21 adalah hasil *exclusive-or* (XOR) dari $RotR_i(x)$.
- 17 $RotR_i(x)$ merupakan operasi rotasi ke kanan x sebanyak i -bit yang sudah dijelaskan pa-
- 18 da proses Ekspansi Blok *Message* (Subbab 2.5.3). Rumus dari *Rotate* ditunjukkan pada

1 persamaan 2.24.

$$Rotate(x) = RotR_{28}(x) \oplus RotR_{34}(x) \oplus RotR_{39}(x) \quad (2.24)$$

2 Hasil pertambahan *bit-bit* operasi *Conditional*, operasi *Rotate*, W_i , K_i , dan *word* H akan
3 ditambahkan dengan *word* D untuk menghasilkan *word* Y .

4 Kemudian, *word* X pada Gambar 2.7 diperoleh dari persamaan 2.25.

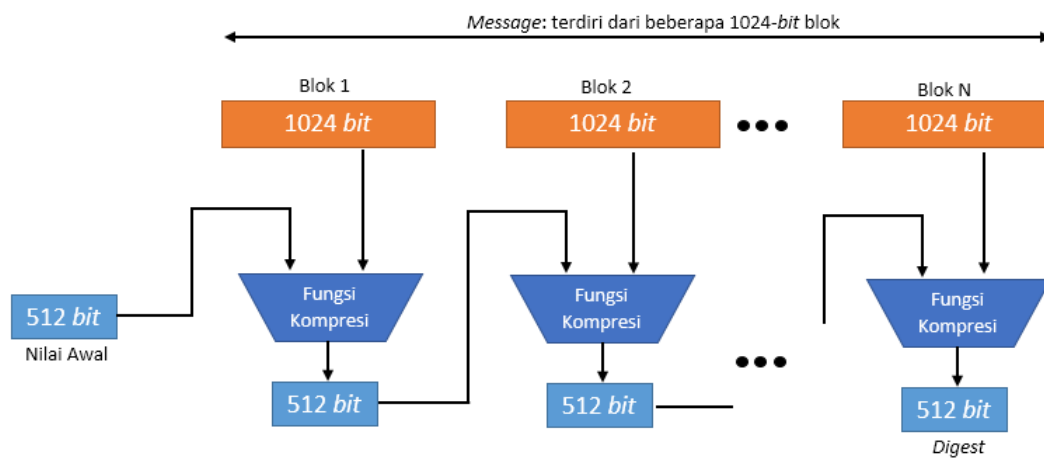
$$X = (Majority(A, B, C) + Rotate(A)) + (Conditional(E, F, G) + Rotate(E) + W_i + K_i + H) \quad (2.25)$$

5 Untuk operasi *Conditional* dan *Rotate* sudah dijelaskan pada persamaan 2.23 dan 2.23.
6 Sementara itu, untuk operasi *Majority* pada persamaan 2.25 adalah operasi *AND*, *OR* dan
7 *XOR* dari *bit-bit* setiap *word*. Operasi *Majority* ditunjukkan pada persamaan 2.26.

$$Majority(x, y, z) = (x \text{ AND } y) \oplus (y \text{ AND } z) \oplus (z \text{ AND } x) \quad (2.26)$$

8 Hasil akhir dari fungsi khusus adalah *word* X dan *word* Y . *Word* X akan menjadi *word*
9 keluaran A dan *word* Y akan menjadi *word* keluaran E . Ilustrasi dari hasil keluaran ini dapat
10 dilihat pada Gambar 2.6.

11 Setelah 80 putaran dilakukan, operasi pertambahan *bit-bit* akan dilakukan pada hasil
12 putaran ke-80 dengan masukan untuk putaran ke-1. Hasil operasi pertambahan *bit-bit* ini
13 berupa blok dengan panjang 512-bit terdiri dari 8 *word*. Blok 512-bit ini akan menjadi
14 hasil akhir (*digest*) atau menjadi masukan untuk fungsi kompresi yang digunakan oleh blok
15 *message* ke-2 dan seterusnya. Gambar 2.8 menunjukkan proses yang sudah dijelaskan.



Gambar 2.8: Proses Keseluruhan dari SHA-512

16 2.6 Otentikasi

17 Otentikasi adalah proses untuk menentukan keaslian identitas dari sebuah entitas saat akan
18 mengakses sumber daya sebuah sistem. Berdasarkan entitas yang diotentikasi [2], otentikasi
19 dibagi menjadi 2 jenis, yaitu:

20 1. Otentikasi pesan

21 Otentikasi pesan adalah proses otentikasi untuk memastikan bahwa pesan berasal dari
22 sumber data yang bisa dipercaya. Otentikasi pesan juga memastikan bahwa pesan
23 tidak diubah saat pengiriman pesan sedang berlangsung. Beberapa teknik otentikasi
24 pesan adalah *Modification Detection Code* dan *Message Authentication Code*.

2. Otentikasi entitas

Otentikasi entitas adalah proses otentikasi untuk memastikan kebenaran identitas seseorang. Entitas yang diotentikasi bisa berupa orang atau pengguna (*user*). Beberapa teknik otentikasi entitas adalah *password*, *zero-knowledge*, *challenge-response*, dan biometrik.

Sementara itu, berdasarkan bentuknya[2], otentikasi dibagi menjadi 3 jenis, yaitu:

1. Sesuatu yang diketahui (*something known*)

Sesuatu yang diketahui oleh pengirim pesan dan kebenarannya bisa dipastikan oleh penerima pesan. Contohnya antara lain adalah *password*, nomor PIN, *passphrase*, dan sebagainya.

2. Sesuatu yang dimiliki (*something possessed*)

Sesuatu yang dimiliki adalah sesuatu yang menunjukkan identitas dari pengirim pesan. Contohnya adalah paspor, KTP, kartu kredit, SIM, dan sebagainya.

3. Sesuatu yang melekat (*something inherent*)

Sesuatu yang melekat adalah sesuatu yang menempel atau sebagai bagian dari pengirim pesan. Contohnya adalah sidik jari, suara, pola retina, dan sebagainya.

2.6.1 Password

Password adalah sekumpulan huruf, angka, dan simbol yang sifatnya rahasia. *Password* merupakan salah satu teknik dari otentikasi entitas. *Password* digunakan saat seseorang hendak mengakses sumber daya sebuah sistem, seperti *email*, akun media sosial, dan sebagainya. *Password* ini sifatnya rahasia dan tidak boleh diketahui oleh pihak yang tidak berhak.

Berdasarkan cara penggunaannya[2], *password* dibagi menjadi 2 jenis, yaitu:

1. One-Time Password

One-Time Password adalah *password* yang digunakan hanya satu kali untuk setiap akses kepada sistem. Jadi, setiap kali pengguna mengakses sistem dalam sesi waktu yang berbeda, *password* yang digunakan pun akan berbeda-beda. Beberapa contoh dari *One-Time Password* adalah *List of Passwords*, *Sequentially Updated Password*, dan *Lamport One-Time Password*.

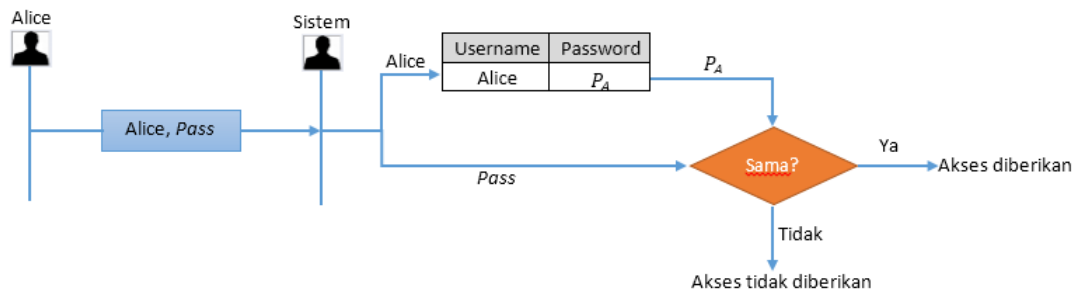
2. Password Tetap

Password tetap adalah *password* yang digunakan berulang-ulang setiap kali pengguna akan mengakses sistem. *Password* yang digunakan untuk mengakses sistem selalu sama. Berikut adalah beberapa skema dari *password* tetap.

Skema 1

Dalam skema ini, sistem menyimpan setiap *password* pada sebuah tabel basis data. *Password* yang disimpan di tabel basis data berupa *plaintext*, artinya bisa dibaca dan dimengerti. Masing-masing dari *password* memiliki *username* yang disimpan juga di tabel basis data. Saat pengguna akan mengakses sistem, pengguna akan memasukan *username* dan *password*.

Kemudian, saat pengguna sudah memasukan *username* dan *password*, sistem akan mencari informasi dari pengguna di tabel basis data lewat *username*. Karena setiap *username* memiliki *password*, sistem akan menyesuaikan *username* dan *password* di tabel basis data dengan *username* dan *password* yang dimasukan oleh pengguna saat hendak mengakses sistem. Jika *username* dan *password* yang dimasukan pengguna sesuai dengan *username* dan *password* di tabel basis data maka hak akses sistem akan diberikan. Gambar 2.9 menunjukkan proses yang dijelaskan.

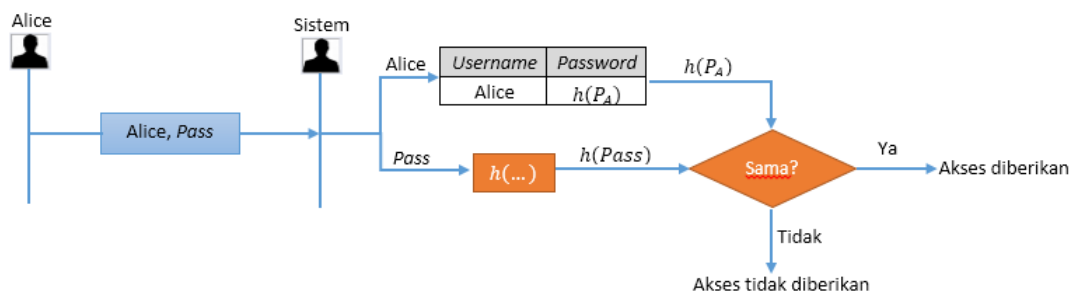
Gambar 2.9: *Username dan Password*

Kelebihan dari skema ini adalah skema ini mudah untuk diimplementasikan dan tidak membutuhkan proses yang rumit. Sementara itu, kekurangan dari skema ini adalah *password* yang disimpan di tabel basis data bisa dibaca dan dimengerti karena disimpan dalam bentuk *plaintext*. Akibatnya, jika ada pihak yang tidak memiliki hak akses berhasil memperoleh *password* yang disimpan di tabel basis data, maka *password* sudah tidak rahasia lagi.

Skema 2

Dalam skema ini, sistem tetap menyimpan *username* dan *password* dalam tabel basis data. *Password* yang disimpan tidak dalam bentuk *plaintext*nya, tetapi disimpan dalam bentuk *digest*nya. Saat pengguna hendak mengakses sistem, pengguna tetap memasukkan *username* dan *password* dalam bentuk *plaintext*.

Kemudian, saat pengguna sudah memasukkan *username* dan *password*, sistem akan terlebih dahulu menghitung *digest* dari *password* yang dimasukkan menggunakan fungsi *hash*. Setelah itu, *username* dan *digest* akan disesuaikan dengan *username* dan *digest* yang disimpan dalam tabel basis data. Jika sesuai, maka pengguna akan diberikan hak akses ke sistem. Gambar 2.10 menunjukkan proses yang sudah dijelaskan.

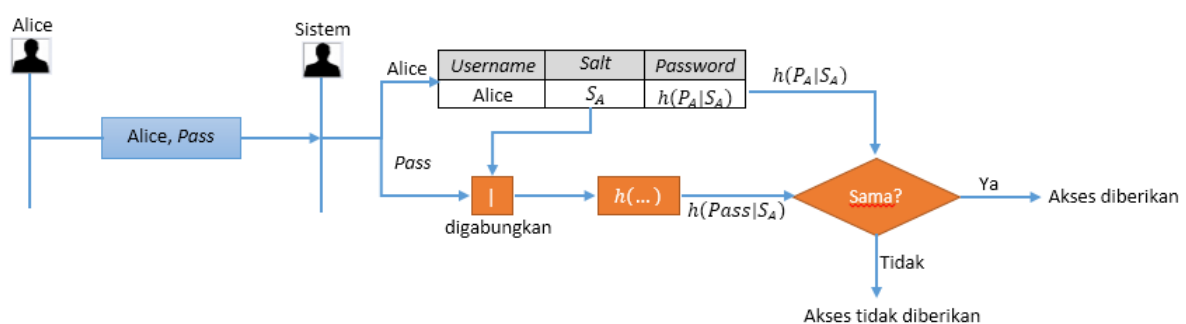
Gambar 2.10: *Password hashing*

Kelebihan dari skema ini adalah walaupun *password* yang disimpan dalam tabel basis data diketahui oleh pihak yang tidak berhak, *password* tidak akan bisa dimengerti karena disimpan dalam bentuk *digest*nya. Sementara itu, *digest* tidak bisa dikembalikan ke dalam bentuk *plaintext* untuk mendapatkan *password* karena fungsi *hash* adalah fungsi satu arah seperti yang sudah dibahas dalam 2.4. Sementara itu, kekurangan dari skema ini adalah *digest* yang disimpan masih rentan terhadap *dictionary attack*. Penjelasan tentang *dictionary attack* akan dijelaskan pada skema selanjutnya.

Skema 3

Dalam skema 3, sistem tetap menyimpan *username*. Password juga disimpan dalam bentuk *digest*nya. Dalam skema ini, sebelum *digest password* dibuat, *password* akan dikonkatenasi

1 dengan *salt*. *Salt* adalah sebuah *string* acak yang bisa berisi angka, huruf, atau simbol.
 2 Penggunaan *salt* disini bertujuan untuk mengurangi tingkat keberhasilan *dictionary at-*
 3 *tack*. *Dictionary attack* adalah serangan dengan mencoba semua kemungkinan *string* ma-
 4 sukkan untuk fungsi *hash* sampai menghasilkan *digest* yang sesuai. Dengan adanya penam-
 5 bahan *salt*, maka akan mengurangi kemungkinan keberhasilan dari *dictionary attack* karena
 6 banyak kemungkinan dari *string* masukan akan bertambah sehingga semakin sulit untuk
 7 mendapatkan *digest* yang sesuai.
 8 Karena *salt* dibutuhkan untuk mengurangi tingkat keberhasilan *dictionary attack*, nilai
 9 *salt* akan disimpan juga dalam tabel basis data. Kemudian, saat pengguna sudah memasuk-
 10 kan *username* dan *password*, sistem akan menerima *password* yang dimasukan. Selanjutnya,
 11 *password* dikonkatenasi dengan *salt* yang disimpan lalu sistem akan menghitung *digest* dari
 12 hasil konkatenasi *password* dengan *salt*. Setelah itu, sistem akan membandingkan dengan
 13 *digest* yang disimpan dalam tabel basis data. Jika sesuai, pengguna akan diberikan hak
 14 akses ke sistem. Gambar 2.11 menunjukkan proses yang dijelaskan.



Gambar 2.11: *Password salting*

15 Kelebihan dari skema ini adalah *password* tidak akan bisa diketahui dengan mudah
 16 lewat *dictionary attack*. Banyak kemungkinan *digest* yang semakin bertambah menyebabkan
 17 serangan dengan *dictionary attack* semakin sulit. Sementara itu, kekurangan dari skema ini
 18 adalah rumit karena membutuhkan banyak proses hanya untuk memberikan akses.

19 2.7 Eliminasi Gauss-Jordan

20 Eliminasi Gauss-Jordan adalah suatu metode untuk menyelesaikan sistem persamaan linear
 21 dengan mereduksi matriks menjadi eselon baris tereduksi[3]. Suatu matriks R dikatakan
 22 bentuk eselon baris tereduksi jika memenuhi syarat sebagai berikut[3].

- 23 1. Terdapat baris yang tidak seluruhnya terdiri dari angka 0
- 24 Angka bukan 0 pertama dari sebelah kiri dari baris tersebut disebut 1 utama.
- 25 2. Baris yang seluruhnya terdiri dari angka 0 harus menjadi baris paling bawah.
- 26 3. Pada kolom 1 utama, seluruh angka di bawah 1 utama harus 0.

27 Sebagai contoh, matriks-matriks eselon baris tereduksi ditunjukkan oleh Matriks 2.27.

$$\begin{bmatrix} 1 & 12 & 5 & 4 \\ 0 & 2 & 4 & 8 \\ 0 & 0 & 9 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 & 5 \\ 0 & 5 & 4 & 8 \\ 0 & 0 & 4 & 10 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.27)$$

Proses eliminasi Gauss-Jordan dibagi menjadi 2 proses, yaitu proses mereduksi matriks menjadi bentuk eselon baris dan proses substitusi balik ke sistem persamaan linear untuk memperoleh solusi sistem persamaan linear. Diasumsikan sistem persamaan linear yang akan dicari solusinya ditunjukkan oleh persamaan 2.28. Berikut akan dijelaskan proses mereduksi matriks menjadi bentuk eselon baris.

$$\begin{aligned}x + y + z &= 10 \\x + 2y + 4z &= 21 \\x + 3y + 9z &= 38\end{aligned}\tag{2.28}$$

2.7.1 Proses Reduksi Matriks

Proses mereduksi matriks menjadi eselon baris dilakukan dengan cara operasi baris. Operasi baris adalah suatu metode untuk mereduksi matriks menjadi eselon baris dengan cara sebagai berikut.

1. Mengalikan baris dengan konstanta selain 0.
2. Menukar 2 baris.
3. Mengurangi sebuah baris dengan baris lainnya.

Sebagai contoh, diasumsikan bentuk matriks dari persamaan 2.28 ditunjukkan oleh Matriks 2.29.

$$\begin{bmatrix} 1 & 1 & 1 & 10 \\ 1 & 2 & 4 & 21 \\ 1 & 3 & 9 & 38 \end{bmatrix}\tag{2.29}$$

Operasi baris pertama adalah mengurangi baris ke-3 dan baris ke-2 dengan baris ke-1. Maka, hasil pengurangan baris ditunjukkan oleh Matriks 2.30.

$$\begin{bmatrix} 1 & 1 & 1 & 10 \\ 1-1 & 2-1 & 4-1 & 21-10 \\ 1-1 & 3-1 & 9-1 & 38-10 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 10 \\ 0 & 1 & 3 & 11 \\ 0 & 2 & 8 & 28 \end{bmatrix}\tag{2.30}$$

Kemudian, operasi baris kedua adalah mengurangi baris ke-3 dengan baris ke-2 yang dikali dengan konstanta 2. Hasil operasi baris kedua ditunjukkan oleh Matriks 2.31.

$$\begin{bmatrix} 1 & 1 & 1 & 10 \\ 0 & 1 & 3 & 11 \\ 0 & 2-(1 \cdot 2) & 8-(3 \cdot 2) & 28-(11 \cdot 2) \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 10 \\ 0 & 1 & 3 & 11 \\ 0 & 0 & 2 & 6 \end{bmatrix}\tag{2.31}$$

Setelah operasi baris kedua, maka diperoleh Matriks 2.32 yang merupakan matriks dengan bentuk eselon baris tereduksi.

$$\begin{bmatrix} 1 & 1 & 1 & 10 \\ 0 & 1 & 3 & 11 \\ 0 & 0 & 2 & 6 \end{bmatrix} \quad (2.32)$$

2.7.2 Proses Substitusi Balik

Setelah mengubah matriks menjadi bentuk eselon baris tereduksi, proses substitusi balik adalah proses untuk mencari nilai koefisien dari masing-masing variabel untuk memperoleh solusi dari persamaan linear 2.28. Kolom paling kanan (kolom ke- n) dari matriks menunjukkan nilai solusi dari masing-masing baris. Sementara itu, kolom ke-1 sampai kolom ke- $(n-1)$ menunjukkan koefisien dari persamaan linear.

Sebagai contoh, dari Matriks 2.32 diperoleh hasilnya sebagai berikut.

$$\begin{aligned} 2z &= 6 \\ z &= 3 \end{aligned} \quad (2.33)$$

Kemudian, untuk nilai y .

$$\begin{aligned} y + 3z &= 11 \\ y + 3 \cdot 3 &= 11 \\ y + 9 &= 11 \\ y &= 2 \end{aligned} \quad (2.34)$$

Kemudian, untuk nilai x .

$$\begin{aligned} x + y + z &= 10 \\ x + 2 + 3 &= 10 \\ x + 5 &= 10 \\ x &= 5 \end{aligned} \quad (2.35)$$

Jadi, solusi dari persamaan 2.28 yang diselesaikan dengan eliminasi Gauss-Jordan adalah $x = 5$, $y = 2$, dan $z = 3$.

2.8 Secret Sharing Shamir

Pada bagian ini akan dijelaskan mengenai sejarah singkat yang mengawali munculnya secret sharing Shamir dan pembahasan mengenai secret sharing Shamir.

2.8.1 Sejarah Singkat

Secret sharing adalah metode untuk membagi informasi (rahasia) menjadi beberapa bagian. Bagian-bagian tersebut disebut *share* dan setiap bagian dibagikan kepada beberapa partisipan. Untuk mendapatkan kembali informasi, maka dibutuhkan setiap *share*.

Permasalahan muncul jika *share* dan partisipan bertambah banyak. Proses untuk mendapatkan kembali rahasia akan menjadi sulit karena setiap *share* harus ada. Karena permasalahan ini, pada tahun 1979 Adi Shamir memublikasikan pengembangan dari metode *secret*

1 *sharing* dalam esai yg berjudul '*How to Share a Secret*'[4]. Metode yang dikembangkan Adi
2 Shamir dinamakan *secret sharing* Shamir.

3 2.8.2 Pembahasan *Secret Sharing* Shamir

4 Untuk mengatasi permasalahan yang sudah dibahas, Shamir mengubah cara untuk menda-
5 patkan kembali informasi. Misalkan, informasi diasumsikan sebagai data D . Dalam metode
6 *secret sharing* Shamir data D yang dibagi menjadi n *share* hanya memerlukan minimal k
7 *share* untuk memperoleh kembali D . Skema yang dikembangkan Shamir ini dinamakan
8 skema *threshold*(k, n),

9 Skema *Threshold*(k, n)

10 Skema *threshold*(k, n) adalah skema *secret sharing* dimana hanya minimal k *share* dari n
11 *share* dibutuhkan untuk mengembalikan data D . Skema ini memiliki ketentuan sebagai
12 berikut[4].

- 13 • Jika *share* yang dimiliki sebanyak k *share* atau lebih, D bisa dibentuk kembali.
- 14 • Jika *share* yang ada hanya sebanyak $k-1$ atau kurang maka D tidak bisa dibentuk
15 kembali.

16 Ada 2 proses dalam skema *threshold*(k, n), yaitu proses pembangunan *share* dari rahasia
17 dan proses rekonstruksi rahasia dari *share* yang dimiliki. Diasumsikan rahasia adalah D .
18 Proses pertama adalah proses pembangunan *share* dari D . Berikut akan dijelaskan proses
19 pembangunan *share*.

20 Proses Pembangunan *Share*

21 Langkah pertama adalah memilih nilai k . Kemudian, setelah memilih nilai k langkah selan-
22 jutnya adalah membentuk $k - 1$ derajat fungsi $f(x)$. Persamaan 2.36 menunjukkan fungsi
23 $f(x)$ yang dibentuk.

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1} \quad (2.36)$$

24 dimana $a_0 = D$.

25 Setelah membentuk fungsi $f(x)$, langkah selanjutnya adalah memilih banyak *share*, yaitu
26 nilai n . Setelah memilih n , $x = 1$ sampai $x = n$ akan dipetakan dengan fungsi $f(x)$ untuk
27 memperoleh D_i . Persamaan 2.37 menunjukkan hasil pemetaan dengan fungsi $f(x)$.

$$D_1 = f(1), D_2 = f(2), \dots, D_i = f(i), \dots, D_n = f(n) \quad (2.37)$$

28 Nilai D_1 sampai D_n adalah *share* dari data D .

29 Proses Rekonstruksi Rahasia

30 Pada bagian ini akan dijelaskan proses rekonstruksi D dari D_1, D_2, \dots, D_n yang sudah diba-
31 ngun dalam Proses Pembangunan *Share*. Langkah pertama adalah membentuk membentuk
32 $k - 1$ derajat fungsi $f(x)$ dari k yang sudah dipilih dalam Proses Pembangunan *Share*.
33 Persamaan 2.38 menunjukkan fungsi $f(x)$ yang dibentuk.

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1} \quad (2.38)$$

34 Setelah itu, langkah selanjutnya adalah membentuk k fungsi $f(x)$ dengan memetakan k
35 *share* yang dimiliki dengan fungsi $f(x)$. Hasil pemetaan dengan fungsi $f(x)$ ini adalah *share*

- 1 yang sudah dibangun pada Proses Pembangunan *Share*. Persamaan 2.39 menunjukkan hasil
 2 pemetaan masing-masing fungsi $f(x)$.

$$\begin{aligned}
 f(1) &= a_0 + a_1 \cdot 1 + a_2 \cdot 1^2 + \dots + a_{k-1} \cdot 1^{k-1} = D_1 \\
 f(2) &= a_0 + a_1 \cdot 2 + a_2 \cdot 2^2 + \dots + a_{k-1} \cdot 2^{k-1} = D_2 \\
 &\vdots \\
 f(k) &= a_0 + a_1 \cdot k + a_2 \cdot k^2 + \dots + a_{k-1} \cdot k^{k-1} = D_k
 \end{aligned} \tag{2.39}$$

- 3 Dari hasil pemetaan yang ditunjukkan persamaan 2.39, langkah selanjutnya adalah mem-
 4 bentuk persamaan linear. persamaan 2.40 menunjukkan persamaan linear yang dibentuk.

$$\begin{aligned}
 a_0 + a_1 \cdot 1 + a_2 \cdot 1^2 + \dots + a_{k-1} \cdot 1^{k-1} &= D_1 & \dots \textcircled{1} \\
 a_0 + a_1 \cdot 2 + a_2 \cdot 2^2 + \dots + a_{k-1} \cdot 2^{k-1} &= D_2 & \dots \textcircled{2} \\
 &\vdots \\
 a_0 + a_1 \cdot k + a_2 \cdot k^2 + \dots + a_{k-1} \cdot k^{k-1} &= D_k & \dots \textcircled{k}
 \end{aligned} \tag{2.40}$$

- 5 Setelah membentuk persamaan linear, langkah selanjutnya adalah menyelesaikan persa-
 6 maan linear tersebut dengan metode Eliminasi Gauss-Jordan yang sudah dijelaskan pada
 7 Subbab 2.7. Tujuannya adalah untuk memperoleh nilai a_1, a_2, \dots, a_{k-1} kemudian bisa dipe-
 8 roleh nilai a_0 yang adalah data D .

9 2.9 Probabilitas

- 10 Probabilitas atau peluang merupakan salah cara dalam ilmu matematika untuk mengukur
 11 tingkat kepercayaan akan suatu kejadian. Teori probabilitas sangat luas penggunaannya,
 12 baik dalam kehidupan sehari-hari maupun dalam percobaan-percobaan ilmiah. Teori pro-
 13 babilitas ini seringkali digunakan oleh para pengambil keputusan untuk memprediksi suatu
 14 kejadian sehingga nantinya bisa mengambil keputusan yang tepat.

- 15 Seluruh kemungkinan keluaran yang akan terjadi dalam probabilitas disebut ruang sam-
 16 pel sedangkan masing-masing kemungkinan yang dapat terjadi dalam ruang sampel dina-
 17 makan elemen kejadian atau anggota dari ruang sampel. Ruang sampel dilambangkan de-
 18 ngan huruf S dan elemen kejadian dilambangkan dengan huruf x_i . Dalam ruang sampel S
 19 dengan i elemen kejadian, ditunjukkan pada persamaan 2.41.

$$S = x_1, x_2, x_3, \dots, x_i \tag{2.41}$$

- 20 Sedangkan probabilitas kejadian x_i akan terjadi dilambangkan dengan $P(x_i)$. Maka,
 21 rumus matematikanya ditunjukkan pada persamaan 2.42.

$$P(x_i) = \frac{n}{N} \tag{2.42}$$

- 22 dimana n adalah banyaknya kemunculan kejadian x_i dalam sebuah ruang sampel S dan N
 23 adalah banyaknya kejadian yang terjadi dalam ruang sampel S .

24 2.9.1 Distribusi Binom

- 25 Setiap eksperimen atau percobaan yang dilakukan secara berkali-kali pasti memiliki dua ke-
 26 luaran, yaitu sukses atau gagal. Untuk setiap keluaran yang diperoleh (baik sukses maupun
 27 gagal) bisa ditetapkan sebagai sukses. Proses ini dinamakan proses Bernouli dan setiap eks-
 28 perimen yang dilakukan untuk setiap proses bernouli dinamakan percobaan Bernouli. Ada

1 beberapa syarat sebuah eksperimen bisa dinamakan percobaan Bernouli[5]:

- 2 1. Eksperimen harus diulang sebanyak n kali.
- 3 2. Hasil keluaran setiap perulangan hanya 2 kemungkinan, yaitu keluaran sukses atau
- 4 keluaran gagal.
- 5 3. Hasil keluaran setiap perulangan tidak mempengaruhi dengan perulangan yang lain.
- 6 4. Probabilitas bahwa hasil keluarannya sukses, p , harus selalu sama untuk setiap kali
- 7 perulangan.

8 Percobaan Bernouli digunakan untuk menghitung probabilitas x buah hasil keluaran
 9 yang sukses dari n percobaan. Diasumsikan bahwa probabilitas hasil keluaran setiap per-
 10 ulangan sukses adalah p . Sebaliknya, probabilitas hasil keluaran setiap perulangan gagal
 11 adalah $q = 1 - p$. Persamaan 2.43 untuk menghitung probabilitas x hasil keluaran yang
 12 sukses dari n percobaan.

$$P(x, n, p) = \binom{n}{x} p^x q^{n-x} \quad (2.43)$$

$$x = 0, 1, 2, \dots, n$$

13 $\binom{n}{x}$ pada persamaan 2.43 menunjukkan bahwa dari n percobaan akan dipilih x hasil
 14 keluaran yang sukses.

15 2.10 Entropi

16 Pada bagian ini akan dijelaskan mengenai entropi dimulai dari sejarah singkat entropi dan
 17 pembahasan mengenai entropi.

18 2.10.1 Sejarah Singkat

19 Istilah entropi muncul pertama kali dalam esai 'A Mathematical Theory of Communication'
 20 pada tahun 1948. Esai ini dibuat oleh Claude E. Shannon seorang ilmuwan asal Amerika
 21 Serikat. Dalam esainya, Shannon menulis bahwa entropi adalah konsep keacakan atau suatu
 22 ketidakpastian[6]. Istilah dari entropi ini dinamakan Shannon *Entropy*.

23 2.10.2 Pembahasan

24 Entropi adalah rata-rata suatu informasi yang dimiliki oleh sebuah pesan. Informasi yang
 25 dimaksud adalah kejadian yang spesifik atau sebuah elemen tertentu yang dimiliki oleh
 26 pesan. Maka dari itu, entropi bisa dijadikan alat ukur ketidakpastian yang dimiliki oleh
 27 sebuah pesan atau sumber informasi.

28 Nilai entropi yang tinggi menunjukkan bahwa informasi yang dimiliki sebuah pesan cu-
 29 kup tinggi. Nilai informasi yang cukup tinggi memiliki arti bahwa isi dari pesan bisa dip-
 30 rediksi. Sementara itu, jika nilai entropi yang rendah menunjukkan bahwa informasi yang
 31 dimiliki sebuah pesan cukup rendah. Nilai informasi yang cukup rendah memiliki arti bahwa
 32 isi dari pesan tidak bisa dengan mudah diprediksi.

33 Sebagai contoh, nilai entropi akan rendah untuk memastikan panjang umur seseorang
 34 karena tidak bisa diketahui kapan orang tersebut akan meninggal. Contoh yang lain adalah
 35 nilai entropi akan tinggi untuk kasus melemparkan koin karena hasilnya hanya ada dua
 36 kemungkinan yaitu, kepala atau buntut.

1 Dari penjelasan mengenai entropi yang sudah dijelaskan, diasumsikan probabilitas ke-
2 munculan informasi x_i dalam sebuah pesan X adalah p_i . p_i ditunjukkan oleh persamaan
3 2.44.

$$P(x_i) = p_i \log\left(\frac{1}{p_i}\right) \quad (2.44)$$

4 Maka, nilai entropi pesan X untuk setiap informasi p_1, p_2, \dots, p_m ditunjukkan oleh per-
5 samaan 2.45.

$$\begin{aligned} H(X) &= p_1 \log\left(\frac{1}{p_1}\right) + p_2 \log\left(\frac{1}{p_2}\right) + \dots + p_m \log\left(\frac{1}{p_m}\right) \\ &= \sum_{i=1}^m p_i \log\left(\frac{1}{p_i}\right) \end{aligned} \quad (2.45)$$

BAB 3

ANALISIS

Pada bab ini akan dibahas analisis terhadap teori-teori yang telah dibahas sebelumnya. Analisis akan meliputi studi kasus untuk penerapan metode *secret sharing* Shamir, pemilihan n dan k , dan perancangan perangkat lunak.

3.1 Studi Kasus

Pada bagian ini akan dibahas studi kasus tentang bagaimana penerapan metode *secret sharing* Shamir untuk banyak password. Studi kasus meliputi pengenalan kasus, pembangunan share, dan rekonstruksi rahasia.

3.1.1 Pengenalan Kasus

Langkah awal yang dibutuhkan untuk mengembalikan banyak *password* dengan metode *secret sharing* Shamir, diperlukan beberapa tahap proses. Proses pertama adalah proses penyimpanan *password*. Kemudian, proses selanjutnya adalah proses untuk mengembalikan banyak *password*. Proses pertama membutuhkan beberapa *password*. Untuk n buah *password*, maka akan dibuat n buah pertanyaan keamanan. Sementara itu, untuk proses mengembalikan *password* dibutuhkan pertanyaan keamanan yang sudah dibuat dalam proses sebelumnya.

Untuk kedua proses di atas, diasumsikan banyak *password* yang akan disimpan sebanyak 5 buah. Setiap *password* akan diberi label p_1 , p_2 , sampai p_5 . Persamaan 3.1 sampai 3.5 menunjukkan p_1 sampai p_5 .

$$p_1 = 123456 \quad (3.1)$$

$$p_2 = \text{password} \quad (3.2)$$

$$p_3 = \text{hello123} \quad (3.3)$$

$$p_4 = \text{secret} \quad (3.4)$$

$$p_5 = \text{foobar} \quad (3.5)$$

3.1.2 Proses Penyimpanan *Password*

Proses penyimpanan *password* dibagi menjadi 2 proses, yaitu proses pembangunan *share* untuk masing-masing *password* dan proses enkripsi dari setiap *share* yang sudah dibangun. Pada bagian ini akan dibahas kedua proses tersebut.

Proses Pembangunan *Share*

Pada proses ini, akan dilakukan pembangunan *share* dari masing-masing *password*. Langkah-langkah untuk membangun *share* adalah sebagai berikut.

1. Membagi setiap *password* p_i menjadi beberapa karakter, masing-masing karakter akan diubah menjadi nilai ASCIInya, c_1, c_2, \dots, c_m .

2. Memilih nilai n , yaitu banyak *share* yang akan dibangun.
3. Memilih nilai k , yaitu banyak minimal pertanyaan keamanan yang harus dijawab dengan benar, dimana $0 < k \leq n$.
4. Memilih $k - 1$ angka acak, d_1, d_2, \dots, d_{k-1} , untuk masing-masing karakter c_1, c_2, \dots, c_m .
5. Membentuk fungsi $f_m(x)$ untuk masing-masing karakter c_1, c_2, \dots, c_m . Komponen dari fungsi $f_m(x)$ terdiri atas c_m sebagai konstanta tanpa koefisien, d_1, d_2, \dots, d_{k-1} sebagai konstanta dengan koefisien. Persamaan 3.6 menunjukkan persamaan dari fungsi $f_m(x)$ yang harus dibentuk.

$$f_m(x) = c_m + d_1x + d_2x^2 + d_3x^3 + \dots + d_{k-1}x^{k-1} \quad (3.6)$$

6. Menghitung masing-masing nilai x dari $x = 1, x = 2, \dots, x = n$ untuk fungsi $f_m(x)$.
7. Nilai $f_m(1)$ sampai $f_m(n)$ adalah nilai *share* untuk password p_i .

Kembali kepada kasus pada Subbab 3.1.1, misalkan *password* yang akan dibangun *share*-*sharenya* adalah p_1 . Langkah pertama adalah membagi p_1 menjadi beberapa karakter dan mengubah masing-masing karakter menjadi nilai ASCII-nya. Persamaan 3.7 sampai 3.13 menunjukkan langkah pertama.

$$p_1 = 123456 \quad (3.7)$$

$$c_1 = '1' = 49 \quad (3.8)$$

$$c_2 = '2' = 50 \quad (3.9)$$

$$c_3 = '3' = 51 \quad (3.10)$$

$$c_4 = '4' = 52 \quad (3.11)$$

$$c_5 = '5' = 53 \quad (3.12)$$

$$c_6 = '6' = 54 \quad (3.13)$$

Langkah selanjutnya adalah memilih nilai n . Karena banyak *password* p_i adalah 5, maka banyak *share* untuk masing-masing *password* sebanyak 5. Maka, $n = 5$.

Setelah memilih nilai n , langkah berikutnya adalah memilih nilai k . Nilai k ini nanti akan berhubungan dengan banyak minimal pertanyaan keamanan yang harus dijawab benar untuk mengembalikan *password*. Untuk kasus ini, dipilih $k = 3$.

Langkah selanjutnya adalah memilih $k - 1$ angka acak untuk masing-masing karakter c_1 sampai c_6 . Karena $k = 3$, maka dipilih 2 angka acak untuk masing-masing karakter. Berikut angka acak untuk masing-masing karakter.

- c_1 : 12 dan 6.
- c_2 : 15 dan 11.
- c_3 : 22 dan 1.
- c_4 : 21 dan 3.
- c_5 : 19 dan 8.
- c_6 : 25 dan 17.

Setelah memilih angka acak untuk masing-masing karakter, langkah selanjutnya adalah membentuk fungsi $f(x)$ untuk masing-masing karakter. Maka, fungsi $f_1(x)$ sampai $f_6(x)$ yang dibentuk adalah sebagai ditunjukkan pada persamaan 3.14 sampai 3.19.

$$f_1(x) = 49 + 12x + 6x^2 \quad (3.14)$$

$$f_2(x) = 50 + 15x + 11x^2 \quad (3.15)$$

$$f_3(x) = 51 + 22x + x^2 \quad (3.16)$$

$$f_4(x) = 52 + 21x + 3x^2 \quad (3.17)$$

$$f_5(x) = 53 + 19x + 8x^2 \quad (3.18)$$

$$f_6(x) = 54 + 25x + 17x^2 \quad (3.19)$$

Langkah selanjutnya adalah menghitung nilai $x = 1, x = 2, \dots, x = n$ untuk fungsi $f_1(x)$ sampai $f_6(x)$. Tabel 3.1 menunjukkan nilai $x = 1$ sampai $x = 5$ untuk masing-masing fungsi $f(x)$.

Tabel 3.1: Nilai x untuk masing-masing $f(x)$

	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
1	67	76	74	76	80	96
2	97	124	99	106	123	172
3	139	194	126	142	182	282
4	193	286	155	184	257	426
5	259	400	186	232	348	604

Setiap nilai x pada Tabel 3.1 adalah nilai *share-share* untuk password p_1 . Setiap nilai x ini akan diberi label s_{11} untuk *share* pertama dari fungsi pertama, s_{21} untuk *share* kedua dari fungsi pertama, dan seterusnya sampai s_{56} untuk *share* kelima dari fungsi keenam. Nilai *share* yang sudah diberi label ditunjukkan pada persamaan 3.20.

$$s_{11} = 67, s_{21} = 97, \dots, s_{34} = 155, \dots, s_{56} = 604 \quad (3.20)$$

Sementara itu, untuk menghitung nilai *share* dari password p_2 sampai p_5 , proses yang sama untuk menghitung password p_1 akan dilakukan. Setelah menghitung nilai *share* untuk password p_1 , langkah selanjutnya adalah proses enkripsi masing-masing *share* ini.

Proses Enkripsi Share

Pada proses ini, sebelum masing-masing *share* disimpan, masing-masing *share* harus dienkripsi terlebih dahulu. Dalam proses ini juga, n buah pertanyaan keamanan akan dibuat. Langkah-langkah proses enkripsi *share* adalah sebagai berikut.

1. Membuat n pertanyaan keamanan, q_1, q_2, \dots, q_n .
2. Menentukan jawaban dari masing-masing pertanyaan keamanan, a_1, a_2, \dots, a_n .
3. Menentukan nilai *salt*, r_s .
4. Menghitung *digest* untuk masing-masing konkatenasi dari pertanyaan, jawaban, dan *salt*. Persamaan 3.21 menunjukkan proses menghitung *digest*.

$$h_n = H(q_n + a_n + r_s) \quad (3.21)$$

5. Setiap nilai *share*, $s_{11}, s_{21}, \dots, s_{56}$ akan dienkripsi dengan menggunakan *digest* sebagai kunci. Persamaan 2.1 menunjukkan langkah enkripsi *share*.

$$E_{h_n}(s_{nm}) = c_{nm} \quad (3.22)$$

1 Pada persamaan 2.1, m merupakan banyak karakter dari masing-masing *password* p_i .

2 Kembali kepada kasus pada Subbab 3.1.1, misalkan *password* yang akan dienkripsi *share-*
 3 *share*nya adalah p_1 . Langkah pertama adalah membuat n pertanyaan keamanan, karena
 4 $n = 5$ maka ada 5 pertanyaan keamanan. Setiap pertanyaan keamanan akan diberi label
 5 q_1, q_2, \dots, q_5 . Untuk kasus ini, diasumsikan pertanyaan keamanan yang dibuat adalah sebagai
 6 berikut.

- 7 1. Siapa nama anda? (q_1)
- 8 2. Dimana kota tempat anda tinggal? (q_2)
- 9 3. Apa jenis kelamin anda? (q_3)
- 10 4. Pada bulan apa anda lahir? (q_4)
- 11 5. Apa nama belakang anda? (q_5)

12 Setelah membuat pertanyaan keamanan yang akan digunakan, langkah selanjutnya ada-
 13 lah menentukan jawaban dari masing-masing pertanyaan keamanan. Setiap jawaban untuk
 14 pertanyaan keamanan akan diberi label a_1 untuk q_1 , a_2 untuk q_2 , dan seterusnya sampai a_5
 15 untuk q_5 . Jawaban dari masing-masing pertanyaan keamanan adalah sebagai berikut.

- 16 1. Samuel (a_1)
- 17 2. Bandung (a_2)
- 18 3. Laki-laki (a_3)
- 19 4. Juli (a_4)
- 20 5. Christian (a_5)

21 Langkah selanjutnya adalah memilih nilai *salt*, r_s . Untuk kasus ini, misalkan $r_s = 31$.

22 Setelah memilih nilai *salt*, langkah selanjutnya adalah menghitung *digest*. Masing-masing
 23 dari pertanyaan keamanan akan dikonkatenasi dengan jawabannya dan r_s . Asumsi hasil
 24 penghitungan *digest*, h_n , untuk setiap pertanyaan ditunjukkan pada persamaan 3.23 sampai
 25 3.27.

$$h_1 = (q_1 + a_1 + r_s) = 7a916 \quad (3.23)$$

$$h_2 = (q_2 + a_2 + r_s) = cdc62 \quad (3.24)$$

$$h_3 = (q_3 + a_3 + r_s) = de09b \quad (3.25)$$

$$h_4 = (q_4 + a_4 + r_s) = d1320 \quad (3.26)$$

$$h_5 = (q_5 + a_5 + r_s) = b59e9 \quad (3.27)$$

26 Langkah selanjutnya adalah mengenkripsi setiap nilai *share* yang sudah dibangun dengan
 27 *digest* yang sudah dihitung sebagai kuncinya. Asumsi hasil enkripsi setiap *share* untuk p_1 ,
 28 ditunjukkan pada Tabel 3.2.

Tabel 3.2: Hasil Enkripsi setiap *Share* untuk *Password* Pertama

	$E(f_1)$	$E(f_2)$	$E(f_3)$	$E(f_4)$	$E(f_5)$	$E(f_6)$
h_1	aa7cm	a45sf	1xz5q	x15z6	cx96v	6zx51
h_2	ff3ds	5cv1s	rf51s	xcq89	a9er8	9wrt8
h_3	fg9e5	afa65	ge65r	we65q	s6dv5	xf8xj
h_4	d3d64	eq89v	85vbn	nm6f5	51gvq	x91qw
h_5	a54q1	z1x56	as46c	na6e5	cz98q	ha658

Angka yang ditunjuk oleh kolom $E(f_1)$ dan baris h_1 adalah hasil enkripsi untuk share pertama dari fungsi pertama. Sementara itu, angka yang ditunjuk oleh kolom $E(f_2)$ dan baris h_1 adalah hasil enkripsi untuk share pertama dari fungsi kedua dan seterusnya.

Langkah enkripsi setiap *share* ini dilakukan untuk setiap *password* p_2 sampai p_5 . Kemudian setelah proses enkripsi ini, pertanyaan keamanan, jawaban, hasil enkripsi (*ciphertext*), nilai *salt*, dan nilai k akan disimpan.

3.1.3 Proses Pengembalian *Password*

Setelah *password* disimpan dalam proses Penyimpanan *Password* (Subbab 3.1.2), pada bagian ini akan dijelaskan proses bagaimana *password* bisa dikembalikan dengan menggunakan metode *secret sharing* Shamir. Proses pengembalian *password* ini dibagi menjadi 2 proses, yaitu proses dekripsi setiap *share* dan proses rekonstruksi kembali *password* dari *share-share* yang sudah didekripsi.

Proses Dekripsi *Share*

Proses dekripsi *share* adalah proses mengembalikan *ciphertext* dari masing-masing *share* kembali kepada bentuk *plaintext*nya. Langkah-langkah dari proses dekripsi *share* adalah sebagai berikut.

1. Menjawab n pertanyaan keamanan yang sebelumnya disimpan, q_1, q_2, \dots, q_n untuk menghasilkan jawaban a'_1, a'_2, \dots, a'_n .
2. Menghitung *digest* untuk masing-masing konkatenasi dari pertanyaan yang disimpan, jawaban, dan *salt* yang disimpan. Persamaan 3.28 menunjukkan proses menghitung *digest*.

$$h'_n = H(q_n + a'_n + r_s) \quad (3.28)$$

3. Mendekripsi $c_{11}, c_{21}, \dots, c_{nm}$ dengan menggunakan h'_1, h'_2, \dots, h'_n sebagai kunci. Persamaan 3.29 menunjukkan langkah yang dijelaskan.

$$D_{h'_n}(c_{nm}) = s'_{nm} \quad (3.29)$$

Kembali kepada kasus yang dijelaskan pada Subbab 3.1.1, langkah pertama adalah menjawab pertanyaan keamanan yang sebelumnya disimpan. Berikut pertanyaan keamanan yang disimpan dan jawaban untuk masing-masing pertanyaan keamanan.

1. Siapa nama anda? (q_1): Samuel (a'_1)
2. Dimana kota tempat anda tinggal? (q_2): Bandung (a'_2)
3. Apa jenis kelamin anda? (q_3): Laki-laki (a'_3)
4. Pada bulan apa anda lahir? (q_4): Juli (a'_4)
5. Apa nama belakang anda? (q_5): Christian (a'_5)

Kemudian, langkah selanjutnya adalah menghitung *digest* masing-masing konkatenasi dari pertanyaan yang disimpan, jawaban, dan *salt* yang disimpan, $r_s = 31$. Asumsi hasil penghitungan *digest*, h'_n , untuk setiap pertanyaan ditunjukkan pada persamaan 3.30 sampai 3.34.

$$h'_1 = (q_1 + a'_1 + r_s) = 7a916 \quad (3.30)$$

$$h'_2 = (q_2 + a'_2 + r_s) = cdc62 \quad (3.31)$$

$$h'_3 = (q_3 + a'_3 + r_s) = de09b \quad (3.32)$$

$$h'_4 = (q_4 + a'_4 + r_s) = d1320 \quad (3.33)$$

$$h'_5 = (q_5 + a'_5 + r_s) = b59e9 \quad (3.34)$$

1 Setelah memperoleh *digest*, langkah selanjutnya adalah mendekripsi setiap *share* dalam
 2 Tabel 3.2 dengan *digest* h'_1, h'_2, \dots, h'_5 sebagai kunci. Persamaan 3.35 dan 3.36 menunjukkan
 3 langkah dari dekripsi salah satu *share*.

$$c_{11} = aa7cm \quad (3.35)$$

$$D_{h_1}(c_{11}) = s_{11} = 67 \quad (3.36)$$

4 Kemudian, proses dekripsi diulang untuk setiap *share* dari *password* p_1 . Tabel 3.3 me-
 5 nunjukkan hasil dari dekripsi setiap *share*.

Tabel 3.3: Hasil Dekripsi *Share*

	1	2	3	4	5	6
s_1	67	76	74	76	80	96
s_2	97	124	99	106	123	172
s_3	139	194	126	142	182	282
s_4	193	286	155	184	257	426
s_5	259	400	186	232	348	604

6 Kolom pada Tabel 3.3 menunjukkan urutan karakter dari *password* p_1 , sedangkan baris
 7 pada Tabel 3.3 menunjukkan urutan *share* dari masing-masing karakter. Sebagai contoh,
 8 baris s_1 kolom 1 menunjukkan *share* pertama untuk karakter pertama dari *password* p_1
 9 dan seterusnya sampai baris s_5 kolom 6 menunjukkan *share* kelima untuk karakter keenam
 10 *password* p_1 .

11 Proses Rekonstruksi *Password*

12 Setelah memperoleh hasil dekripsi *share* untuk masing-masing karakter dari masing-masing
 13 *password* p_1 sampai p_5 , proses selanjutnya adalah proses rekonstruksi masing-masing passwo-
 14 rd. Dalam kasus ini, password yang akan direkonstruksi adalah p_1 . Berikut langkah-langkah
 15 dari rekonstruksi p_1 .

1. Membentuk fungsi dasar $f(x)$ untuk masing-masing karakter dari password p_i berda-
 sarkan nilai k yang disimpan. Nilai k mempengaruhi derajat dari fungsi $f(x)$ yang
 akan dibentuk. Persamaan 3.37 menunjukkan fungsi $f(x)$ yang akan dibentuk.

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1} \quad (3.37)$$

2. Setiap karakter dari *password* p_i diwakili oleh 1 fungsi $f(x)$. Maka, untuk setiap
 karakter dibentuk fungsi $f_m(x)$ masing-masing, dimana m adalah banyak karakter
 dari *password* p_i . Persamaan 3.38 menunjukkan langkah yang dijelaskan.

$$f_m(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1} \quad (3.38)$$

3. Menghitung nilai *share* yang dimiliki untuk masing-masing fungsi $f(x)$ setiap karakter. Persamaan 3.39 menunjukkan langkah yang dijelaskan.

$$f_m(n) = a_0 + a_1n + a_2n^2 + \dots + a_{k-1}n^{k-1} = s_{nm} \quad (3.39)$$

4. Menghitung konstanta bebas dari berdasarkan fungsi $f(x)$ yang ada untuk masing-masing karakter, dari $f_1(x), f_2(x), \dots, f_m(x)$.
5. Mengubah konstanta bebas yang diperoleh dari langkah sebelumnya menjadi karakter ASCII.

Setelah diperoleh nilai setiap *share* yang ditunjukkan pada Tabel 3.3, langkah pertama yang dilakukan untuk mengembalikan *password* adalah membentuk fungsi dasar $f(x)$ untuk masing-masing karakter dari password p_i berdasarkan nilai k yang disimpan. Dalam kasus Subbab 3.1.1, k yang dipilih adalah $k = 3$, maka fungsi $f(x)$ yang dibentuk memiliki derajat $k - 1$. Persamaan 3.40 menunjukkan fungsi $f(x)$ yang dibentuk.

$$f(x) = c + bx + ax^2 \quad (3.40)$$

- Langkah selanjutnya adalah membentuk fungsi $f(x)$ untuk setiap karakter *password* p_1 . Persamaan 3.41 sampai 3.46 menunjukkan fungsi $f(x)$ untuk setiap karakter *password* p_1 .

$$f_1(x) = c + bx + ax^2 \quad (3.41)$$

$$f_2(x) = c + bx + ax^2 \quad (3.42)$$

$$f_3(x) = c + bx + ax^2 \quad (3.43)$$

$$f_4(x) = c + bx + ax^2 \quad (3.44)$$

$$f_5(x) = c + bx + ax^2 \quad (3.45)$$

$$f_6(x) = c + bx + ax^2 \quad (3.46)$$

- Setelah itu, langkah selanjutnya adalah menghitung nilai *share* yang dimiliki pada fungsi $f(x)$ yang sudah dibentuk. Untuk langkah ini, akan ditunjukkan proses pengembalian salah satu karakter dari *password* p_1 , yaitu karakter pertama.

Diasumsikan *share* yang digunakan untuk rekonstruksi karakter pertama adalah s_{11}, s_{21} , dan s_{31} . Maka, nilai masing-masing *share* ini pada fungsi $f_1(x)$ ditunjukkan pada persamaan 3.47 sampai 3.49.

$$f_1(1) = c + b + a = 67 \quad (3.47)$$

$$f_1(2) = c + 2b + 4a = 97 \quad (3.48)$$

$$f_1(3) = c + 3b + 9a = 139 \quad (3.49)$$

- Langkah selanjutnya adalah menghitung konstanta bebas, yaitu dalam kasus ini konstanta bebas c . Proses eliminasi Gauss-Jordan digunakan dalam menghitung konstanta bebas. Langkah pertama adalah transformasi $f_1(x), f_2(x)$, dan $f_3(x)$ menjadi matriks. Matriks 3.50 menunjukkan hasil transformasi $f_1(x), f_2(x)$, dan $f_3(x)$.

$$\begin{bmatrix} 1 & 1 & 1 & 67 \\ 1 & 2 & 4 & 97 \\ 1 & 3 & 9 & 139 \end{bmatrix} \quad (3.50)$$

Kolom paling kanan dari Matriks 3.50 menunjukkan nilai $f_1(x)$, $f_2(x)$, dan $f_3(x)$, sedangkan kolom lainnya menunjukkan nilai koefisien dari setiap variabel dalam $f_1(x)$, $f_2(x)$, dan $f_3(x)$. Kemudian, setiap baris akan diberi label. Baris 1 diberi label L_1 , baris 2 diberi label L_2 , dan baris 3 diberi label L_3 .

Setelah transformasi matriks, langkah selanjutnya adalah operasi setiap baris untuk memperoleh matriks segitiga atas. Operasi pertama yang dilakukan ditunjukkan oleh persamaan 3.51.

$$\begin{aligned} L_3 - L_1 \\ L_2 - L_1 \end{aligned} \quad (3.51)$$

Operasi pertama menghasilkan Matriks 3.52.

$$\begin{bmatrix} 1 & 1 & 1 & 67 \\ 0 & 1 & 3 & 30 \\ 0 & 2 & 8 & 72 \end{bmatrix} \quad (3.52)$$

Langkah selanjutnya adalah operasi baris kembali sampai memperoleh matriks segitiga atas. Operasi kedua ditunjukkan pada persamaan 3.53.

$$L_3 - 2L_2 \quad (3.53)$$

Operasi kedua menghasilkan Matriks 3.54.

$$\begin{bmatrix} 1 & 1 & 1 & 67 \\ 0 & 1 & 3 & 30 \\ 0 & 0 & 2 & 12 \end{bmatrix} \quad (3.54)$$

Setelah operasi kedua, diperoleh matriks segitiga atas yang ditunjukkan oleh Matriks 3.54. Langkah selanjutnya setelah memperoleh matriks segitiga atas adalah substitusi balik untuk memperoleh masing-masing nilai koefisien untuk setiap variabel a , b , dan c . Proses substitusi balik pertama adalah untuk memperoleh nilai a . Persamaan 3.55 menunjukkan proses substitusi balik pertama.

$$\begin{aligned} 2a &= 12 \\ a &= 6 \end{aligned} \quad (3.55)$$

Proses substitusi balik kedua adalah untuk memperoleh nilai b . Proses substitusi balik kedua ditunjukkan pada persamaan 3.56.

$$\begin{aligned} b + 3a &= 30 \\ b + 3 \cdot 6 &= 30 \\ b + 18 &= 30 \\ b &= 12 \end{aligned} \quad (3.56)$$

1 Proses substitusi balik ketiga adalah untuk memperoleh nilai c . Proses substitusi balik
2 ketiga ditunjukkan pada persamaan 3.57.

$$\begin{aligned} c + b + a &= 67 \\ c + 12 + 6 &= 67 \\ c + 18 &= 67 \\ c &= 49 \end{aligned} \tag{3.57}$$

3 Setelah proses substitusi balik ketiga diperoleh konstanta bebas $c = 49$ untuk karak-
4 ter pertama. Langkah selanjutnya setelah memperoleh konstanta bebas adalah mengubah
5 konstanta bebas menjadi karakter ASCII. Karakter ASCII ke-49 adalah '1'. Maka, untuk
6 karakter pertama dari p_1 adalah '1'.

7 Proses yang sama akan dilakukan untuk karakter kedua, ketiga, sampai karakter keenam.
8 Setelah semua karakter diperoleh, setiap karakter akan dikonkatenasi menjadi sebuah *string*.
9 Maka, hasil akhir dari p_1 ditunjukkan pada persamaan 3.58.

$$p_1 = 123456 \tag{3.58}$$

10 3.2 Pemilihan n dan k

11 Pengguna dapat memilih n dan k sesuai dengan kebutuhan. Pemilihan n dan k yang baik,
12 tidak hanya dapat membuat *password* tidak akan mudah dikembalikan oleh pihak yang tidak
13 berhak, tetapi dapat juga membuat pengguna bisa dengan mudah mengembalikan *passwo-*
14 *rd*[7]. Pada bagian ini, akan dijelaskan bagaimana pemilihan n dan k dapat mempengaruhi
15 kedua hal tersebut.

16 3.2.1 Pemilihan k

17 Nilai k adalah banyak minimal pertanyaan benar yang perlu dijawab agar bisa memperoleh
18 *password*. Setiap dari pertanyaan keamanan memiliki kemungkinan jawabannya masing-
19 masing. Setiap kemungkinan jawaban dari pertanyaan ini memiliki nilai entropi e_i . Per-
20 tanyaan keamanan yang memiliki kemungkinan jawaban hanya 2 (ya/tidak) memiliki nilai
21 entropi e_i yang besar sehingga mudah ditebak.

22 Maka dengan bertambahnya nilai k dan diasumsikan e_i dari setiap pertanyaan sangat
23 kecil, maka kemungkinan jawaban dari setiap pertanyaan akan bervariasi. Namun, nilai k
24 yang terlalu besar juga akan menyulitkan pemilik *password* untuk mengembalikan *password*
25 karena semakin banyak pertanyaan yang harus dijawab dengan tepat.

26 3.2.2 Pemilihan n

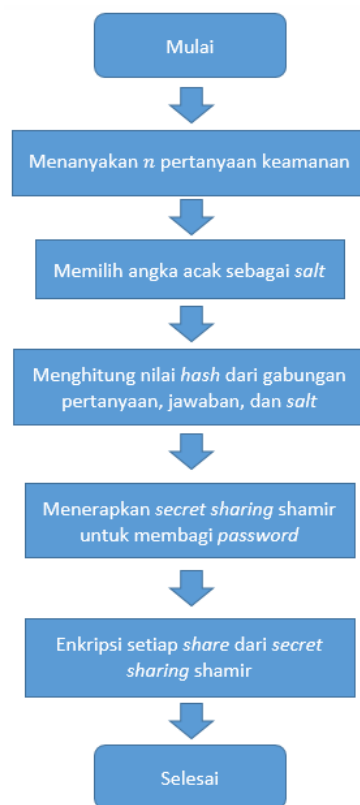
27 Nilai n adalah banyak pertanyaan keamanan yang dibuat. Pemilihan nilai n bergantung
28 pada pemilihan nilai k . Jika diasumsikan P_0 adalah probabilitas masing-masing pertanyaan
29 dijawab dengan benar, maka untuk menghitung probabilitas k pertanyaan yang benar dari
30 n pertanyaan. (?)

31 3.3 Perancangan Perangkat Lunak

32 Bagian ini akan berisi mengenai perancangan perangkat lunak yang mencakup alur proses
33 (*flowchart*) yang bisa dilakukan, diagram *use case*, dan rancangan awal diagram kelas.

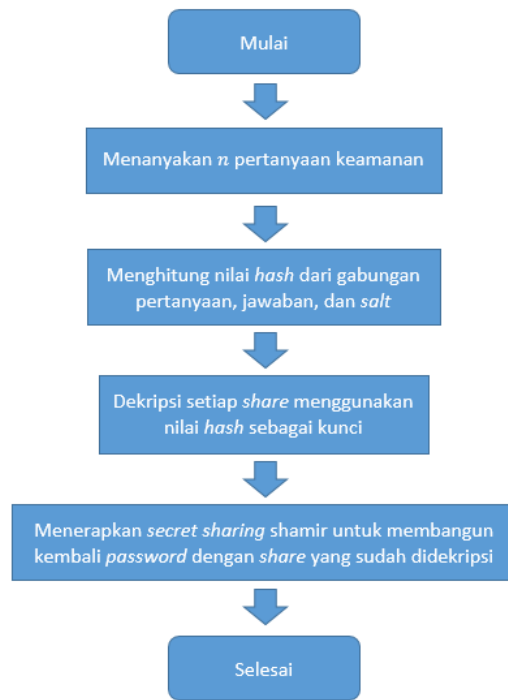
3.3.1 Alur Proses

Pada bagian ini akan dijelaskan alur proses berdasarkan proses-proses yang sudah dipaparkan sebelumnya. Proses ini akan dibagi menjadi 2 bagian, yaitu proses pembangunan *share* dari *password* dan proses pembangunan kembali atau rekonstruksi *password* dari *share-share* yang ada. Dalam alur proses ini diasumsikan bahwa n dan k sudah dipilih dengan baik dan optimal dan pesan rahasia disini adalah *password*. Gambar 3.1 menunjukkan alur proses pembangunan *share* dari *password*.



Gambar 3.1: Proses pembangunan *share* dari *password*

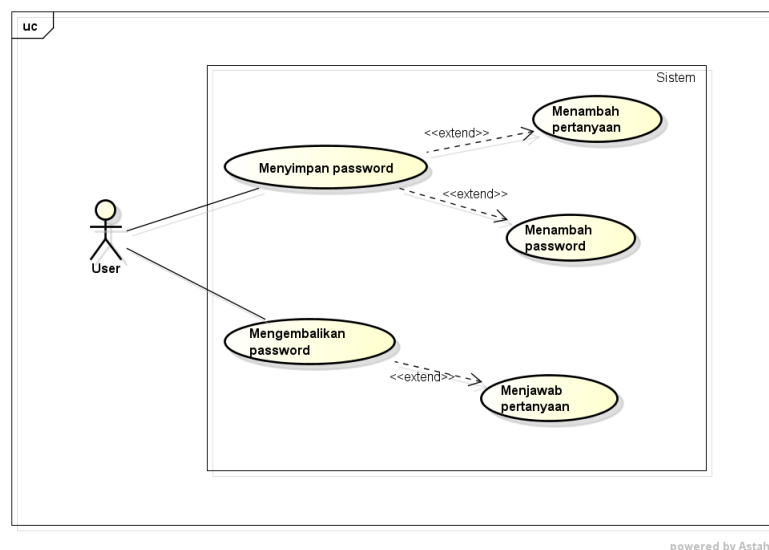
Kemudian, untuk alur proses pembangunan kembali atau rekonstruksi *password* dari *share-share* yang ada ditunjukkan oleh Gambar 3.2.



Gambar 3.2: Proses pembangunan kembali atau rekonstruksi *password*

1 3.3.2 Diagram *Use Case*

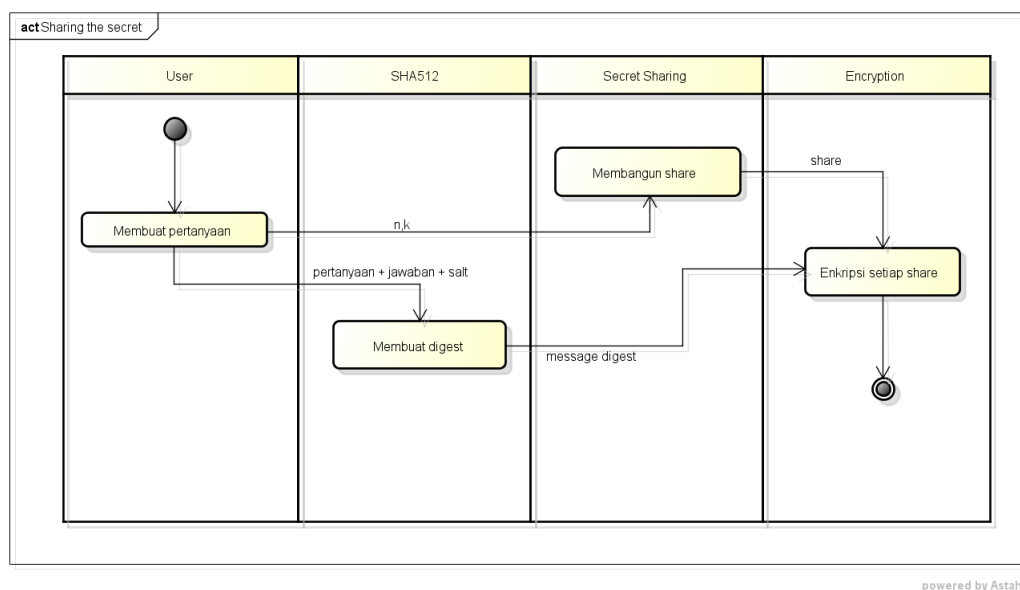
2 Perangkat lunak yang dibangun akan memiliki 2 fitur utama, yaitu menyimpan *password*
 3 beserta pertanyaan keamanan yang sifatnya personal dan mengembalikan *password*. Saat
 4 menyimpan *password*, pengguna akan diminta untuk menambahkan pertanyaan keamanan
 5 yang sifatnya personal dan saat mengembalikan *password*, pengguna akan diminta untuk
 6 menjawab pertanyaan keamanan yang sudah disimpan saat menyimpan *password*. Gambar
 7 3.3 menunjukkan diagram *use case* dari perangkat lunak.



Gambar 3.3: Diagram *use case* dari perangkat lunak

3.3.3 Diagram Aktivitas

- Perangkat lunak yang dibangun memiliki 2 proses, yaitu menyimpan *password* atau *secret* dan mengembalikan *password* atau *secret*. Gambar 3.4 menunjukkan diagram aktivitas untuk proses menyimpan *password* atau *secret*.

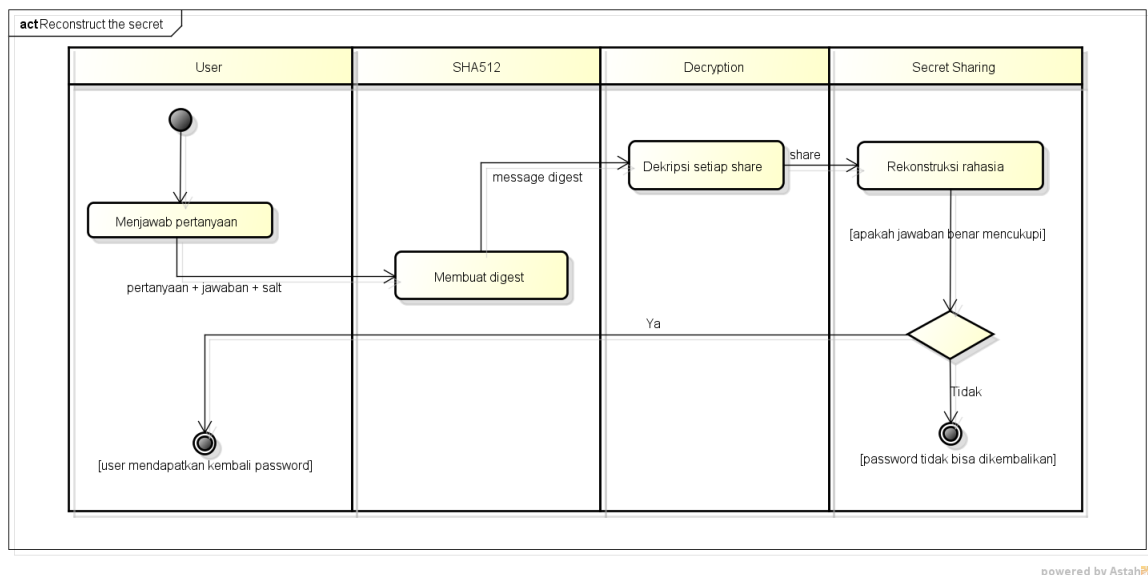


Gambar 3.4: Diagram aktivitas untuk menyimpan *password*

- Dalam proses menyimpan *password*, awalnya *user* harus terlebih dahulu menentukan banyak pertanyaan keamanan yang hendak digunakan (n) dan banyak minimal pertanyaan keamanan yang bisa dijawab dengan benar untuk memperoleh kembali *password* (k). Kemudian, *user* akan menentukan pertanyaan keamanan personal yang akan digunakan.

- Pertanyaan keamanan ini nantinya akan kembali digunakan untuk memperoleh kembali *password* yang hilang atau dilupakan. Kemudian, setelah *user* memilih dan menjawab setiap pertanyaan keamanan, setiap pertanyaan keamanan ini akan dihitung nilai *hash*-nya. Selanjutnya dengan menggunakan skema *threshold* (k, n) untuk membagi *password* menjadi sebanyak n *share*. Setiap *share* ini akan dienkripsi dengan kunci nilai *hash*.

- Selanjutnya adalah proses untuk mengembalikan *password*. Gambar 3.5 menunjukkan diagram aktivitas untuk proses mengembalikan *password*.

Gambar 3.5: Diagram aktivitas untuk mengembalikan *password*

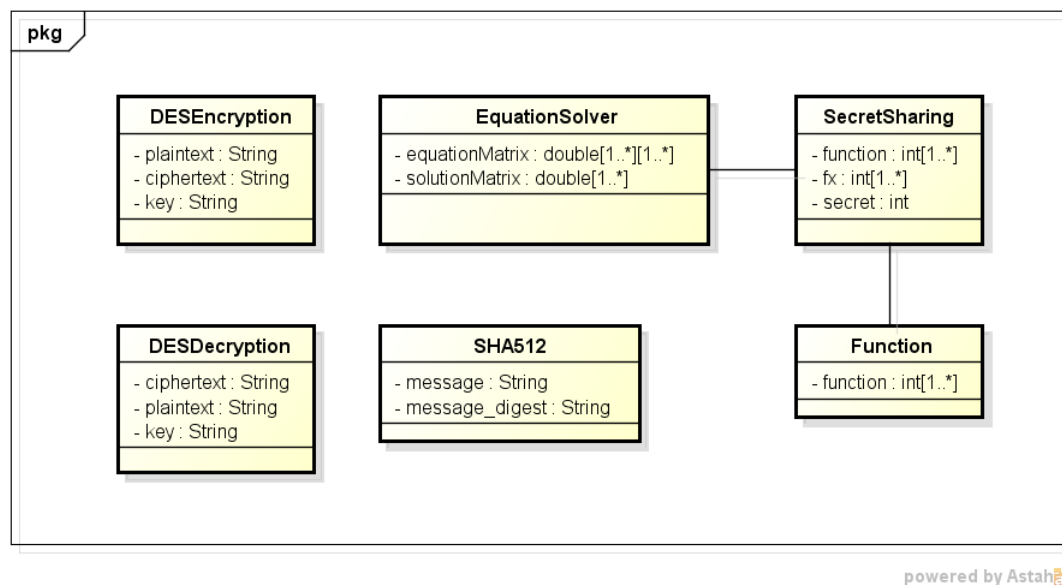
1 Dalam proses untuk mengembalikan *password*, *user* akan diminta untuk menjawab bebe-
 2 rapa pertanyaan keamanan yang sudah dipilih saat *user* menyimpan *password*. Selanjutnya
 3 adalah proses yang sama saat menyimpan *password*, yaitu menghitung nilai *hash* dari per-
 4 tanyaan keamanan yang sudah dijawab oleh *user*. Langkah selanjutnya adalah mendekripsi
 5 setiap *share* dengan menggunakan kunci nilai *hash*.

6 Langkah selanjutnya adalah dengan menggunakan skema *threshold* (k, n) membangun
 7 atau rekontruksi ulang *password*. Jika banyak pertanyaan yang dijawab benar oleh *user*
 8 sama dengan atau lebih dari k *share*, maka *user* bisa mendapatkan kembali *password*, dan
 9 jika kurang dari k *share* maka *user* tidak bisa mendapatkan kembali *password*.

10 3.3.4 Diagram Kelas

11 Perangkat lunak yang dibangun memiliki 2 bagian utama, yaitu bagian *engine* dan bagian
 12 antarmuka (*user interface*). Bagian *engine* berfungsi untuk menyimpan dan mengembalikan
 13 *password*, melakukan proses enkripsi dan dekripsi, dan melakukan *secret sharing*.

14 Bagian *engine* merupakan sekumpulan kelas *Java*, sedangkan bagian *antarmuka* akan
 15 terdiri dari sekumpulan *Java Server Page* atau JSP. Pada bagian ini akan dijelaskan bagian
 16 *engine* saja. Gambar 3.6 menunjukkan diagram kelas *engine*.

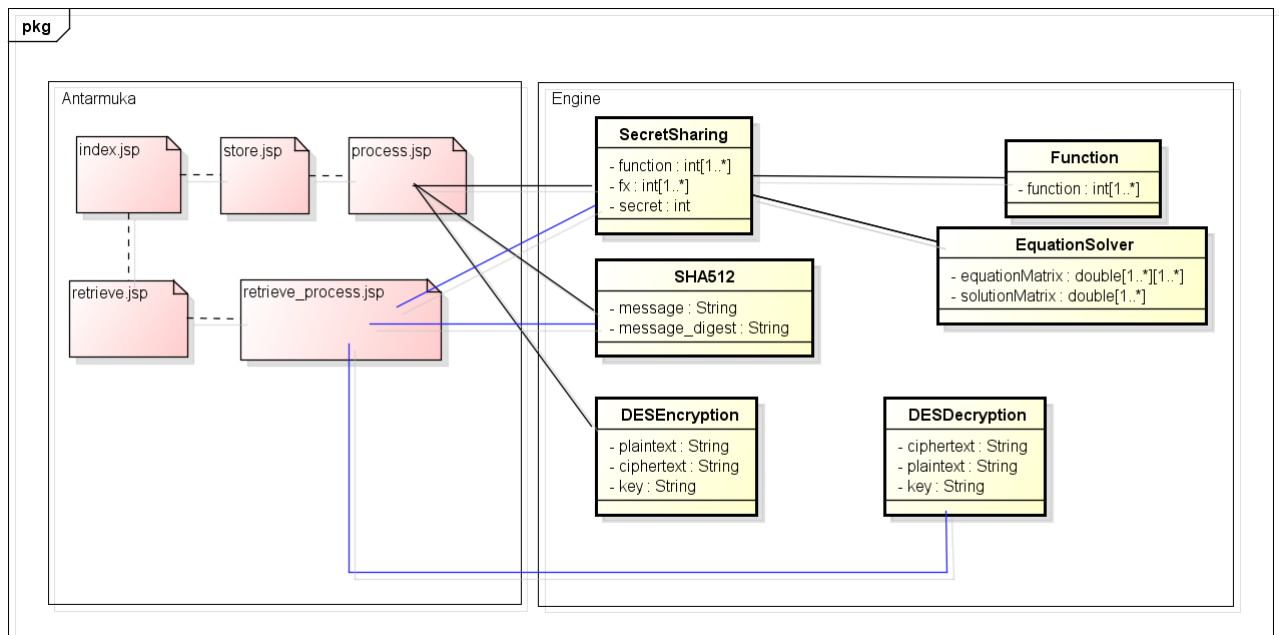
Gambar 3.6: Diagram kelas *engine*

1 Untuk proses penyimpanan *password*, kelas SHA512 berfungsi untuk menghitung ni-
 2 lai *hash* dari gabungan pertanyaan, jawaban, dan *salt*. Selanjutnya, kelas SecretSharing
 3 akan membagi *password* menjadi beberapa *share*. Kemudian, kelas DESEncryption akan
 4 mengenkripsi setiap *share* dengan nilai *hash* sebagai kunci rahasia. Setiap *ciphertext* hasil
 5 enkripsi, nilai *salt*, dan pertanyaan akan disimpan.

6 Untuk proses pengembalian *password*, kelas SHA512 akan menghitung nilai *hash* dari ga-
 7 bungan pertanyaan, jawaban, dan *salt*. Kemudian, kelas DESDecryption akan mendekripsi
 8 *ciphertext* hasil enkripsi yang disimpan dan kunci rahasia dari nilai *hash* untuk memperoleh
 9 *plaintext*. Kelas SecretSharing akan merekonstruksi *password* berdasarkan hasil dekripsi dari
 10 kelas DESDecryption. Jika, banyak pertanyaan benar sesuai, maka *password* bisa dikemba-
 11 likan.

12 3.3.5 Arsitektur Perangkat Lunak

13 Pada bagian sebelumnya sudah dijelaskan mengenai alur proses, diagram *use case*, diagram
 14 aktivitas, dan diagram kelas dari perangkat lunak yang dibangun. Pada bagian ini akan
 15 dijelaskan mengenai seluruh bagian perangkat lunak yang dibangun. Seperti yang sudah di-
 16 jelaskan sebelumnya perangkat lunak yang dibangun memiliki 2 bagian utama, yaitu bagian
 17 *engine* dan bagian antarmuka. Gambar 3.7 menunjukkan arsitektur dari perangkat lunak.



powered by Astah

Gambar 3.7: Arsitektur perangkat lunak

- 1 Untuk proses penyimpanan *password*, sama seperti pada bagian sebelumnya, kelas yang
- 2 akan digunakan adalah kelas SHA512 untuk menghasilkan *hash*, kemudian kelas SecretSha-
- 3 ring untuk menghasilkan *share* dari *password*, dan kelas DESEncryption untuk mengenkripsi
- 4 masing-masing dari *share* dengan nilai *hash* sebagai kunci rahasia.
- 5 Selanjutnya, untuk proses pengembalian *password*, kelas SHA512 akan digunakan kem-
- 6 bali untuk menghasilkan *digest*. Setelah *digest* dihasilkan, kelas DESDecryption akan men-
- 7 dekripsi *share-share* yang dimiliki dengan kunci *digest* yang dihasilkan. Selanjutnya, kelas
- 8 SecretSharing akan merekonstruksi ulang hasil dekripsi dari kelas DESDecryption dan me-
- 9 nentukan apakah *password* bisa dikembalikan atau tidak.

1
2

BAB 4

PERANCANGAN

3 Pada bab ini akan dibahas mengenai perancangan perangkat lunak. Perancangan perangkat
4 lunak akan mencakup diagram kelas rinci, perancangan berorientasi objek, dan perancangan
5 antarmuka.

6 4.1 Diagram Kelas Rinci

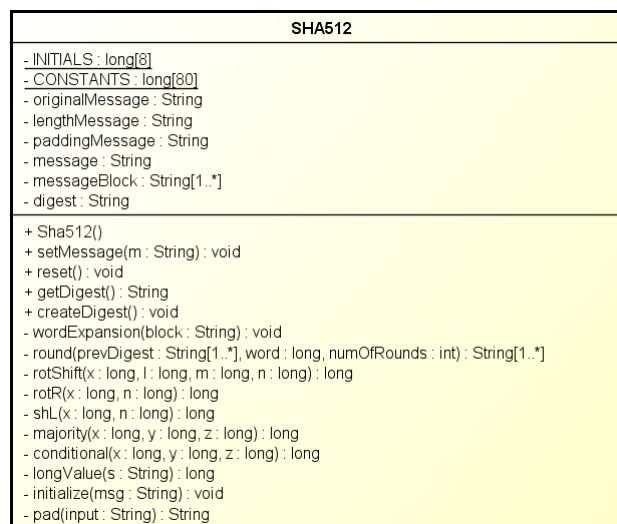
7 Diagram kelas rinci digunakan sebagai gambaran umum untuk setiap kelas yang ada dalam
8 perangkat lunak yang dibangun serta keterkaitan setiap kelas. Diagram kelas rinci dapat
9 dilihat pada Gambar 4.1. Ada perbedaan antara diagram kelas pada Gambar 4.1 dengan
10 kelas diagram pada Bab 3. Pada diagram kelas rinci ditambahkan beberapa atribut dan
11 fungsi sesuai dengan kebutuhan dari masing-masing kelas.

12 4.2 Deskripsi Kelas dan Fungsi

13 Pada bagian ini akan berisi mengenai penjelasan secara rinci masing-masing kelas. Tujuan-
14 nya adalah menjelaskan peran setiap kelas dalam perangkat lunak yang dibangun.

15 4.2.1 Kelas *SHA512*

16 Kelas *SHA512* merupakan kelas yang mengimplementasikan *Secure Hashing Algorithm 512*
17 (SHA-512). Cara kerja algoritma dapat dilihat pada bagian 2.5. Kelas SHA512 ditunjukkan
18 pada Gambar 4.2.



Gambar 4.2: Kelas SHA512

19 Adapun atribut dari kelas *SHA512*, yaitu *INITIALS*, *CONSTANTS*, *originalMessage*,
20 *lengthMessage*, *paddingMessage*, *message*, *messageBlock*, dan *digest*. Berikut penjelasan

1 masing-masing atribut tersebut:

2 1. *long[8] INITIALS*

3 Atribut yang berguna untuk menyimpan nilai dari konstanta awal.

4 2. *long[80] INITIALS*

5 Atribut yang berguna untuk menyimpan konstanta yang digunakan dalam setiap putaran SHA-512.

7 3. *String originalMessage*

8 Atribut yang berguna untuk menyimpan *message* yang belum *padding* dalam bentuk *string* biner.

10 4. *String lengthMessage*

11 Atribut yang berguna untuk menyimpan informasi mengenai panjang atribut *originalMessage* dalam bentuk *string* biner.

13 5. *String paddingMessage*

14 Atribut yang berguna untuk menyimpan blok *padding* dalam bentuk *string* biner.

15 6. *String message*

16 Atribut yang berguna untuk menyimpan *message* yang sudah *padding* dalam bentuk *string* biner.

18 7. *String digest*

19 Atribut yang berguna untuk menyimpan *digest* dari atribut *message* dalam bentuk *string* biner.

21 Adapun fungsi yang membangun kelas *SHA512*, yaitu *Sha512*, *setMessage*, *reset*, *getDigest*, *wordExpansion*, *round*, *rotShift*, *rotR*, *shL*, *majority*, *conditional*, *longValue*, *initialize*, dan *pad*. Berikut penjelasan masing-masing fungsi tersebut:

24 1. *Sha512*

25 Merupakan konstruktor dari kelas *SHA512*.

26 2. *void setMessage(String m)*

27 Menyimpan nilai *string m* ke dalam atribut *originalMessage*.

28 3. *void reset*

29 Mengembalikan nilai atribut *originalMessage*, *lengthMessage*, *paddingMessage*, *message*, dan *digest* menjadi *string* kosong dan mengembalikan nilai atribut *messageBlock* menjadi *array string* kosong.

32 4. *String getDigest*

33 Mengembalikan *string digest* dalam bentuk heksadesimal.

34 5. *void createDigest*

35 Membuat *digest* dari *message*. Algoritma untuk membuat digest ini dapat dilihat pada bagian 2.5.

37 6. *void wordExpansion(String block)*

38 4.2.2 Kelas *Function*

39 Kelas *Function* merupakan kelas yang merepresentasikan sebuah fungsi polinomial $f(x)$.

40 Kelas *Function* ditunjukkan pada Gambar 4.3.

Function
- function : int[1..*]
+ Function(func : int[1..*])
+ countFunction(x : int) : int

Gambar 4.3: Kelas *Function*

Adapun atribut dari kelas *Function* adalah *function*. Atribut *function* berguna untuk menyimpan nilai setiap koefisien dari fungsi polinomial $f(x)$ dalam tipe data *array* bilangan bulat.

Sementara itu, fungsi yang dimiliki oleh kelas *Function*, yaitu *Function* dan *countFunction*. Berikut penjelasan masing-masing fungsi:

1. *Function(int[] func)*

Merupakan konstruktor dari kelas *Function* yang menerima masukan *array* bilangan bulat.

2. *int countFunction(int x)*

Menghitung nilai x untuk fungsi polinomial $f(x)$. Algoritma dari fungsi ini ditunjukkan pada Algoritma 1.

Algorithm 1 countFunction

```

1: function COUNTFUNCTION(x)
2:   for i < panjang array dari atribut function do
3:     res = res + (function[i] · xi)
4:   end for
5:   return res
6: end function

```

11

4.2.3 Kelas *EquationSolver*

Kelas *EquationSolver* adalah kelas yang mengimplementasikan eliminasi Gauss-Jordan (Bagian 2.7). Kelas ini berperan untuk menyelesaikan persamaan linear. Gambar 4.4 menunjukkan kelas *EquationSolver*.

EquationSolver
- equationMatrix : double[1..*][1..*] - solutionMatrix : double[1..*]
+ EquationSolver(equation : double[1..*][1..*], solution : double[1..*]) + reset() : void + solve() : double[1..*]

Gambar 4.4: Kelas *EquationSolver*

Adapun atribut dari kelas *EquationSolver* yaitu, *equationMatrix* dan *solutionMatrix*. Berikut penjelasan masing-masing atribut:

1. *double[][] equationMatrix*

Atribut yang menyimpan bentuk matriks dari persamaan linear.

2. *double[] solutionMatrix*

Atribut yang menyimpan bentuk matriks dari solusi masing-masing persamaan linear.

Sementara itu, fungsi yang dimiliki oleh kelas *EquationSolver* yaitu, *EquationSolver*, *reset*, dan *solve*. Berikut penjelasan masing-masing fungsi:

1. *EquationSolver(double[][] equation, double[] solution)*
Merupakan konstruktor dari kelas *EquationSolver* yang menerima masukan berupa matriks persamaan dan matriks solusi.
2. *void reset*
Mengembalikan nilai atribut *equationMatrix* dan *solutionMatrix* menjadi array kosong.
3. *double[] solve*
Mencari solusi dari persamaan linear dan mengembalikan solusi dalam bentuk *array*.

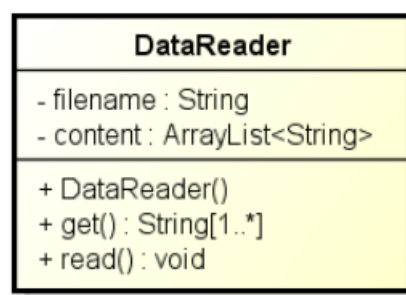
4.2.4 Kelas *SecretSharing*

4.2.5 Kelas *DESEncryption*

4.2.6 Kelas *DESDecryption*

4.2.7 Kelas *DataReader*

Kelas *DataReader* merupakan kelas yang berperan untuk membaca berkas teks. Kelas *DataReader* ditunjukkan pada Gambar 4.5.



Gambar 4.5: Kelas *DataReader*

Kelas ini memiliki 2 atribut, yaitu *filename* dan *content*. Berikut penjelasan masing-masing atribut:

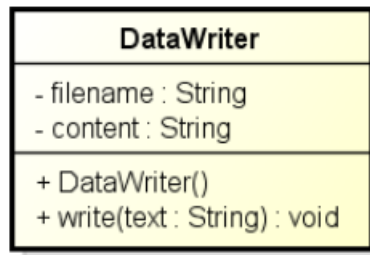
1. *String filename*
Atribut yang menyimpan nama dari berkas teks yang dibaca.
2. *ArrayList<String> content*
Atribut yang menyimpan isi dari berkas teks yang dibaca.

Adapun kelas ini memiliki 3 fungsi, yaitu *DataReader*, *get*, dan *read*. Berikut penjelasan masing-masing fungsi:

1. *DataReader*
Merupakan konstruktor dari kelas *DataReader*.
2. *String[] get*
Fungsi yang berguna untuk mengembalikan atribut *content*.
3. *void read*
Fungsi yang berperan membaca berkas teks.

1 4.2.8 Kelas *DataWriter*

- 2 Kelas *DataWriter* adalah kelas yang berperan untuk menulis keluaran ke dalam berkas teks.
 3 Kelas *DataWriter* ditunjukkan pada Gambar 4.6.



Gambar 4.6: Kelas *DataWriter*

- 4 Kelas ini memiliki 2 atribut, yaitu *filename* dan *content*. Berikut penjelasan masing-
 5 masing atribut:

6 1. *String filename*

- 7 Atribut yang menyimpan nama dari berkas teks yang akan ditulis.

8 2. *String content*

- 9 Atribut yang menyimpan isi dari berkas teks yang akan ditulis.

- 10 Adapun kelas ini memiliki 2 fungsi, yaitu *DataWriter* dan *write*. Berikut penjelasan
 11 masing-masing fungsi:

12 1. *DataWriter*

- 13 Merupakan konstruktor dari kelas *DataWriter*.

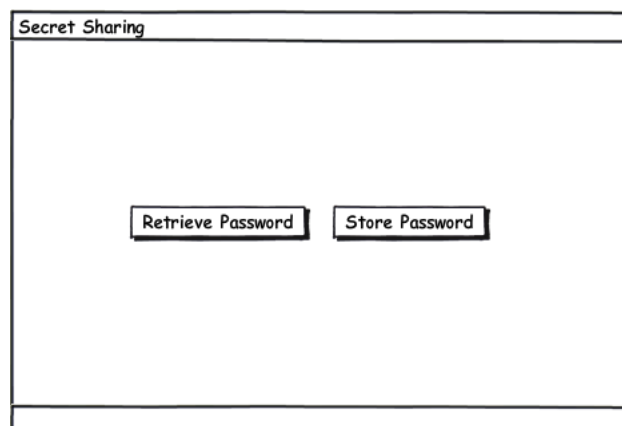
14 2. *write(String text)*

- 15 Fungsi yang berperan untuk menulis isi dari berkas teks.

16 4.3 Perancangan Antarmuka

- 17 Perangkat lunak yang dikembangkan akan memiliki 3 tampilan utama, tampilan untuk me-
 18 nyimpan *password*, tampilan untuk mengembalikan *password*, dan tampilan untuk memilih
 19 menyimpan *password* atau mengembalikan *password*.

- 20 Gambar 4.7 menunjukkan tampilan awal yang akan dimunculkan pertama kali untuk
 21 memilih menyimpan *password* atau mengembalikan *password*.



Gambar 4.7: Perancangan Tampilan Awal

- 1 Tampilan utama ini cukup sederhana. Dalam tampilan utama pada Gambar 4.7, hanya
 2 terdapat 2 pilihan, yaitu *store password* untuk menyimpan *password* dan *retrieve password*
 3 untuk mengembalikan *password*. Selanjutnya, jika pengguna memilih *store password*, maka
 4 akan ditampilkan halaman *store password*.

The screenshot shows a window titled "Secret Sharing". At the top is a button labeled "Add Password". Below it is a text input field labeled "Password:". A horizontal line separates this from the "Security Questions" section. In this section, there is a text input field labeled "Question" followed by an "Add" button. Below this, there is a list of questions, with the first one being "1. Question#1" followed by an "Answer" input field. At the bottom of the window are two buttons: "Submit" and "Cancel".

Gambar 4.8: Perancangan Tampilan Menyimpan *Password*

- 5 Pada tampilan menyimpan *password* di Gambar 4.8, tombol "*Add Password*" berfungsi
 6 untuk menambah *text box password*, pada bagian ini pengguna bisa mengisi *password* yang
 7 akan disimpan. Bagian "*Security Questions*" berisi pertanyaan keamanan yang dibuat oleh
 8 pengguna. Setelah pengguna mengisi pertanyaan personal pada *text box* di bagian "*Secu-*
 9 *rity Questions*" dan menekan tombol "*Add*", akan muncul pertanyaan yang sudah dibuat,
 10 kemudian pengguna harus mengisi jawaban dari pertanyaan keamanan yang sudah dibuat.

- 11 Setelah mengisi seluruh pertanyaan keamanan, pengguna bisa menyimpan *password* de-
 12 ngan menekan tombol "*Submit*". Tombol "*Cancel*" berfungsi untuk kembali ke tampilan
 13 awal. Setelah tombol "*Submit*" ditekan, maka *password* sudah disimpan dan akan kembali
 14 ditampilkan tampilan awal.

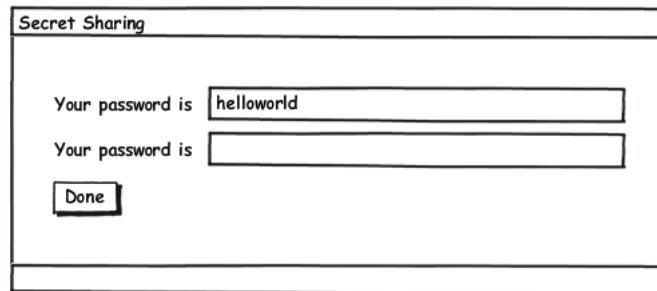
- 15 Berikutnya adalah tampilan untuk mengembalikan *password*. Gambar 4.9 menunjukkan
 16 tampilan untuk mengembalikan *password*.

The screenshot shows a window titled "Secret Sharing". It contains a section labeled "Security Questions" with a list of five questions: "1. Question#1", "2. Question#2", "3. Question#3", "4. Question#4", and "5. Question#5". To the right of each question is an empty rectangular input box. At the bottom of the window are two buttons: "Submit" and "Cancel".

Gambar 4.9: Perancangan Tampilan Mengembalikan *Password*

- 17 Pada bagian untuk mengembalikan *password*, tampilannya cukup sederhana dan penggu-
 18 na hanya cukup memasukkan setiap jawaban dari pertanyaan keamanan yang sudah dibuat
 19 sebelumnya di bagian penyimpanan *password*. Pada bagian ini, pengguna bebas untuk
 20 memilih mengisi setiap pertanyaan atau tidak menjawab pertanyaan keamanan. Setelah se-

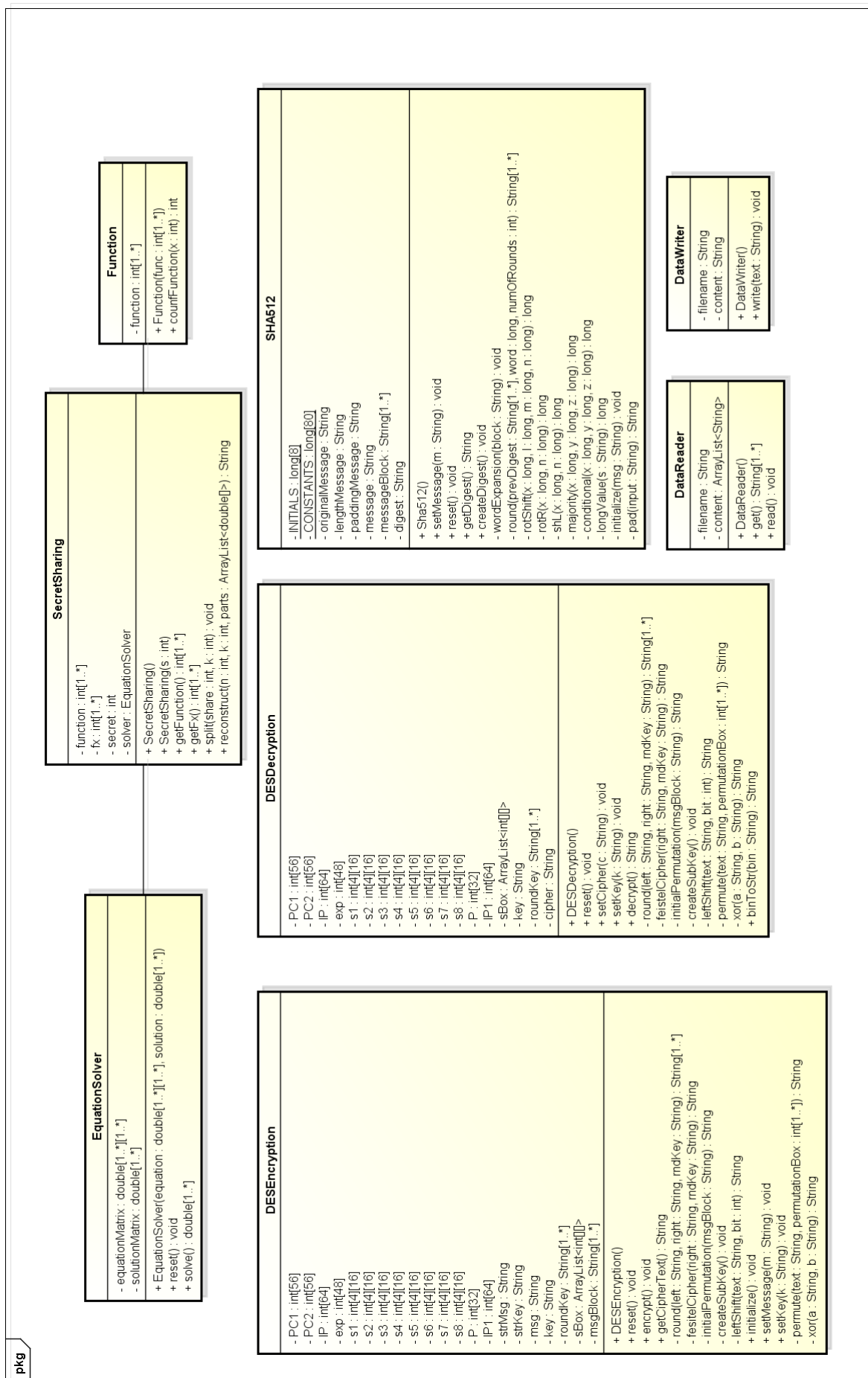
- 1 luruh pertanyaan sudah dijawab, pengguna dapat menekan tombol "*Submit*" yang kemudian
- 2 akan menunjukkan *password* pengguna.
- 3 Gambar 4.10 menunjukkan tampilan sesudah pengguna menekan tombol "*Submit*" pada
- 4 bagian di Gambar 4.9.



The image shows a window titled "Secret Sharing". Inside the window, there are two lines of text, each followed by a text input field. The first line says "Your password is" and the input field contains the text "helloworld". The second line also says "Your password is" and the input field is empty. Below these two lines is a button labeled "Done".

Gambar 4.10: Perancangan Tampilan Mengembalikan *Password*

- 5 Jika banyak pertanyaan keamanan yang dijawab benar oleh pengguna sesuai dengan
- 6 minimal banyak pertanyaan keamanan yang dijawab benar maka pengguna bisa melihat
- 7 *password* yang sudah disimpan. Tapi, jika banyak pertanyaan keamanan yang dijawab benar
- 8 oleh pengguna kurang dari minimal banyak pertanyaan keamanan yang harus dijawab benar
- 9 maka pengguna tidak bisa melihat *password* yang sudah disimpan.



Gambar 4.1: Diagram Kelas Rinci

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan berisi mengenai implementasi perangkat lunak dan pengujian perangkat lunak yang dibangun.

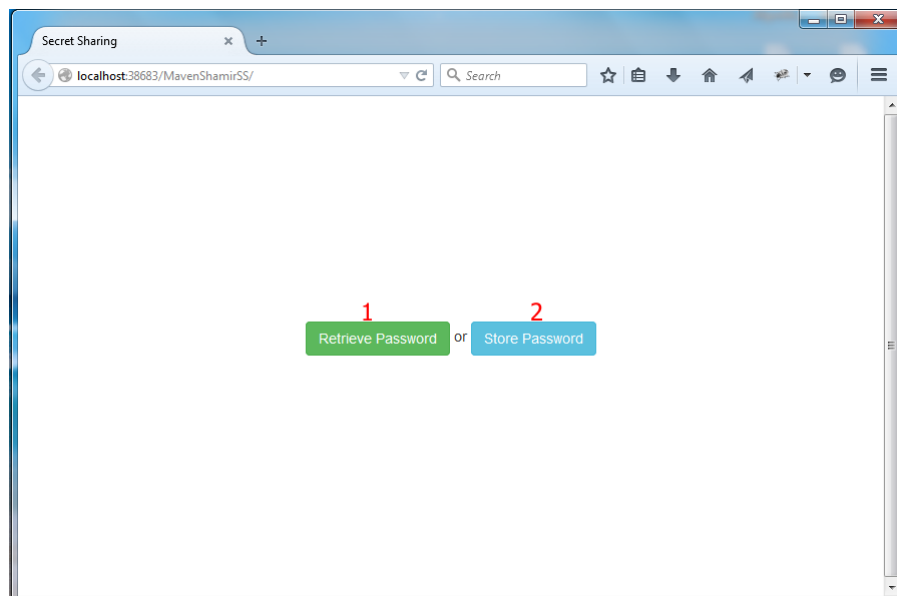
5.1 Implementasi Perangkat Lunak

Pada bagian ini akan dibahas mengenai tampilan antarmuka perangkat lunak yang sudah dibangun.

5.1.1 Tampilan Antarmuka Perangkat Lunak

Tampilan antarmuka awal perangkat lunak dapat dilihat pada Gambar 5.1 dengan keterangan bagian-bagian sebagai berikut.

- Bagian no 1 merupakan tombol untuk mengembalikan *password*.
- Bagian no 2 merupakan tombol untuk menyimpan *password*.

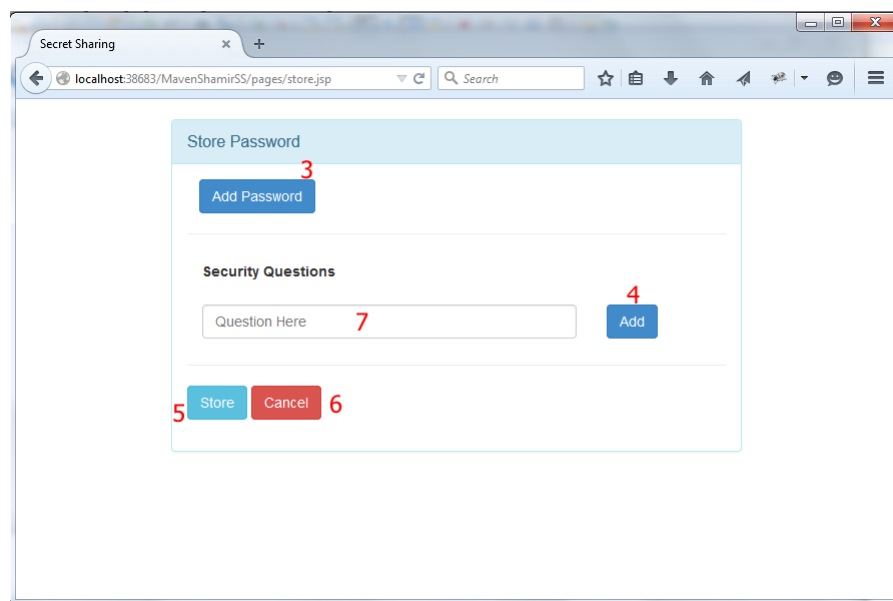


Gambar 5.1: Tampilan antarmuka awal

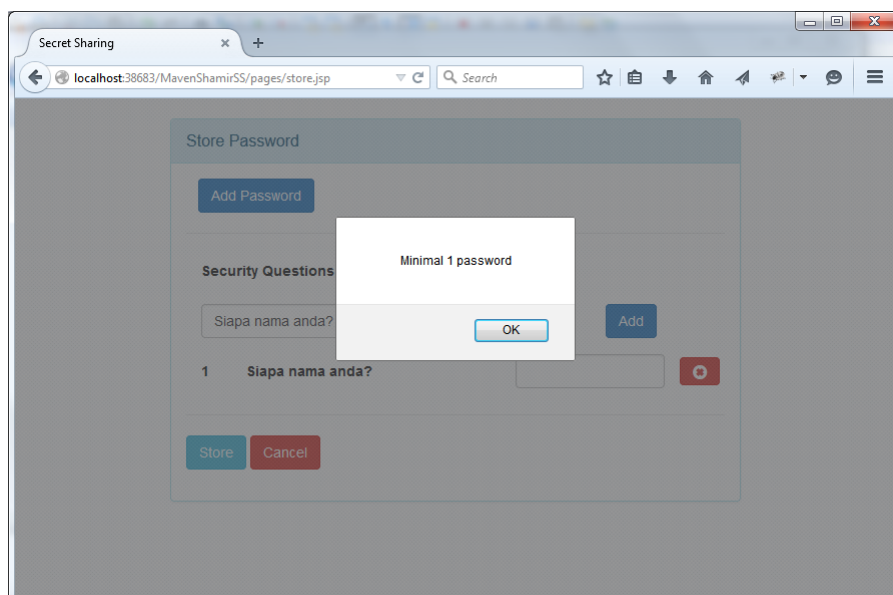
Setelah tombol "*Store Password*" ditekan, tampilan antarmuka perangkat lunak akan terlihat seperti pada Gambar 5.2 dengan keterangan sebagai berikut.

- Bagian no 3 merupakan tombol untuk menambah *password*. Pengguna minimal harus menambahkan 1 *password*, jika tidak maka akan muncul notifikasi seperti pada Gambar 5.3.

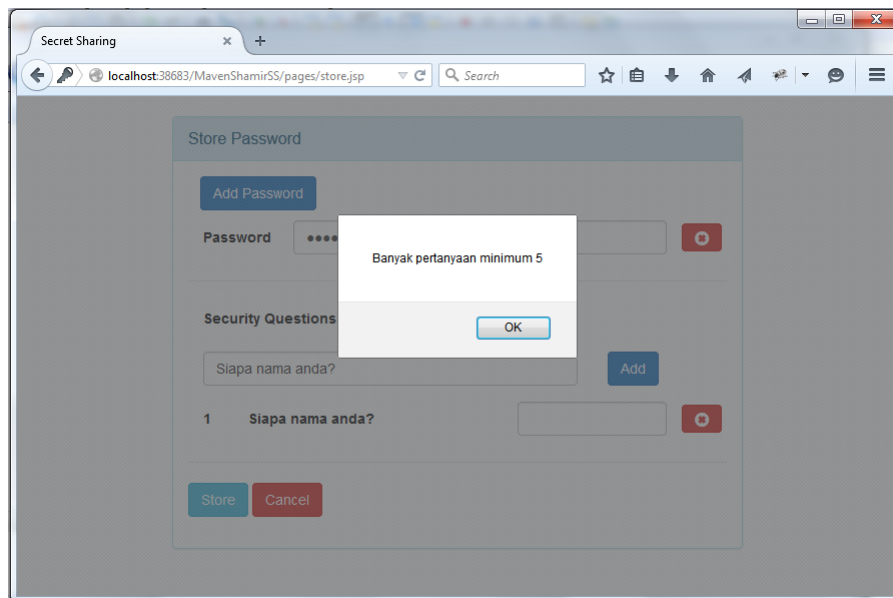
- 1 • Bagian no 4 merupakan tombol untuk menambah pertanyaan keamanan. Pengguna
2 minimal harus menambahkan 5 pertanyaan keamanan, jika kurang maka akan muncul
3 notifikasi seperti pada Gambar 5.4.
- 4 • Bagian no 5 merupakan tombol untuk melanjutkan menyimpan *password*.
- 5 • Bagian no 6 merupakan tombol untuk kembali ke tampilan antarmuka awal.
- 6 • Bagian no 7 merupakan teks masukkan untuk pertanyaan keamanan yang hendak
7 ditambahkan.



Gambar 5.2: Tampilan antarmuka untuk menyimpan *password*

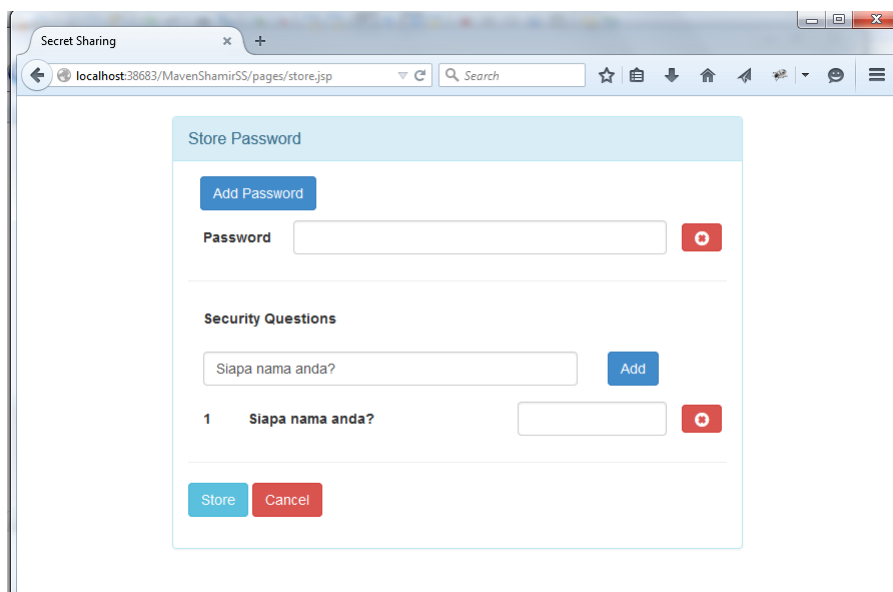


Gambar 5.3: Tampilan antarmuka untuk menyimpan *password*



Gambar 5.4: Tampilan antarmuka untuk menyimpan *password*

- 1 Setelah tombol "*Add Password*" ditekan, maka tampilan antarmuka akan menambahk-
2 an masukkan teks untuk memasukkan *password* yang hendak disimpan. Setelah tombol
3 "*Add*" ditekan, maka tampilan antarmuka akan menambahkan teks masukkan untuk jawab-
4 an dari pertanyaan keamanan yang sudah diisi di Bagian no 7. Tampilan yang ditunjukkan
5 perangkat lunak dapat dilihat pada Gambar 5.5.

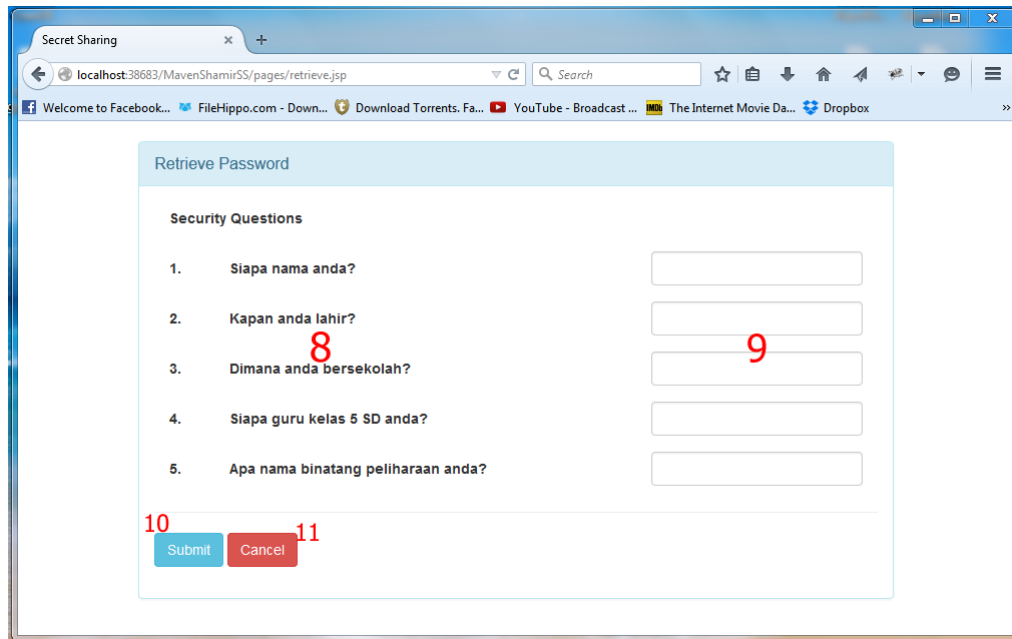


Gambar 5.5: Tampilan antarmuka untuk menyimpan *password*

- 6 Setelah tombol "*Store*" ditekan, maka tampilan antarmuka perangkat lunak akan kembali
7 ke tampilan antarmuka awal. *Password* sudah berhasil disimpan. Kemudian, setelah tombol
8 "*Retrieve Password*" ditekan, maka tampilan perangkat lunak akan terlihat seperti pada
9 Gambar 5.6 dengan keterangan sebagai berikut.

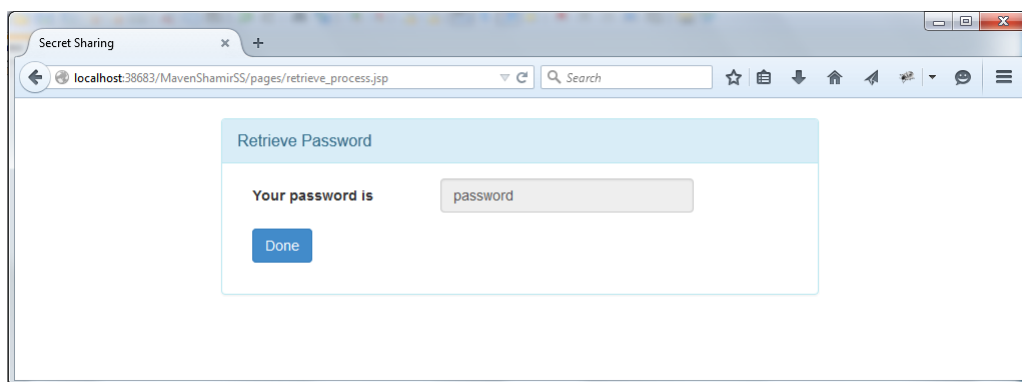
- 10 • Bagian no 8 merupakan bagian dari pertanyaan keamanan yang harus dijawab oleh
11 pengguna.
12 • Bagian no 9 merupakan bagian dari jawaban setiap pertanyaan keamanan yang harus
13 dijawab.

- 1 • Bagian no 10 merupakan tombol untuk mengembalikan *password*.
- 2 • Bagian no 11 merupakan tombol untuk kembali ke tampilan antarmuka awal.



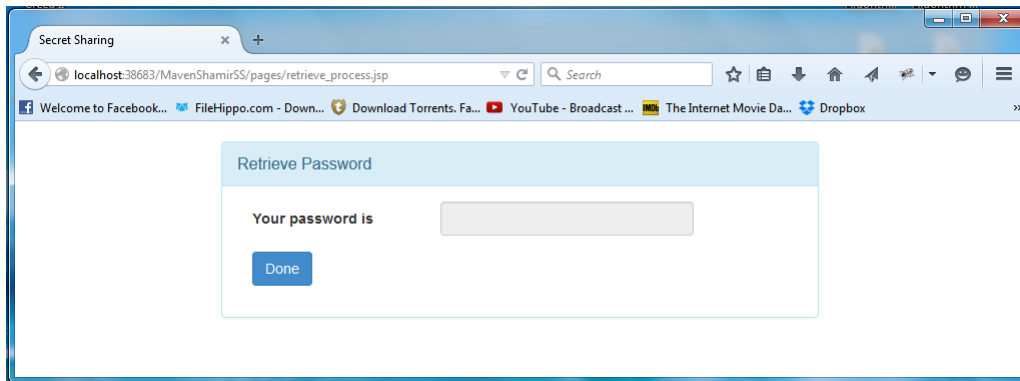
Gambar 5.6: Tampilan antarmuka untuk mengembalikan *password*

- 3 Setelah tombol "Submit" pada Gambar 5.6 ditekan, perangkat lunak akan memroses
- 4 setiap pertanyaan dan jawaban. Jika banyak jawaban benar dari pertanyaan keamanan
- 5 yang dijawab oleh pengguna sesuai dengan minimal banyak pertanyaan keamanan yang
- 6 dijawab benar yang sudah ditentukan sebelumnya, maka tampilan perangkat lunak akan
- 7 terlihat seperti pada Gambar 5.7.



Gambar 5.7: Tampilan antarmuka untuk mengembalikan *password*

- 8 Sedangkan, jika tidak sesuai, maka tampilan perangkat lunak akan terlihat seperti pada
- 9 Gambar 5.8.



Gambar 5.8: Tampilan antarmuka untuk mengembalikan *password*

5.2 Pengujian Perangkat Lunak

Pada bagian ini akan berisi tentang metode pengujian, hasil pengujian, analisis pengujian, dan kesimpulan dari pengujian perangkat lunak yang sudah dibangun.

5.2.1 Metode Pengujian

Pengujian terhadap perangkat lunak yang sudah dibangun akan dibagi menjadi 2 bagian, yaitu

- Pengujian fungsional
- Pengujian survei

Pengujian fungsional bertujuan untuk menguji apakah perangkat lunak yang dibangun sudah bisa mengimplementasikan *secret sharing* shamir. Pengujian survei bertujuan untuk menguji bagaimana kualitas pertanyaan keamanan yang dibuat.

5.2.2 Hasil Pengujian Fungsional

Pengujian fungsional adalah pengujian yang dilakukan terhadap perangkat lunak yang sudah dibangun dengan tujuan untuk memastikan bahwa perangkat lunak yang dibangun sudah bisa mengimplementasikan *secret sharing* shamir. Pada bagian ini akan diberikan satu contoh kasus dimana *password* dapat dikembalikan dan satu contoh kasus dimana *password* tidak bisa dikembalikan.

Pada bagian ini juga akan dijelaskan langkah-langkah dimulai dari menyimpan *password* sampai dengan mengembalikan *password*. Dalam contoh kasus untuk pengujian fungsional, banyak *share* 5, $n = 5$ dan minimal *share* yang dimiliki supaya bisa mengembalikan *password* sebanyak 4, $k = 4$. Langkah awal adalah menyimpan *password*. Gambar 5.9 menunjukkan tampilan antarmuka perangkat lunak untuk langkah awal.

Store Password

Add Password

Password :

Security Questions

Question Here Add

1	Siapa nama anda?	<input type="text" value="Samuel"/>	<input type="button" value="✕"/>
2	Dimana anda lahir?	<input type="text" value="Bandung"/>	<input type="button" value="✕"/>
3	Apakah anda memiliki binatang peliharaan?	<input type="text" value="Tidak"/>	<input type="button" value="✕"/>
4	Kapan anda lahir?	<input type="text" value="31 Juli 1993"/>	<input type="button" value="✕"/>
5	Dimana anda berkuliah?	<input type="text" value="UNPAR"/>	<input type="button" value="✕"/>

Store Cancel

Gambar 5.9: Langkah menyimpan *password*

1 Pada bagian ini, jenis pertanyaan yang dibuat tidak dipermasalahkan karena tujuannya
 2 hanya untuk fungsionalitas. Selain itu, pada bagian masukkan teks untuk *password*, *pass-*
 3 *word* ditunjukkan sekedar bagian dari pengujian. Langkah selanjutnya setelah menyimpan
 4 *password* adalah mengembalikan *password*. Gambar 5.10 menunjukkan tampilan antarmuka
 5 perangkat lunak untuk mengembalikan *password*.

Retrieve Password

Security Questions

1.	Siapa nama anda?	<input type="text" value="Samuel"/>
2.	Dimana anda lahir?	<input type="text" value="Bandung"/>
3.	Apakah anda memiliki binatang peliharaan?	<input type="text" value="Tidak"/>
4.	Kapan anda lahir?	<input type="text" value="31 Juli 1993"/>
5.	Dimana anda berkuliah?	<input type="text" value="UNPAR"/>

Submit Cancel

Gambar 5.10: Langkah menjawab pertanyaan keamanan

6 Gambar 5.10 menunjukkan bahwa seluruh pertanyaan keamanan dijawab dengan be-
 7 nar dan *password* akan berhasil dikembalikan. Gambar 5.11 menunjukkan bahwa *password*
 8 berhasil dikembalikan.

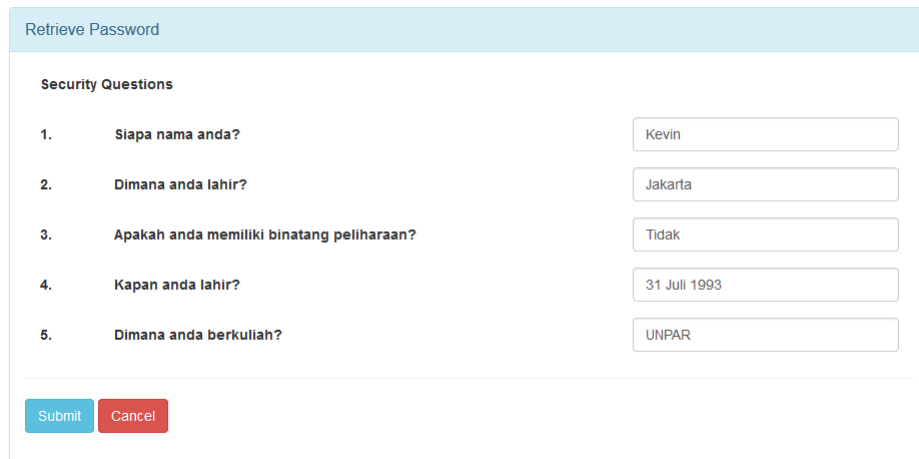
Retrieve Password

Your password is

Done

Gambar 5.11: *Password* berhasil dikembalikan

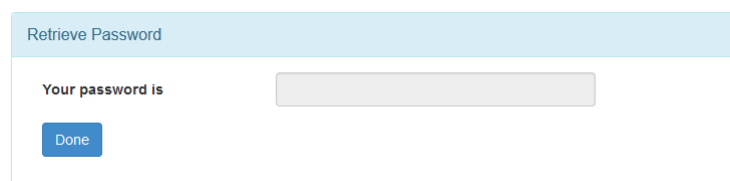
- 1 Contoh kasus diatas adalah contoh kasus *password* berhasil didapatkan, berikutnya akan
 2 ditunjukkan contoh kasus dimana *password* tidak berhasil dikembalikan. Langkah awal
 3 dimulai langsung dari langkah mengembalikan *password* dan ditunjukkan pada Gambar 5.12.



The screenshot shows a 'Retrieve Password' form with a light blue header. Below the header is a section titled 'Security Questions'. It contains five numbered questions, each with a text input field to its right. The answers are: 1. 'Siapa nama anda?' with 'Kevin', 2. 'Dimana anda lahir?' with 'Jakarta', 3. 'Apakah anda memiliki binatang peliharaan?' with 'Tidak', 4. 'Kapan anda lahir?' with '31 Juli 1993', and 5. 'Dimana anda berkuliah?' with 'UNPAR'. At the bottom of the form are two buttons: 'Submit' (blue) and 'Cancel' (red).

Gambar 5.12: Langkah menjawab pertanyaan keamanan

- 4 Gambar 5.12 menunjukkan bahwa dari $k = 4$, banyak pertanyaan keamanan yang bisa
 5 dijawab benar hanya 3 pertanyaan saja atau $k - 1$, yaitu pertanyaan nomor 3, 4, dan 5.
 6 Gambar 5.13 menunjukkan bahwa *password* tidak berhasil didapatkan.



The screenshot shows the 'Retrieve Password' form with a light blue header. Below the header, the text 'Your password is' is displayed next to a greyed-out password input field. Below this is a single blue button labeled 'Done'.

Gambar 5.13: *Password* tidak berhasil dikembalikan

7 5.2.3 Hasil Pengujian Survei

- 8 Pada bagian ini akan ditunjukkan hasil pengujian survei. Hasil pengujian survei bertujuan
 9 untuk menilai kualitas dari pertanyaan keamanan dengan melihat tingkat kesulitan untuk
 10 menebak atau menjawab jawaban benar. Asumsi yang digunakan dalam pengujian ini adalah
 11 seluruh jawaban relevan dengan pertanyaan keamanan.

- 12 Pengujian survei ini dilakukan terhadap 7 orang responden terbagi atas 4 kasus. Res-
 13 ponden melakukan survei dengan cara mencoba untuk menebak jawaban dari pertanyaan
 14 keamanan untuk mengembalikan password. Setiap orang bebas memilih cara untuk men-
 15 dapatkan jawaban dari pertanyaan selain tidak bertanya kepada pembuat pertanyaan kea-
 16 manan.

- 17 Tujuh orang responden yang melakukan survei terdiri dari 3 responden teman, 2 res-
 18 ponden teman dekat, dan 2 responden keluarga dari pembuat pertanyaan keamanan. Kasus
 19 yang digunakan untuk pengujian survei terbagi menjadi 4 topik, yaitu

- 20 • Topik kasus 1: pertanyaan yang kemungkinan jawabannya hanya 2, yaitu Ya dan
 21 Tidak dan pertanyaan yang bisa ditemukan dalam sosial media.
- 22 • Topik kasus 2: pertanyaan yang jawabannya berupa angka (tanggal lahir, bulan, tahun,
 23 dan sebagainya) dan pertanyaan yang tidak stabil, yaitu ketika mengisi dan nanti
 24 menjawab belum tentu sama.
- 25 • Topik kasus 3: pertanyaan keamanan yang sifatnya personal (ada kemungkinan bisa
 26 dijawab bila ada hubungan keluarga).

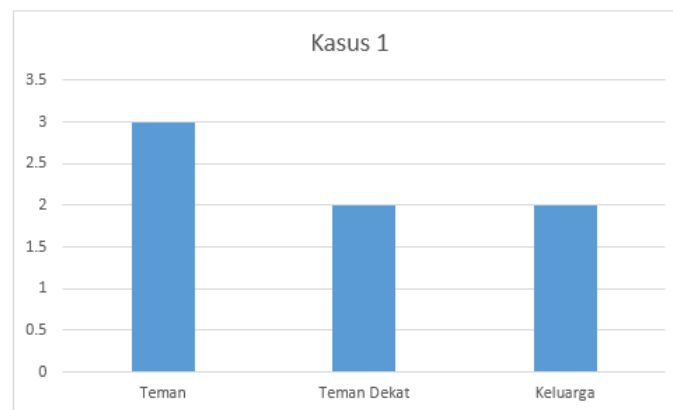
- 1 • Topik kasus 4: gabungan dari topik 1, 2, dan 3 secara merata.
- 2 Setiap kasus memiliki minimal banyak pertanyaan keamanan yang dijawab benar yang
- 3 sama, $n = 10, k = 4$. Berikut tabel hasil survei untuk setiap kasus beserta dengan penjela-
- 4 sannya.

5 Kasus 1

6 Berikut daftar pertanyaan keamanan yang digunakan dalam kasus 1.

- 7 • Apa jenis kelamin anda? (Laki-laki/Perempuan)
- 8 • Apakah anda pernah ke luar negeri?
- 9 • Apakah anda mempunyai binatang peliharaan?
- 10 • Apakah anda bermain alat musik?
- 11 • Apakah anda pernah tidak naik kelas?
- 12 • Apakah anda pernah mengalami kecelakaan?
- 13 • Apakah anda menyukai kegiatan olahraga?
- 14 • Apa nama belakang anda?
- 15 • Siapa nama ibu anda?
- 16 • Siapa nama ayah anda?

17 Kemudian, Grafik 5.14 menunjukkan hasil survei kasus 1.



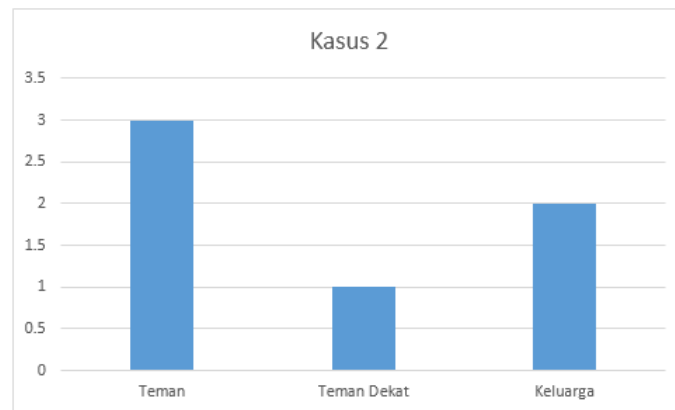
Gambar 5.14: Pengujian survei kasus 1

18 Kasus 2

19 Berikut daftar pertanyaan keamanan yang digunakan dalam kasus 2.

- 20 • Pada tahun berapa anda lahir?
- 21 • Pada tanggal berapa anda lahir?
- 22 • Pada bulan apa anda lahir?
- 23 • Berapa perbedaan umur anda dengan ayah anda?
- 24 • Berapa perbedaan umur anda dengan ibu anda?
- 25 • Berapa orang saudara anda?

- 1 • Berapa nomor rumah tempat anda tinggal?
- 2 • Dimana anda tinggal?
- 3 • Apa merek kendaraan yang anda pakai?
- 4 • Pada hari apa anda lahir?
- 5 Kemudian, Grafik 5.15 menunjukkan hasil survei kasus 2.



Gambar 5.15: Pengujian survei kasus 1

6 Kasus 3

7 Berikut daftar pertanyaan keamanan yang digunakan dalam kasus 3.

- 8 • Pada jam berapa anda lahir?(jj:mm)
- 9 • Apa nama sekolah dasar tempat anda bersekolah?
- 10 • Siapa nama belakang sepupu paling tua dari keluarga sisi ibu anda?
- 11 • Siapa nama belakang sepupu paling tua dari keluarga sisi ayah anda?
- 12 • Apa cita-cita anda dulu sewaktu kecil?
- 13 • Siapa nama anak paling tua dari nenek sisi ibu anda?
- 14 • Apa binatang peliharaan pertama anda?
- 15 • Apa alat musik yang anda mainkan pertama kali?
- 16 • Dimana kerabat terdekat anda tinggal/berasal?
- 17 • Siapa nama guru kelas 3 SD anda?

18 Tidak ada responden yang berhasil mengembalikan password untuk kasus 3, karena dari
19 itu grafik untuk kasus 3 tidak ditunjukkan.

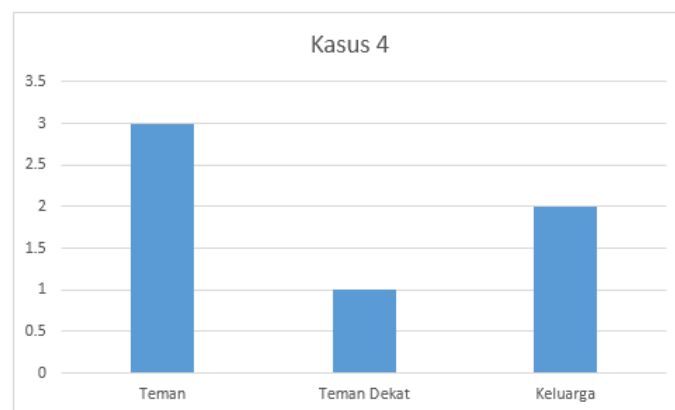
20 Kasus 4

21 Untuk kasus 4, karena merupakan gabungan dari kasus 1, 2, dan 3 maka banyak pertanyaan
22 pun ditambah menjadi 15 pertanyaan, masing-masing 5 pertanyaan untuk setiap kasus.

23 Berikut daftar pertanyaan keamanan yang digunakan dalam kasus 4.

- 24 • Apakah anda mempunyai binatang peliharaan?
- 25 • Apakah anda bermain alat musik?

- 1 • Apakah anda pernah tidak naik kelas?
- 2 • Apa nama belakang anda?
- 3 • Siapa nama ibu anda?
- 4 • Pada hari apa anda lahir?
- 5 • Pada tanggal berapa anda lahir?
- 6 • Pada bulan apa anda lahir?
- 7 • Berapa perbedaan umur anda dengan ayah anda?
- 8 • Berapa nomor rumah tempat anda tinggal?
- 9 • Pada jam berapa anda lahir?(jj:mm)
- 10 • Apa cita-cita anda dulu sewaktu kecil?
- 11 • Siapa nama anak paling tua dari nenek sisi ibu anda?
- 12 • Apa binatang peliharaan pertama anda?
- 13 • Siapa nama guru kelas 3 SD anda?
- 14 Kemudian, Grafik 5.16 menunjukkan hasil survei kasus 4.



Gambar 5.16: Pengujian survei kasus 1

5.2.4 Analisis Pengujian

Dari data-data hasil pengujian pada bagian subbab 5.2.2 dan subbab 5.2.3 akan dilakukan analisis. Untuk data hasil pengujian pada subbab 5.2.2 akan dilakukan analisis mengenai tingkat keberhasilan implementasi *secret sharing* shamir pada perangkat lunak yang dibangun. Untuk data hasil pengujian pada subbab 5.2.3 akan dilakukan analisis kualitas pertanyaan keamanan yang dibangun serta hubungannya dengan tingkat keberhasilan mengembalikan *password*.

Dinilai dari tingkat keberhasilan implementasi *secret sharing* shamir pada perangkat lunak bisa disebut berhasil. Dalam kasus untuk mengembalikan *password* pada subbab 5.2.2, dengan menjawab benar 4 pertanyaan atau lebih, *password* bisa dikembalikan, tetapi jika pertanyaan yang dijawab benar kurang 1 atau lebih, dalam kasus ini, kurang dari 4 pertanyaan yang dijawab benar, maka *password* tidak bisa dikembalikan. Hal ini berarti implementasi *secret sharing* shamir pada perangkat lunak bisa dinilai berhasil.

Selanjutnya adalah analisis kualitas pertanyaan keamanan yang dibangun serta efek dan hubungannya dengan tingkat keberhasilan mengembalikan *password*. Dilihat dari grafik

1 5.14 untuk topik kasus 1, karena mayoritas kemungkinan jawaban dari pertanyaan keaman-
2 an adalah kemungkinan biner dengan hanya 2 kemungkinan saja (Ya atau Tidak), maka
3 responden bisa dengan mudah mengembalikan *password*.

4 Dilihat dari grafik 5.15 untuk topik kasus 2, tingkat keberhasilan untuk mengembalikan
5 *password* cukup tinggi karena hampir seluruh responden berhasil untuk mengembalikan
6 *password*. Hal ini disebabkan karena kemungkinan jawaban dari pertanyaan keamanan
7 hanya berupa angka saja, khususnya hanya tanggal ulang tahun, bulan lahir, atau tahun
8 lahir.

9 Responden dapat menjawab tanggal lahir karena hanya memiliki 30-31 kemungkinan,
10 sedangkan untuk bulan hanya ada 12 kemungkinan, dan juga beberapa pertanyaan lain
11 yang menyangkut angka. Dapat dilihat juga, bahwa beberapa jawaban untuk pertanyaan
12 keamanan merupakan informasi yang sering ditunjukkan dalam profil sosial media, karena
13 dari itu jawaban yang tepat bisa dengan mudah didapatkan.

14 Untuk topik kasus 3, tidak ada responden yang berhasil mengembalikan *password*. Hal
15 ini karena beberapa pertanyaan sifatnya sangat personal yang bahkan hanya sebagian dari
16 responden keluarga yang mengetahui jawabannya. Pertanyaan yang sifatnya sangat personal
17 akan mempersulit untuk mengembalikan *password* kecuali bagi pembuat pertanyaan.

18 Dilihat dari grafik 5.16 untuk topik kasus 4, tingkat keberhasilannya tetap tinggi walau-
19 pun topik kasus 4 ini merupakan gabungan dari topik kasus 1, 2, dan 3. Hal ini disebabkan
20 karena mayoritas terdiri pertanyaan dari kasus 1 dan kasus 2 sehingga responden masih bisa
21 mengembalikan *password*.

22 Responden hanya cukup menjawab 6 pertanyaan benar dari 15 pertanyaan dalam kasus
23 4, maka responden pun cukup menjawab 3 pertanyaan dari kasus 1 dan 3 pertanyaan dari
24 kasus 2 dengan benar, responden tidak perlu menjawab satupun pertanyaan dari kasus 3.
25 Dapat disimpulkan, bahwa gabungan tidak meningkatkan keamanan dari *password* untuk
26 bisa dikembalikan hanya oleh pembuat pertanyaan.

27 5.2.5 Kesimpulan Pengujian

28 Dari 4 kasus pengujian yang dilakukan maka bisa ditarik beberapa kesimpulan dalam penila-
29 ian kualitas pertanyaan keamanan personal. Pertanyaan keamanan personal harus memiliki
30 5 sifat:

- 31 • Aman
32 Pertanyaan keamanan harus tidak mudah ditebak dan tidak mudah diselidiki (*gooling*).
33
- 34 • Stabil
35 Pertanyaan keamanan tidak boleh berubah seiring berjalannya waktu.
- 36 • Mudah diingat
37 Pertanyaan keamanan harus sifatnya personal sehingga mudah untuk diingat.
- 38 • Sempel
39 Pertanyaan keamanan harus sederhana tetapi sifatnya tetap personal.
- 40 • Memiliki banyak kemungkinan jawaban
41 Pertanyaan keamanan tidak boleh hanya memiliki kemungkinan jawaban yang sedikit
42 karena akan mudah ditebak (dengan teknik *brute force*).

43 Namun, beberapa pertanyaan keamanan mungkin memiliki banyak kemungkinan jawab-
44 an dan aman sehingga tidak mudah ditebak tetapi tidak mudah diingat karena jawabannya
45 terlalu rumit. Beberapa pertanyaan keamanan juga mungkin tidak sesuai dengan situasi
46 atau keadaan dari pembuat pertanyaan. Sehingga, tidak ada pertanyaan keamanan yang
47 memiliki tepat 5 sifat yang diatas.

BAB 6

KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari penelitian yang dilakukan.

6.1 Kesimpulan

Dari penelitian yang dilakukan, *secret sharing* shamir dapat melindungi *password* dengan cara membagi *password* menjadi beberapa bagian atau *share* sehingga selain *password* juga terlindungi dari pihak yang tidak berhak untuk mengetahui, *password* juga terlindungi dari *human error*, musibah, dan sebagainya yang bisa menyebabkan sebagian dari *password* hilang.

Selain itu, dengan adanya pertanyaan keamanan yang digunakan dalam metode *secret sharing* shamir, kerahasiaan *password* juga terjamin. Melalui penelitian ini, dapat disimpulkan bahwa metode *secret sharing* shamir dapat melindungi *password*.

Kualitas dari pertanyaan keamanan bisa dinilai dari 5 sifat:

- Aman
- Stabil
- Mudah diingat
- Sederhana
- Memiliki banyak kemungkinan jawaban

Dari hasil penelitian juga dapat diketahui bahwa tidak ada pertanyaan keamanan yang memiliki kelima sifat secara sekaligus, beberapa dari sifat ada yang berlawanan sehingga tidak mungkin dapat dimiliki oleh sebuah pertanyaan keamanan sekaligus. Dari hasil penelitian yang dilakukan, dapat disimpulkan bahwa perangkat lunak yang mengimplementasikan *secret sharing* shamir berhasil dibangun.

6.2 Saran

Dari penelitian ini, terdapat beberapa saran untuk pengembangan perangkat lunak lebih lanjut, yaitu:

- Algoritma enkripsi yang digunakan bisa diganti dengan menggunakan algoritma enkripsi yang menggunakan panjang kunci lebih panjang dari 64-bit. Pada penelitian ini, algoritma enkripsi yang digunakan adalah *data encryption standard* (DES) dengan panjang kunci maksimal 64-bit. Untuk ukuran keamanan informasi, 64-bit merupakan ukuran yang kurang dan nantinya untuk pengembangan lebih lanjut bisa digunakan algoritma enkripsi yang memiliki panjang kunci lebih dari 64-bit saja.
- Metode *secret sharing* shamir diharapkan dapat diimplementasikan tidak hanya pada perangkat lunak perorangan seperti dalam penelitian ini, tetapi bisa diimplementasikan pada sebuah sistem besar yang memiliki subsistem dan masing-masing dari subsistem ini menyimpan banyak informasi penting.

DAFTAR REFERENSI

1

- 2 [1] R. Munir, *Matematika Diskrit*. Informatika Bandung, 2010.
- 3 [2] B. A. Forouzan, *Cryptography & Network Security*. McGraw-Hill, Inc., 2007.
- 4 [3] D. Norman and D. Wolczuk, *Introduction to Linear Algebra for Science and Engineering*.
5 Pearson, 2012.
- 6 [4] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–
7 613, 1979.
- 8 [5] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability and statistics for*
9 *engineers and scientists*, vol. 5. Macmillan New York, 1993.
- 10 [6] C. Shannon, “A mathematical theory of communication, bell system technical journal
11 27: 379-423 and 623–656,” *Mathematical Reviews (MathSciNet): MR10, 133e*, 1948.
- 12 [7] C. Ellison, C. Hall, R. Milbert, and B. Schneier, “Protecting secret keys with personal
13 entropy,” *Future Generation Computer Systems*, vol. 16, no. 4, pp. 311–318, 2000.

1

LAMPIRAN A

2

THE PROGRAM