

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	viii
<b>1 PENDAHULUAN</b>	<b>1</b>
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
<b>2 DASAR TEORI</b>	<b>3</b>
2.1 Kriptografi	3
2.1.1 <i>Data Encryption Standard</i> (DES)	5
2.1.2 Pembangunan Kunci Ronde	8
2.2 Fungsi <i>Hash</i>	9
2.2.1 <i>Secure Hashing Algorithm</i> 512 (SHA-512)	10
2.3 Otentikasi	15
2.3.1 Otentikasi Pesan ( <i>Message Authentication</i> )	15
2.3.2 Otentikasi Entitas ( <i>Entity Authentication</i> )	16
2.3.3 <i>Password</i>	17
2.4 <i>Secret Sharing</i>	19
2.4.1 Skema <i>Threshold</i> ( $k,n$ )	19
2.5 Probabilitas	20
2.6 Distribusi Binom	21
2.7 Entropi	21
<b>3 ANALISIS</b>	<b>23</b>
3.1 Studi Kasus	23
3.1.1 <i>Secret Sharing</i> Shamir	23
3.1.2 Pengembangan Algoritma <i>Secret Sharing</i> Shamir	26
3.2 Perancangan Perangkat Lunak	40
3.2.1 Alur Proses	40
3.2.2 Diagram <i>Use Case</i>	42
3.2.3 Diagram Aktivitas	42
3.2.4 Diagram Kelas	43
3.2.5 Arsitektur Perangkat Lunak	44
<b>4 PERANCANGAN</b>	<b>47</b>
4.1 Algoritma	47

4.2	Perancangan Antarmuka . . . . .	48
4.3	Diagram Kelas Rinci . . . . .	49
4.4	Deskripsi Kelas dan Fungsi . . . . .	50
4.4.1	SHA512 . . . . .	50
4.4.2	<i>Function</i> . . . . .	50
4.4.3	<i>EquationSolver</i> . . . . .	50
4.4.4	<i>SecretSharing</i> . . . . .	51
4.4.5	<i>DESEncryption</i> . . . . .	51
4.4.6	<i>DESDecryption</i> . . . . .	51
4.4.7	<i>DataReader</i> . . . . .	52
4.4.8	<i>DataWriter</i> . . . . .	52
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>55</b>
	<b>DAFTAR REFERENSI</b>	<b>57</b>
<b>A</b>	<b>THE PROGRAM</b>	<b>59</b>

## DAFTAR GAMBAR

2.1	Proses enkripsi dan dekripsi . . . . .	4
2.2	Proses enkripsi dengan DES . . . . .	5
2.3	Matriks Permutasi . . . . .	5
2.4	Proses Permutasi . . . . .	6
2.5	Ronde dalam DES . . . . .	7
2.6	Proses permutasi ekspansi . . . . .	7
2.7	<i>P-box</i> . . . . .	8
2.8	Proses substitusi <i>S-box</i> . . . . .	8
2.9	<i>S-box</i> . . . . .	9
2.10	Matriks Permutasi Langsung . . . . .	9
2.11	<i>Round-Key Generator</i> . . . . .	10
2.12	Matriks permutasi untuk <i>Parity drop</i> . . . . .	10
2.13	Matriks kompresi <i>P-box</i> . . . . .	11
2.14	Pembuatan <i>message digest</i> . . . . .	11
2.15	<i>Length field</i> dan <i>padding</i> dalam SHA-512 . . . . .	11
2.16	Blok <i>message</i> dan <i>message digest</i> dalam <i>word</i> . . . . .	12
2.17	Ekspansi <i>word</i> . . . . .	12
2.18	Konstanta inisialisasi dalam SHA-512 . . . . .	13
2.19	Fungsi kompresi dalam SHA-512 . . . . .	13
2.20	Struktur ronde dalam SHA-512 . . . . .	14
2.21	Fungsi kompleks dalam SHA-512 . . . . .	14
2.22	<i>Modification detection code</i> . . . . .	15
2.23	<i>Modification authentication code</i> . . . . .	16
2.24	<i>Username</i> dan <i>Password</i> . . . . .	18
2.25	<i>Password hashing</i> . . . . .	18
2.26	<i>Password salting</i> . . . . .	19
3.1	Proses pembangunan <i>share</i> dari <i>password</i> . . . . .	41
3.2	Proses pembangunan kembali atau rekonstruksi <i>password</i> . . . . .	41
3.3	Diagram <i>use case</i> dari perangkat lunak . . . . .	42
3.4	Diagram aktivitas untuk menyimpan <i>password</i> . . . . .	42
3.5	Diagram aktivitas untuk mengembalikan <i>password</i> . . . . .	43
3.6	Diagram kelas <i>engine</i> . . . . .	44
3.7	Arsitektur perangkat lunak . . . . .	45
4.1	Perancangan Tampilan Awal . . . . .	48
4.2	Perancangan Tampilan Menyimpan <i>Password</i> . . . . .	48
4.3	Perancangan Tampilan Mengembalikan <i>Password</i> . . . . .	49
4.4	Perancangan Tampilan Mengembalikan <i>Password</i> . . . . .	49
4.5	Diagram Kelas Rinci . . . . .	53

## DAFTAR TABEL

3.1 <i>Share dari Data <math>S</math></i> . . . . .	27
---	----



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Enkripsi adalah proses merahasiakan sebuah informasi dengan cara menyandikan informasi tersebut sehingga informasi tersebut tidak dapat dibaca oleh pihak yang tidak berwenang. Dalam proses enkripsi, dibutuhkan sebuah kunci rahasia (*private key*), untuk menyandikan informasi sehingga tidak bisa dibaca dan untuk mengembalikan informasi sehingga bisa kembali dibaca. Proses enkripsi ini memiliki kelemahan, yaitu jika kunci yang digunakan untuk enkripsi hilang maka berakibat informasi yang dienkripsi tidak bisa dikembalikan seperti semula.

Pada umumnya, untuk mengembalikan kunci yang hilang ini, beberapa sistem memiliki mekanisme dengan menyediakan sebuah pertanyaan keamanan yang pertanyaan dan jawabannya sudah dirancang oleh pengguna. Jika pengguna menjawab pertanyaan keamanan ini dengan benar maka pengguna bisa mendapatkan kembali kunci yang hilang. Tetapi,

### 1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini berupa:

- Bagaimana cara melindungi *password* dengan *secret sharing* shamir?
- Bagaimana cara mengimplementasikan *secret sharing* shamir pada perangkat lunak?

### 1.3 Tujuan

Tujuan penelitian ini berupa:

- Mempelajari cara kerja *secret sharing* shamir dalam melindungi *password*.
- Membangun perangkat lunak yang mengimplementasikan *secret sharing* shamir

### 1.4 Batasan Masalah

Batasan masalah pada penelitian ini berupa:

- Setiap pertanyaan selalu dijawab dengan jawaban yang relevan dengan pertanyaan.

### 1.5 Metodologi Penelitian

Metodologi dalam penelitian ini berupa:

- Melakukan studi literatur mengenai *secret sharing* shamir
- Melakukan studi literatur mengenai algoritma enkripsi *data encryption standard* (DES)

- 1     • Melakukan studi literatur mengenai *secure-hash-algorithm-512* (SHA-512)
- 2     • Melakukan analisis dan perancangan mengenai perangkat lunak yang akan dibangun
- 3     • Implementasi terhadap hasil analisis dan perancangan perangkat lunak
- 4     • Melakukan pengujian perangkat lunak

## 5   1.6   Sistematika Pembahasan

6   Sistematika pembahasan dalam penelitian ini berupa:

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

## BAB 2

### DASAR TEORI

#### 2.1 Kriptografi

Kriptografi merupakan kata berasal dari bahasa Yunani, yaitu krip-  
to berarti rahasia dan graphia berarti tulisan. Jadi, kriptografi adalah ilmu sekaligus seni  
untuk menjaga keamanan pesan. Keamanan pesan diperoleh dengan menyandikan  
pesan menjadi tidak bermakna atau tidak memiliki arti. Zaman sekarang ini, kerahasiaan  
informasi menjadi suatu hal yang penting. Informasi yang sifatnya rahasia atau personal  
perlu dilindungi dari orang-orang yang tidak berhak untuk membacanya. Kriptografi digu-  
nakan untuk menyamarkan informasi rahasia itu dari orang atau pihak yang tidak berhak  
untuk membaca atau melihatnya.

Kriptografi memiliki 4 layanan utama:

1. Kerahasiaan Data (*data confidentiality*)

Layanan ini menjamin bahwa data atau pesan yang dikirimkan tidak diketahui oleh  
pihak atau orang lain yang tidak berhak untuk membaca atau melihatnya.

2. Integritas Data (*data integrity*)

Layanan yang menjamin data atau pesan yang dikirimkan tidak boleh diubah tanpa  
seijin pemilik pesan, menjamin keaslian dari data atau pesan yang dikirimkan.

3. Otentikasi (*authentication*)

Layanan yang digunakan untuk memvalidasi identitas seseorang, entitas, atau asal  
informasi yang dikirim. Autentikasi dibagi ke dalam 2 jenis dilihat dari subjek yang  
diotentikasinya, yaitu otentikasi entitas dan otentikasi pesan.

4. Non-repudiasi (*nonrepudiation*)

Layanan yang menjamin bahwa tidak ada penyangkalan baik oleh pengirim atau pe-  
nerima pesan.

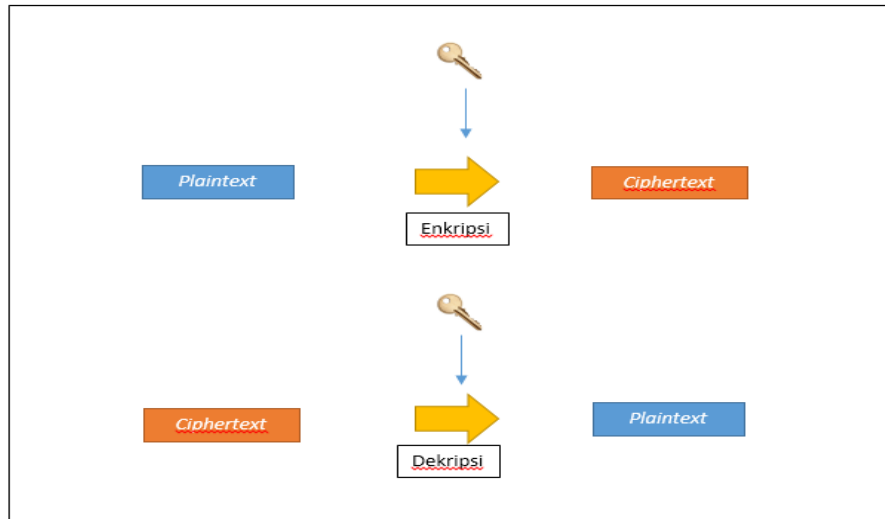
Dalam kriptografi, pesan yang dirahasiakan disebut *plaintext*, sedangkan pesan hasil  
dari penyandian disebut *ciphertext*. Pesan yang telah disandikan dapat dikembalikan lagi  
ke pesan aslinya hanya oleh orang yang berhak. Orang yang berhak adalah orang yang  
mengetahui cara penyandian atau memiliki kunci penyandian. Proses menyandikan *plain-  
text* menjadi *ciphertext* disebut enkripsi (*encryption*) dan proses mengembalikan *ciphertext*  
menjadi *plaintext* disebut dekripsi (*decryption*). Dalam proses enkripsi dan dekripsi meng-  
gunakan kunci (*key*) yaitu sekumpulan huruf, angka, atau simbol. Kunci ini sifatnya rahasia  
dan tidak boleh diketahui pihak lain yang tidak berwenang.

Gambar 2.1 ini menunjukkan proses enkripsi dan dekripsi.

Dalam proses enkripsi, *plaintext* akan dipetakan pada fungsi enkripsi  $E$  menjadi *cipher-  
text* didasarkan pada kunci  $k$  seperti notasi di bawah ini:

$$E_k(\text{plaintext}) = \text{ciphertext}$$

Kemudian dalam proses dekripsi, untuk mengembalikan *ciphertext* ke *plaintext* maka  
*ciphertext* akan dipetakan pada fungsi dekripsi  $D$  didasarkan pada kunci  $k$  seperti notasi di



Gambar 2.1: Proses enkripsi dan dekripsi

bawah ini:

$$D_k(\text{ciphertext}) = \text{plaintext}$$

Sebagai contoh, *plaintext* yang akan dikirimkan sebagai berikut adalah kriptografi disandikan menjadi *ciphertext* dengan fungsi enkripsi  $E$  didasarkan pada kunci  $k$  menjadi:

$$E_k(\text{kriptografi}) = u8@37md$$

*ciphertext* diatas, meskipun tidak dirahasiakan, namun isinya sudah tidak jelas dan tidak dapat dimengerti maksudnya. Hanya orang yang berhak yang dapat mengembalikan *ciphertext* menjadi *plaintext*. Kemudian untuk mengembalikan *ciphertext* ke *plaintext*, *ciphertext* akan dipetakan pada fungsi dekripsi  $D$  didasarkan pada kunci  $k$  menjadi:

$$D_k(u8@37md) = \text{kriptografi}$$

1 Algoritma kriptografi ini dibagi menjadi 2 jenis, yaitu:

2 1. Kriptografi kunci simetris (*symmetric key cryptography*)

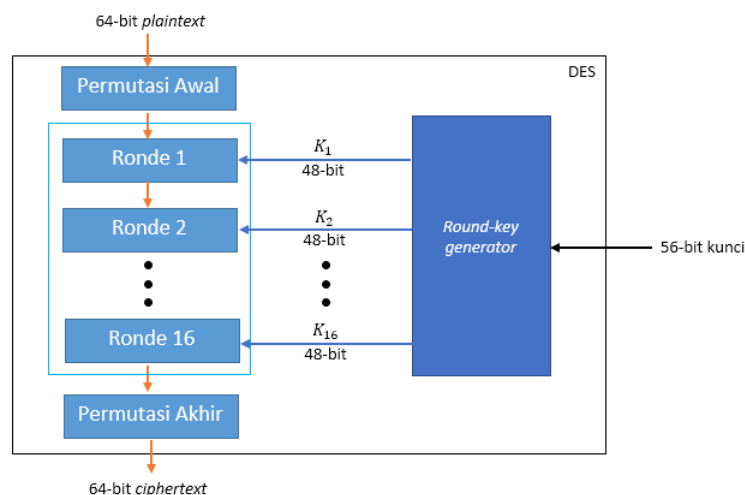
3 Kriptografi kunci simetris atau penyandian kunci simetris menggunakan hanya satu  
4 kunci rahasia. Pengirim pesan mengenkripsi pesan dengan kunci  $k$ , kemudian penerima  
5 pesan juga akan mendekripsi pesan yang diterima dengan kunci  $k$ . Dalam hal ini,  
6 pengirim dan penerima keduanya harus memiliki kunci  $k$ . Kelemahan dari kriptografi  
7 kunci simetris adalah baik pengirim dan penerima harus memiliki kunci yang sama,  
8 sehingga pengirim harus mencari cara lain untuk memberitahukan kunci kepada penerima.  
9 Contoh algoritma kunci simetris antara lain adalah *Data Encryption Standard* (DES),  
10 *Advanced Encryption Standard* (AES), *Twofish*, dan *Blowfish*.

11 2. Kriptografi kunci asimetris (*asymmetric key cryptography*)

12 Dalam kriptografi kunci asimetris atau penyandian kunci simetris kunci rahasia yang  
13 digunakan ada 2 buah kunci, yaitu kunci publik (*public key*) dan kunci pribadi (*private*  
14 *key*). Kunci publik tidak rahasia dan dapat diketahui secara umum, sedangkan kunci  
15 pribadi harus dirahasiakan oleh pengirim pesan. Pengirim pesan akan mengenkripsi  
16 pesan yang dikirim dengan kunci publik penerima pesan, kemudian penerima pesan  
17 akan mendekripsi pesan menggunakan kunci pribadinya yang hanya diketahui oleh  
18 dirinya saja. Contoh algoritma kunci asimetris antara lain adalah Rivest-Shamir-  
19 Adleman (RSA), ElGamal, Diffie-Helman, *Digital Signature Algorithm*, dan *Elliptic*  
20 *Curve Digital Signature Algorithm* (ECDSA).

### 2.1.1 Data Encryption Standard (DES)

*Data Encryption Standard* (DES) adalah algoritma kriptografi kunci simetris, yaitu menggunakan kunci yang sama pada proses enkripsi dan dekripsinya. Masukan dari DES berupa 64-bit *plaintext* dan keluarannya berupa 64-bit *ciphertext* dengan 64-bit kunci. Proses enkripsi terdiri dari 2 proses permutasi, yaitu permutasi awal dan permutasi akhir, dan 16 ronde sandi feistel. Setiap ronde menggunakan kunci 48-bit yang berbeda-beda. Kunci dari setiap ronde ini akan didapat dari round-key generator yang berdasarkan pada 64-bit kunci sandi. Gambar di bawah ini menunjukkan proses enkripsi dari DES.



Gambar 2.2: Proses enkripsi dengan DES

#### Permutasi Awal dan Permutasi Akhir

Permutasi awal dan akhir dalam DES menggunakan matriks permutasi. Kedua permutasi ini menggunakan 64-bit masukan dan matriks yang terdiri 64 nilai yang sudah ditentukan sebelumnya. Matriks ini nilainya selalu sama untuk setiap proses enkripsi dan tidak ada aturan tertentu untuk membuatnya. Gambar di bawah ini menunjukkan salah satu contoh matriks permutasi awal dan matriks permutasi akhir.

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

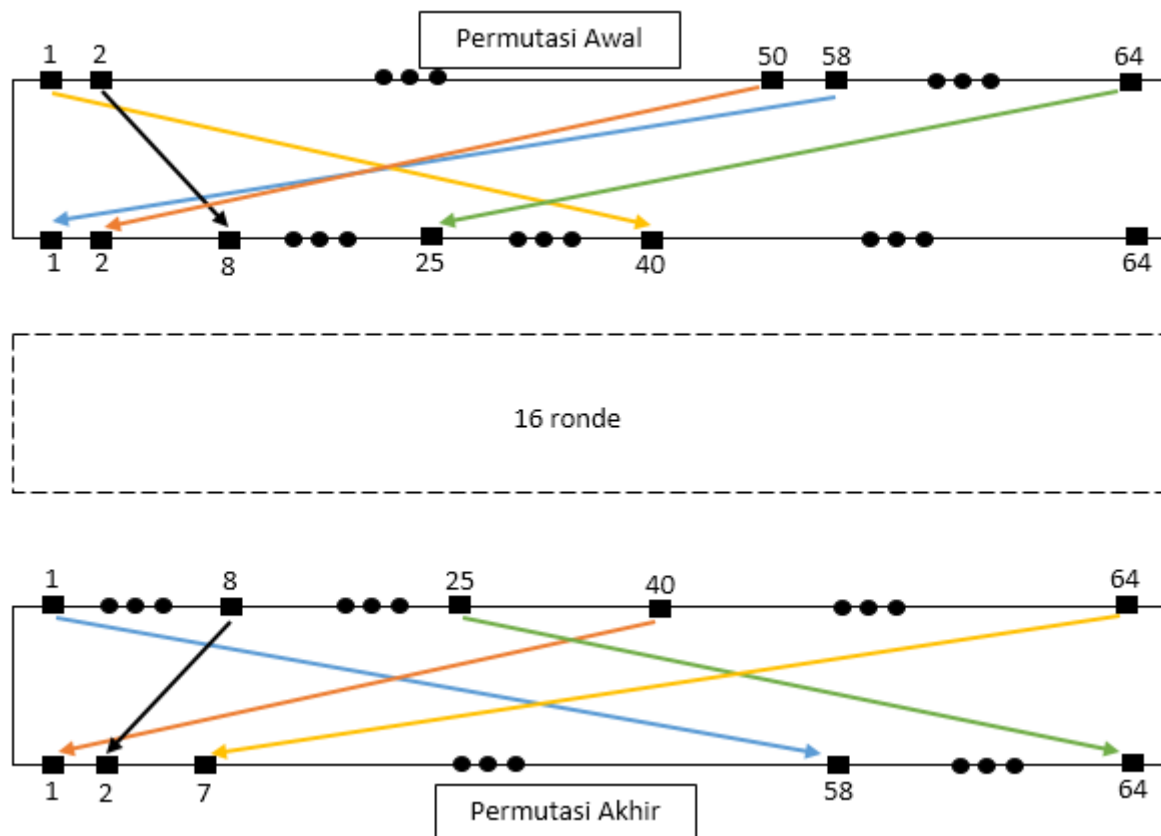
Permutasi Akhir							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Gambar 2.3: Matriks Permutasi

Sebagai contoh, *bit* ke-58 dari input akan menjadi *bit* ke-1 pada output permutasi awal, *bit* ke-50 akan menjadi *bit* ke-2 pada output permutasi awal dan seterusnya. Cara yang sama akan diterapkan pada proses permutasi akhir nanti setelah 16 ronde sandi feistel.

#### Ronde

DES menggunakan 16 ronde. Setiap ronde dari DES adalah sandi feistel yang ditunjukkan pada gambar di bawah.



Gambar 2.4: Proses Permutasi

Setiap ronde menggunakan  $L_{I-1}$  dan  $R_{I-1}$  dari ronde sebelumnya (atau dari permutasi awal) dan memrosesnya menjadi  $L_I$  dan  $R_I$  untuk nanti dilanjutkan ke proses yang berikutnya (atau permutasi akhir).  $f(R_{I-1}, K_I)$  merupakan fungsi DES. Plaintext (64-bit) akan dibagi 2 menjadi bagian kiri ( $L$ ) dan bagian kanan ( $R$ ). Bagian kanan akan diproses lagi oleh fungsi  $f(R_{I-1}, K_I)$ . Kemudian, hasil dari fungsi  $f(R_{I-1}, K_I)$  akan di XOR dengan bagian kiri. Hasil dari XOR akan menjadi  $R_I$  dan hasil dari fungsi  $f(R_{I-1}, K_I)$  akan menjadi  $L_I$ . Pada ronde ke-16 tidak akan terjadi pertukaran antara bagian  $L_{I-1}$  dan  $R_{I-1}$ .

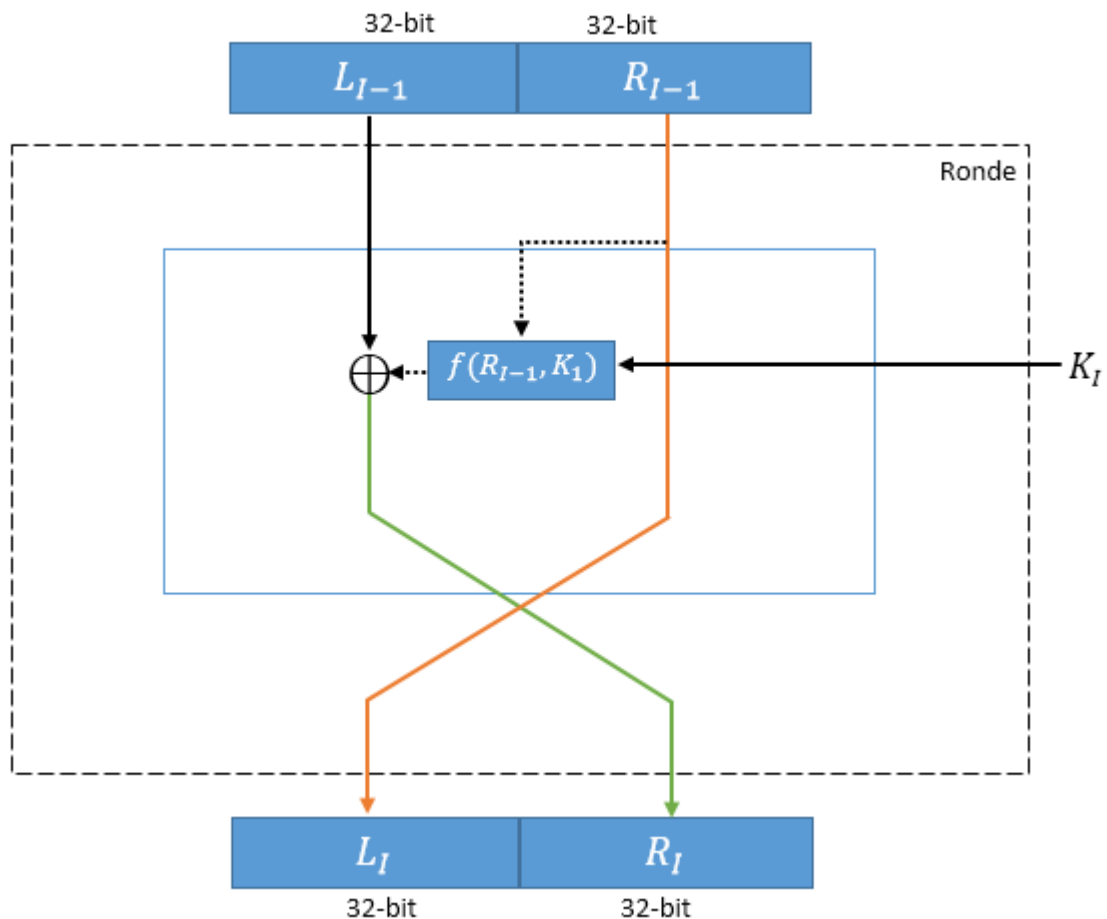
## 8 Fungsi DES

Fungsi DES merupakan inti dari algoritma DES. Fungsi DES menerima masukan kunci ronde 48-bit dan bagian kanan dari plaintext,  $R_{I-1}$  dan menghasilkan 32-bit keluaran. Fungsi ini terdiri dari 4 bagian, yaitu ekspansi *P-box*, operasi XOR, substitusi *S-box*, dan permutasi.

### 1. Ekspansi *P-box*

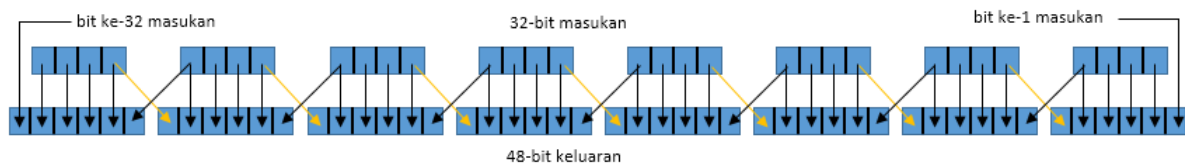
Pada bagian ini, karena  $R_{I-1}$  adalah input dengan panjang 32-bit dan kunci ronde yang digunakan sepanjang 48-bit, maka  $R_{I-1}$  perlu diekspansi menjadi 48-bit.  $R_{I-1}$  akan dibagi menjadi 8 bagian masing-masing 4-bit. Kemudian setiap bagian 4-bit ini akan diekspansi menjadi 6-bit. Ekspansi ini dilakukan berdasarkan aturan yang sudah ditentukan terlebih dahulu, yaitu

- Bit keluaran ke-2, 3, 4, dan 5 diisi oleh bit masukan ke-1, 2, 3, dan 4.
- Bit keluaran ke-1 akan diisi oleh bit masukan ke-4 pada bagian sebelumnya.
- Bit keluaran ke-6 akan diisi oleh bit masukan ke-1 pada bagian sesudahnya.
- Untuk bagian ke-1, bit keluaran ke-1 akan diisi oleh bit ke-32 masukan.
- Untuk bagian ke-8, bit keluaran ke-6 akan diisi oleh bit ke-1 masukan.



Gambar 2.5: Ronde dalam DES

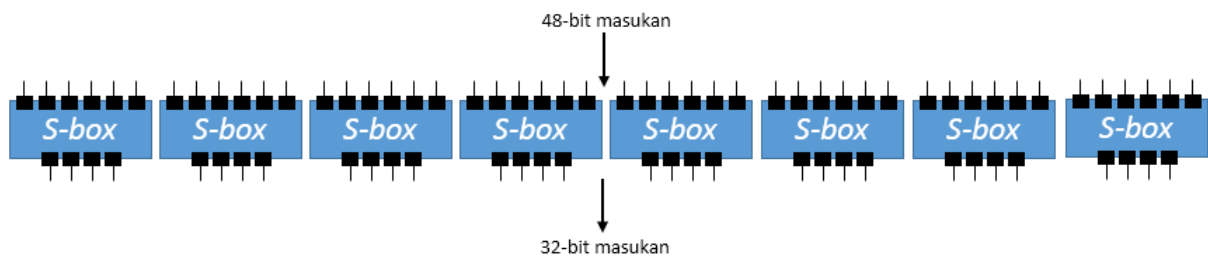
- 1 Ekspansi ini dilakukan berdasarkan nilai-nilai yang ada dalam *P-box*. Gambar di
- 2 bawah ini menunjukkan proses ekspansi dan *P-box*.



Gambar 2.6: Proses permutasi ekspansi

- 3 2. Operasi XOR
- 4 Setelah dilakukan ekspansi dengan menggunakan *P-box*, dilakukan operasi XOR antara
- 5  $R_{I-1}$  dengan kunci ronde. Perlu diketahui, bahwa panjang  $R_{I-1}$  dan kunci ronde sudah
- 6 sama, yaitu 48-bit dan kunci ronde hanya digunakan dalam proses ini saja.
- 7 3. Substitusi *S-box*
- 8 DES menggunakan 8 buah *S-box*, masing-masing dengan 6-bit masukan dan 4-bit
- 9 keluaran. Gambar di bawah ini menunjukkan proses substitusi *S-box* dan salah satu
- 10 contoh dari *S-box*.
- 11 Pada bagian ini, akan dilakukan operasi substitusi berdasarkan aturan pada *S-box*.
- 12 Setiap *S-box* adalah matriks berukuran 4x16 (4 baris, 16 kolom). Setiap isi dari *S-box*
- 13 akan berbeda-beda. Kombinasi biner dari *bit* ke-1 dan ke-6 masukan menunjukkan
- 14 posisi baris yang akan dipilih dan kombinasi biner dari *bit* masukan sisanya menun-
- 15 jukkan posisi kolom yang akan dipilih. Setelah itu, angka yang ditunjuk oleh baris dan

<i>P-box</i>					
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	31	31	32	1

Gambar 2.7: *P-box*Gambar 2.8: Proses substitusi *S-box*

- 1 kolom kombinasi biner ini akan diubah juga menjadi biner.
- 2 Sebagai contoh, misalkan masukan untuk *S-box* diatas adalah 110011. Maka, baris
- 3 yang dipilih adalah 11 dalam biner dan 3 dalam desimal dan kolom yang dipilih adalah
- 4 1001 dalam biner dan 9 dalam desimal. Berarti, nilai yang didapat berdasarkan *S-box*
- 5 diatas adalah 11 dan dalam biner adalah 1011. Maka, keluaran dari 110011 adalah
- 6 1011.
- 7 4. Permutasi
- 8 Bagian terakhir dari fungsi DES adalah permutasi langsung dengan masukan 32-*bit*
- 9 dan keluaran 32-*bit*. Proses nya sama dengan permutasi yang dilakukan pada awal
- 10 proses enkripsi tetapi ukuran matriks permutasi 4x8 (4 baris, 8 kolom). Gambar di
- 11 bawah ini menunjukkan matriks permutasi langsung.
- 12 Sama halnya seperti proses permutasi yang dilakukan di awal enkripsi, angka 16 me-
- 13 nunjukkan bahwa *bit* ke-16 masukan akan menjadi *bit* ke-1 keluaran, *bit* ke-7 masukan
- 14 akan menjadi *bit* ke-2 keluaran, dan seterusnya.

### 2.1.2 Pembangunan Kunci Ronde

Pembangunan kunci ronde adalah algoritma untuk membentuk kunci yang akan digunakan pada setiap ronde. Pembangunan kunci ronde akan menghasilkan 16 kunci masing-masing dengan panjang 48-*bit* dari masukan 56-*bit* kunci sandi. Tetapi, sebenarnya panjang sandi kunci ini adalah 64-*bit* dengan 8-*bit parity bit*. *Parity bit* akan dihilangkan sebelum proses pembangkitan kunci sandi 56-*bit*. Proses pembangkitan kunci sandi 56-*bit* bisa dilihat pada gambar di bawah.

#### *Parity Drop*

Proses yang dilakukan sebelum pembangkitan kunci sandi adalah penghilangan *parity bit* pada kunci sandi 64-*bit*. Bit yang dihilangkan adalah *bit* ke-8, 16, 24, 32, ..., 64 sehingga hasil akhirnya adalah 56-*bit* sandi kunci yang nanti kunci ini akan digunakan untuk membuat



	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

Gambar 2.9: *S-box*

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

Gambar 2.10: Matriks Permutasi Langsung

1 kunci setiap ronde. Matriks permutasi yang digunakan pada proses ini ditunjukkan pada  
2 gambar di bawah.

3  
4  
5  
6

### 7 *Shift Left*

8 Pada tahap ini, kunci yang akan dibentuk akan dipisah menjadi 2 bagian masing-masing  
9 28-bit. Setiap bagian akan digeser ke arah kiri secara sirkular sebanyak 1 atau 2 bit. Ronde  
10 ke-1, 2, 9, dan 16 akan digeser sebanyak 1 bit, selain itu hanya akan digeser sebanyak 2 bit.  
11 Kemudian, kedua bagian ini akan disatukan kembali menjadi satu bagian dengan panjang  
12 56-bit.

### 13 *Kompresi P-box*

14 Kompresi dengan *P-box* adalah tahap permutasi untuk mengubah kunci 56-bit menjadi 48-  
15 bit agar bisa digunakan sebagai ronde kunci. Kompresi ini menggunakan matriks permutasi  
16 *P-box* dengan ukuran 6x8 seperti gambar yang ditunjukkan di bawah ini.

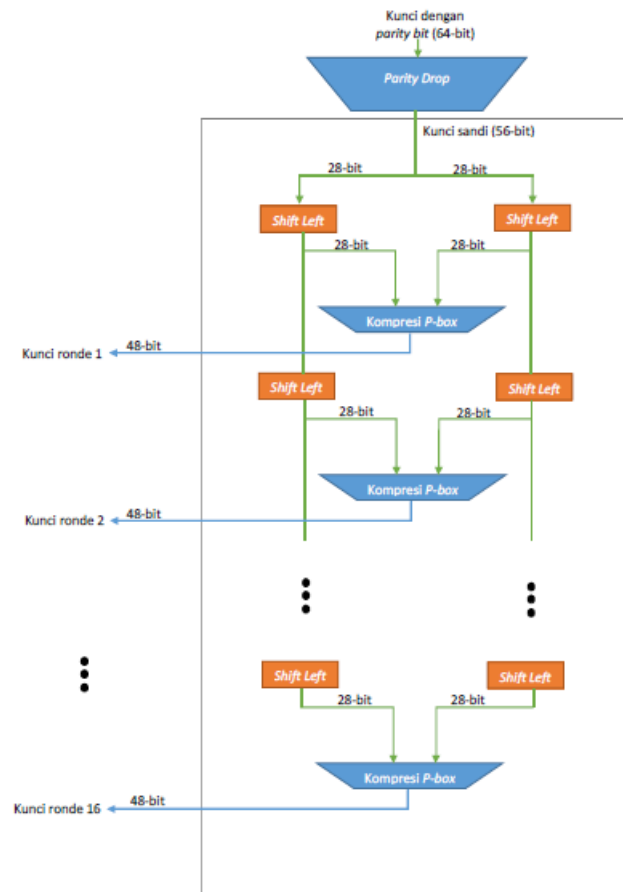
## 17 2.2 Fungsi *Hash*

Fungsi *hash* adalah algoritma kriptografi yang membuat kompresi dari pesan (*message*) atau inti dari pesan, dimana kompresi ini dapat berfungsi sebagai *fingerprint* atau disebut juga *digest*. Fungsi *hash* juga berfungsi untuk menjaga integritas sebuah pesan agar ketika ada perubahan pada pesan tersebut dapat langsung diketahui. Fungsi *hash* akan memroses message (*m*) menjadi *message digest* atau *digest* (*h*). Bentuk umum dari fungsi *hash*, yaitu:

$$h = H(m)$$

18 Fungsi *hash* memiliki 4 kriteria utama, yaitu:

- 19 • Semua nilai hash memiliki panjang yang sama.
- 20 • Mudah untuk menghitung nilai *hash* untuk setiap *message* masukkan.
- 21 • Nilai *hash* tidak bisa dikembalikan menjadi *message*.
- 22 • Tidak ada 2 atau lebih *message* yang memiliki nilai *hash* yang sama.



Gambar 2.11: Round-Key Generator

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

Gambar 2.12: Matriks permutasi untuk Parity drop

- Saat *message* berubah maka nilai hashnya juga akan berubah.
- Contoh fungsi *hash* antara lain SHA-0, SHA-1, SHA-512, MD-2, dan MD-5.

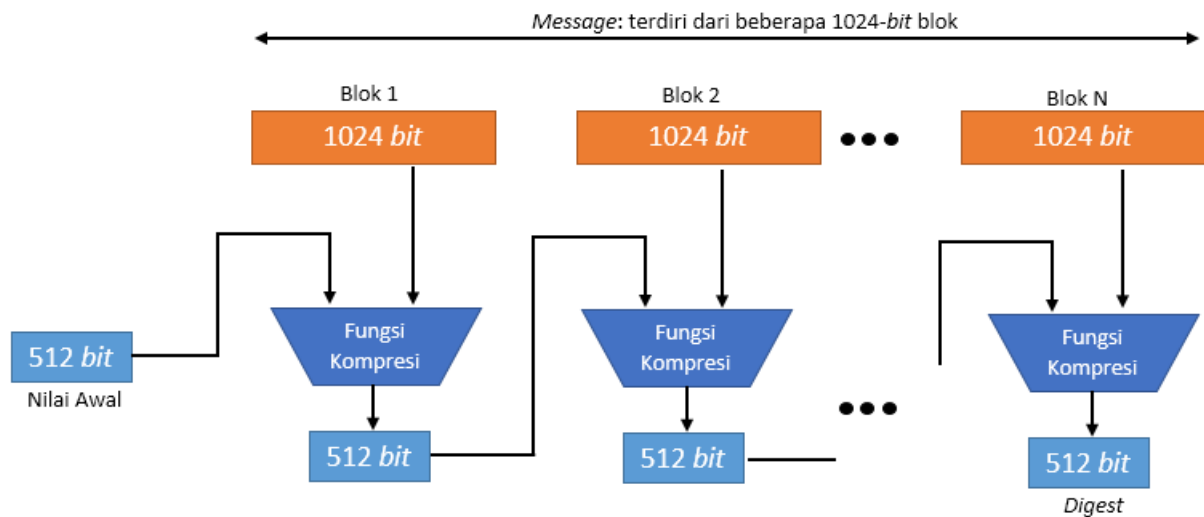
### 2.2.1 Secure Hashing Algorithm 512 (SHA-512)

Secure hashing algorithm 512 atau SHA-512 adalah algoritma fungsi *hash* yang menghasilkan 512-bit *message digest*. SHA-512 merupakan algoritma fungsi *hash* dalam versi SHA yang menghasilkan *message digest* paling panjang (512-bit). SHA-512 membuat 512-bit *digest* yang diambil dari beberapa blok *message*. Setiap blok *message* panjangnya 1024 bit. Gambar di bawah ini menunjukkan cara pembuatan *message digest*.

#### Persiapan Message

SHA-512 menerima masukan *message* dengan panjang kurang dari  $2^{128}$  – bit, berarti jika panjang *message* sama dengan  $2^{128}$  atau lebih maka SHA-512 tidak memroses *message*

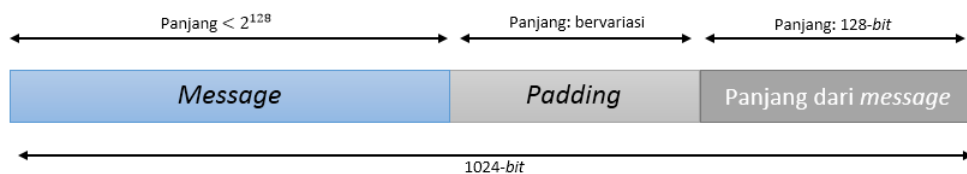
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
32	29	36	50	42	46	53	34

Gambar 2.13: Matriks kompresi *P-box*Gambar 2.14: Pembuatan *message digest*

- 1 tersebut. Hal ini tidak akan menjadi masalah karena ukuran  $2^{128}$  – *bit* cukup besar untuk  
 2 tempat penyimpanan komputer manapun.

### 3 *Length Field* dan *Padding*

- 4 Sebelum *message digest* atau *digest* dapat dibuat, SHA-512 membutuhkan tambahan 128-  
 5 *bit* yang menunjukkan panjang dari *message*. Panjang dari *message* ini direpresentasikan  
 6 oleh 128-*bit* tambahan.

Gambar 2.15: *Length field* dan *padding* dalam SHA-512

- 7 Sebelum menambahkan bagian 128-*bit* yang menunjukkan panjang dari *message* maka  
 8 kita perlu melapisi (*padding*) *message* asli agar panjang totalnya bisa mencapai 1024-*bit*.  
 9 Panjang dari bagian *padding* yang harus ditambahkan ini ditunjukkan dengan cara sebagai  
 10 berikut, dengan *P* adalah *padding*, *M* adalah *message*.

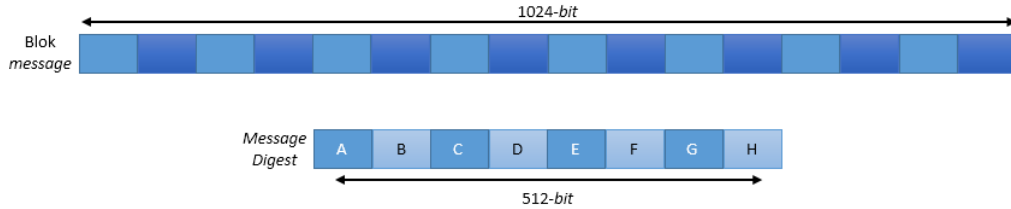
$$(M + P + 128) = 0 \mod 1024$$

$$P = (-M - 128) \mod 1024$$

- 1 Cara menuliskan *padding* diawali dengan angka 1 kemudian diikuti oleh beberapa angka  
 2 0 (nol) sampai panjangnya mencapai  $P$ .

### 3 **Words**

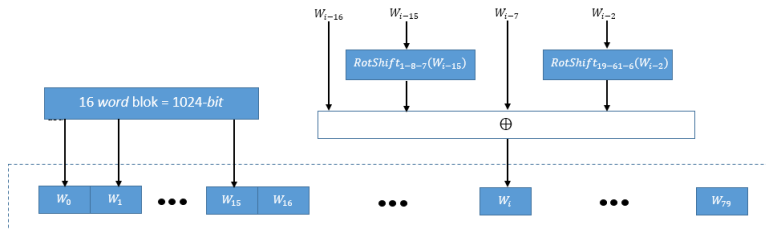
- 4 SHA-512 beroperasi dengan ukuran *words*. Panjang dari satu *word* adalah 64-bit. Ini berarti,  
 5 setelah proses *padding* ditambahkan pada *message*, setiap blok *message* akan terdiri dari 16  
 6 buah 64-bit *word*. *Message digest* atau *digest* juga akan terdiri dari 64-bit *word* namun  
 7 panjangnya hanya 8 *word* dan setiap *word* nya dinamakan  $A, B, C, D, E, F, G$ , dan  $H$ .



Gambar 2.16: Blok *message* dan *message digest* dalam *word*

### 8 **Ekspansi Word**

- 9 Sebelum diproses, setiap blok *message* harus diekspansi, panjang setiap blok ini adalah  
 10 1024-bit terdiri dari 16 buah 64-bit *word*. Pada tahap pemrosesan, dibutuhkan 80 buah  
 11 64-bit *word* maka dari itu, 16 *word* ini harus diekspansi menjadi 80 *word*,  $W_0$  sampai  $W_{79}$ .  
 12 Gambar di bawah ini akan menunjukkan cara ekspansi *word*.



Gambar 2.17: Ekspansi *word*

$$RotShift_{t_{l-m-n}}(x) : RotR_l(x) \oplus RotR_m(x) \oplus ShL_n(x)$$

- 13  $RotR_i(x)$ : Rotasi ke kanan dari  $x$  sebanyak  $i$  bit.  
 14  $ShL_i(x)$ : Shift left  $x$  sebanyak  $i$  bit ditambah (*padding*) dengan 0.

### 15 **Inisialisasi Message Digest**

- 16 SHA-512 menggunakan 8 buah konstanta dalam proses inisialisasi *message digest*. Konstanta  
 17 ini akan diberi nama  $A_0$  sampai  $H_0$ . Setiap nilai dari konstanta didapat dari 8 bilangan prima  
 18 pertama (2, 3, 5, 7, 11, 13, 17, 19). Setiap nilai merupakan nilai pecahan atau angka di  
 19 belakang koma dari akar kuadrat bilangan prima yang bersangkutan.

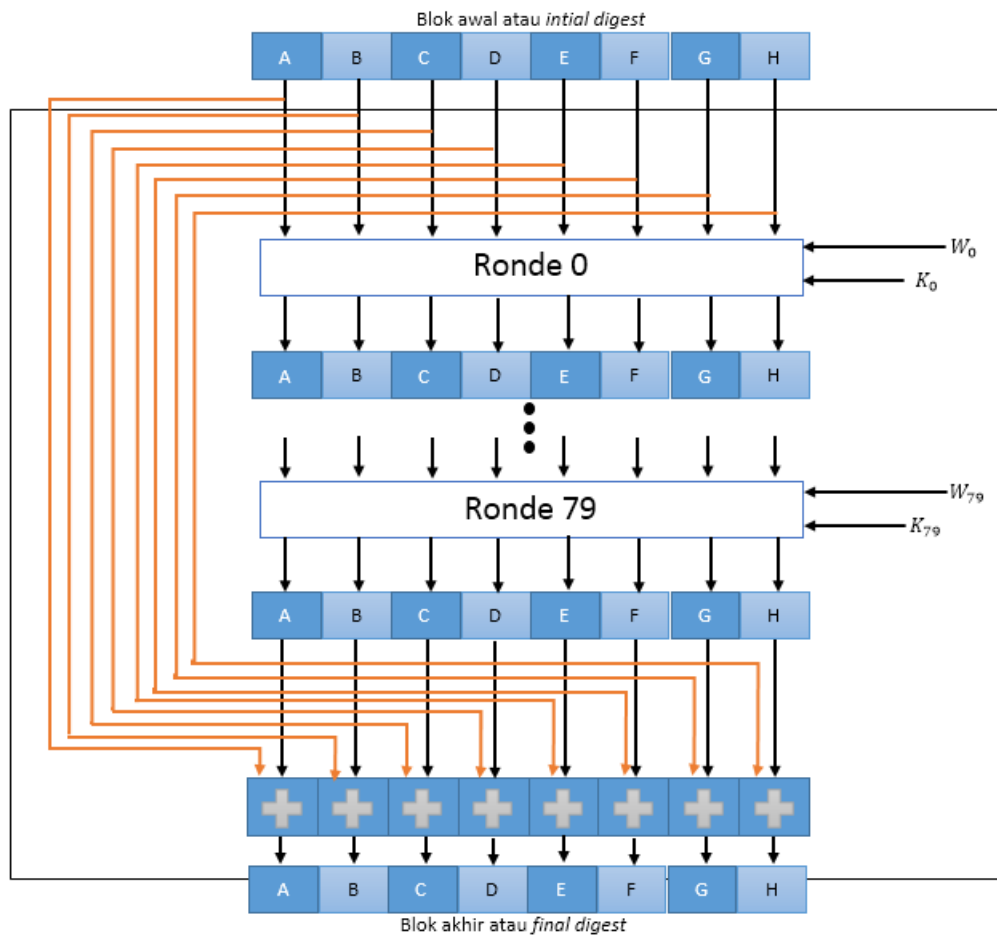
- 20 Sebagai contoh,  $H_0$  didapat dari nilai  $\sqrt{19} = 4.35889894354$ . Kemudian, nilai ini akan  
 21 dikonversi ke biner sepanjang 64-bit maka akan diperoleh  $(100.0101 \ 1011 \ 1110 \dots 1001)_2 =$   
 22  $(4.5BE0CD19137E2179)_{16}$ . SHA-512 hanya mengambil nilai yang di belakang koma saja  
 23 jadi nilai dari  $H_0 = 5BE0CD19137E2179$ . Gambar di bawah ini menunjukkan nilai dari  
 24  $A_0$  sampai  $H_0$  dalam heksadesimal.

Konstanta	Nilai	Konstanta	Nilai
$A_0$	6A09E667F3BCC908	$E_0$	510E527FADE682D1
$B_0$	BB67AE8584CAA73B	$F_0$	9B05688C2B3E6C1F
$C_0$	3C6EF372EF94F828	$G_0$	1F83D9ABFB41BD6B
$D_0$	A54FE53A5F1D36F1	$H_0$	5BE0CD19137E2179

Gambar 2.18: Konstanta inialisasi dalam SHA-512

### 1 Fungsi Kompresi

SHA-512 membuat *message digest* sepanjang 512-bit (8 word) dari beberapa blok *message* yang masing-masing panjangnya 1024-bit. Pemrosesan setiap blok *message* ini dilakukan sebanyak 80 ronde. Setiap ronde akan terdiri dari  $W_i$  digabungkan dengan  $K_i$  untuk membuat  $W_i$  baru yang akan digunakan pada ronde selanjutnya. Pada akhir ronde 79, nilai  $W_0$  sampai  $W_7$  pada ronde 79 akan ditambahkan oleh nilai  $W_0$  sampai  $W_7$  pada ronde yang paling awal (ronde 0) dan dimodulo oleh  $2^{64}$ . Hasil yang diperoleh disebut *message digest* atau *final digest*. Gambar di bawah ini menunjukkan proses fungsi kompresi pada SHA-512.

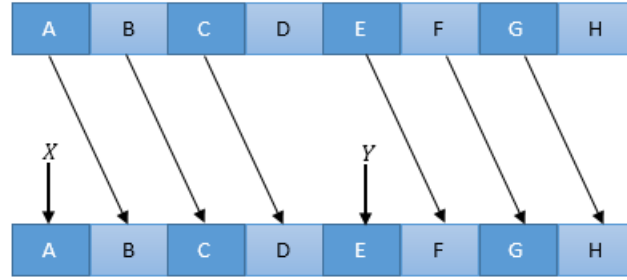


Gambar 2.19: Fungsi kompresi dalam SHA-512

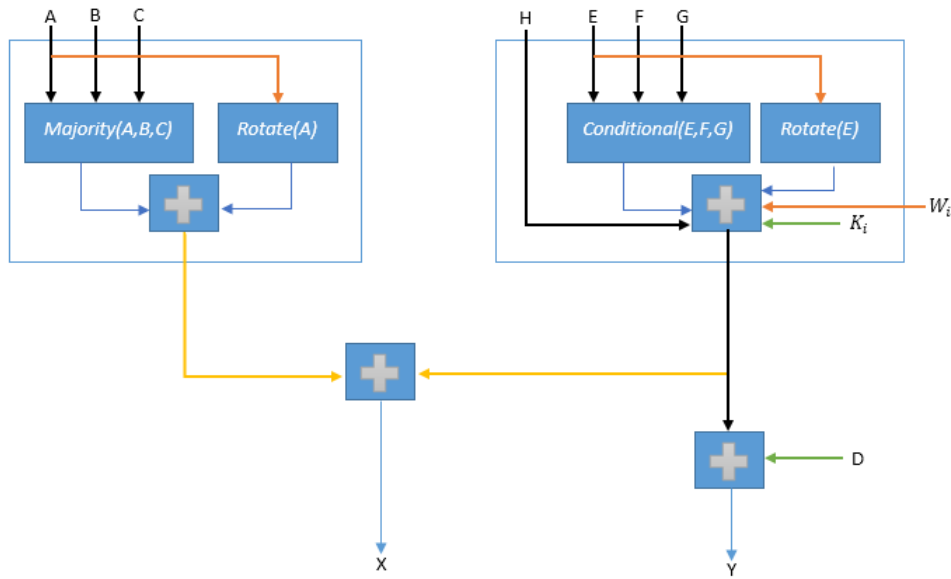
### 9 Ronde

Pada setiap ronde, 8 nilai baru pada 64-bit word dibuat dari 8 nilai 64-bit word pada ronde sebelumnya. Nilai baru ini dinamakan *buffer*. Enam dari delapan *buffer* mendapatkan nilai dari *buffer* ronde sebelumnya. Sedangkan untuk 2 *buffer* lagi nilai keduanya diperoleh dari sebuah fungsi kompleks yang berhubungan dengan *buffer* sisanya serta gabungan tambahan dari  $W_i$  dan  $K_i$ , bersangkutan dengan masing-masing ronde. Gambar di bawah ini

- 1 akan menjelaskan struktur dari setiap ronde serta fungsi kompleks yang digunakan untuk
- 2 mendapatkan nilai untuk 2 *buffer* ( $A$  dan  $E$ ).



Gambar 2.20: Struktur ronde dalam SHA-512



Gambar 2.21: Fungsi kompleks dalam SHA-512

Nilai dari  $Majority(A, B, C)$ ,  $Conditional(E, F, G)$ ,  $Rotate(A)$ , dan  $Rotate(E)$  pada gambar di atas didapat dari rumus di bawah ini.

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

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

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

- 3  $RotR_i(x)$  adalah rotasi ke kanan  $x$  sebanyak  $i$  *bit*. Simbol kotak yang berisi tanda
- 4 tambah berarti adalah penambahan antar *bit-bit* lalu kemudian hasilnya di modulo oleh
- 5  $2^{64}$ .  $K_i$  pada fungsi kompleks diatas didapat dengan cara yang sama saat mencari nilai dari
- 6 konstanta awal  $A_0$  sampai  $H_0$ . Namun, karena yang digunakan sebanyak 80 buah dari  $K_0$
- 7 sampai  $K_{79}$  maka banyak bilangan prima yang digunakan juga sebanyak 80 bilangan prima
- 8 pertama mulai dari 2 sampai 409.

- 9 Perbedaanannya dengan nilai dari konstanta awal adalah nilai  $K_i$  pada fungsi kompleks
- 10 didapat dari akar kubik bilangan prima yang bersangkutan. Selanjutnya, proses yang sa-
- 11 ma dengan cara menentukan nilai konstanta awal tetap digunakan untuk menentukan nilai

1  $K_i$ . Sebagai contoh,  $\sqrt[3]{409} = 7.42291412044$ , kemudian akan dikonversi ke dalam biner  
 2 menjadi  $(111.0110\ 1100\ 0100\ 0100\ \dots 0111)_2$  dan dikonversi ke dalam heksadesimal menjadi  
 3  $(7.6C44198C4A475817)_{16}$ , kemudian nilai yang diambil hanya nilai di belakang koma, maka  
 4 nilai dari  $K_{79} = 6C44198C4A475817$ .

## 5 2.3 Otentikasi

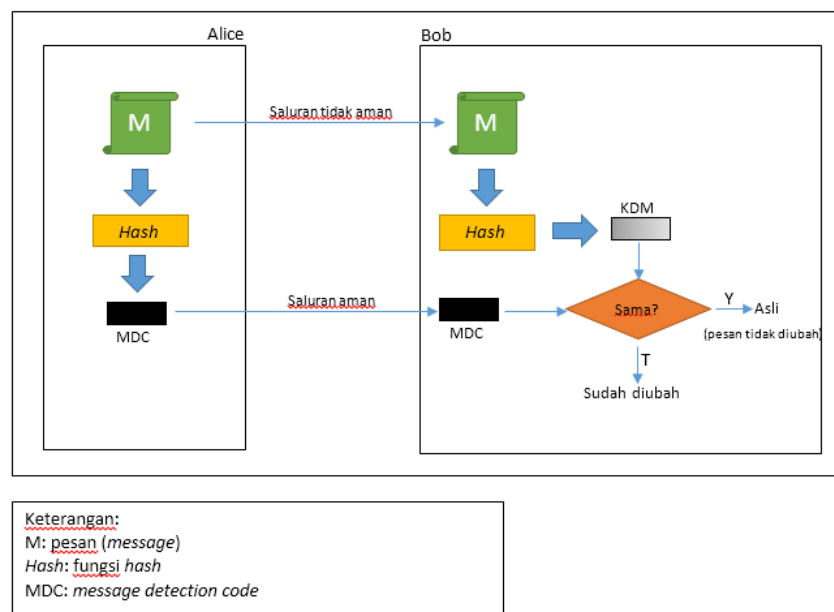
6 Otentikasi adalah proses untuk menentukan apakah sebuah entitas diijinkan untuk meng-  
 7 akses sumber daya yang dimiliki sebuah sistem. Otentikasi juga merupakan proses untuk  
 8 memastikan keaslian data. Otentikasi dibagi menjadi 2 jenis, yaitu:

### 9 2.3.1 Otentikasi Pesan (*Message Authentication*)

10 Sebuah pesan inti (*message digest*) dapat digunakan untuk menjamin integritas sebuah  
 11 pesan, memastikan pesan tidak diubah. Namun, pesan inti tidak bisa memastikan keaslian  
 12 pengirim pesan. Ketika Alice mengirim pesan kepada Bob, Bob harus bisa tahu bahwa  
 13 pesan yang dikirim berasal dari Alice. Supaya Bob bisa memastikan bahwa Alice yang  
 14 mengirim pesan, maka Alice harus bisa menyediakan bukti bahwa Alice adalah pengirimnya.  
 15 Contoh dari otentikasi ini antara lain adalah *modification detection code* (MDC) dan *message*  
 16 *authentication code* (MAC).

#### 17 *Modification Detection Code*

18 *Modification detection code* merupakan sebuah pesan inti yang dapat memastikan integritas  
 19 dari sebuah pesan, bahwa pesan tidak diubah saat proses pengiriman. Alice dapat membuat  
 20 pesan inti, yaitu kode deteksi modifikasi dan mengirimkannya bersama dengan pesan asli  
 21 kemudian Bob akan membuat juga secara terpisah kode deteksi modifikasi berdasarkan  
 22 pesan asli yang dikirim Alice dan membandingkan dengan kode deteksi modifikasi yang  
 23 dikirim Alice. Jika sama, maka pesan yang dikirim oleh Alice asli dan tidak berubah saat  
 24 pengiriman. Proses diatas dapat ditunjukkan pada diagram di bawah ini.



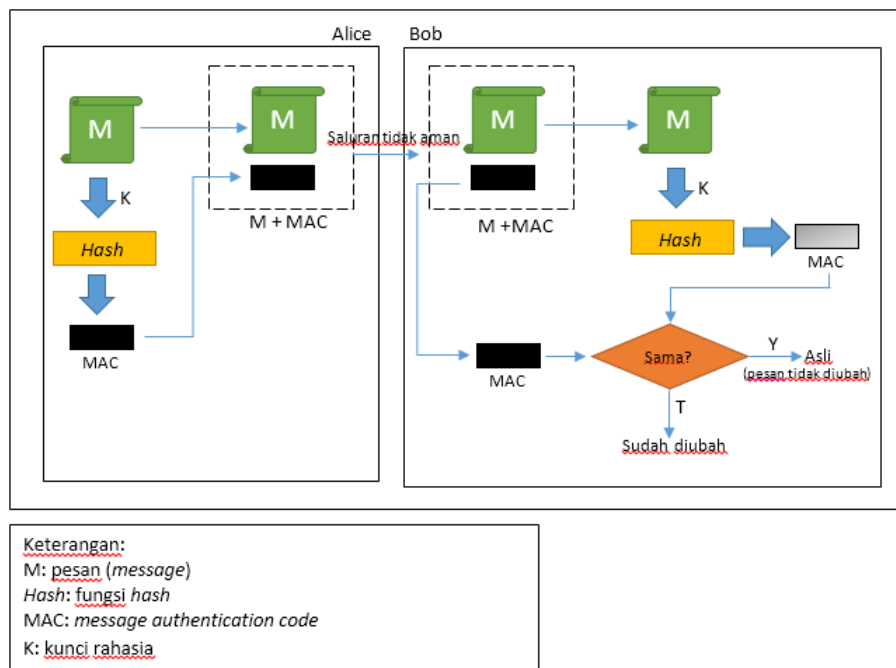
Gambar 2.22: *Modification detection code*

25 Diagram diatas menunjukkan bahwa pesan yang dikirim Alice dapat dikirim melalui  
 26 saluran yang tidak aman, maksudnya pihak lain diluar Alice dan Bob bisa membacanya  
 27 bahkan mengubah isinya, tapi MDC yang dibuat oleh Alice harus dikirimkan pada Bob di

1 saluran aman. Dengan cara ini, MDC tidak akan bisa diubah isinya dan ketika pesan yang  
 2 dikirim melalui saluran tidak aman diubah maka Bob akan tahu bahwa pesan sudah diubah  
 3 oleh pihak lain dengan membandingkan hasil MDC dari pesan yang dia terima dengan MDC  
 4 yang Alice kirimkan melalui saluran aman.

### 5 *Message Authentication Code*

6 Untuk memastikan bahwa memang Alice yang mengirimkan pesan kepada Bob maka kode  
 7 deteksi pesan yang digunakan harus diubah menjadi *message authentication code* (MAC).  
 8 Perbedaan MAC dengan MDC adalah MAC menggunakan sebuah kunci rahasia atau kunci  
 9 pribadi yang hanya diketahui oleh Alice dan Bob saja. Diagram di bawah ini akan menun-  
 10 jukkan cara kerja MAC dalam mengotentikasi pesan.



Gambar 2.23: *Modification authentication code*

11 Alice menggunakan fungsi *hash* untuk membuat MAC dengan menggunakan gabungan  
 12 dari pesan dan kunci rahasia yang hanya diketahui olehnya dan Bob saja. Kemudian, Alice  
 13 mengirimkan pesan beserta dengan MAC pesan tersebut kepada Bob. Bob akan menerima  
 14 pesan dan memisahkan MAC dengan pesan dan kemudian Bob akan membuat MAC dari  
 15 pesan yang diterima dari Alice dengan menggunakan fungsi hash yang sama seperti Alice gu-  
 16 nakan dan tentu saja MAC yang dibuat merupakan gabungan dari pesan dan kunci rahasia.  
 17 Melalui cara ini, jika hasil keluaran MAC yang Bob buat sendiri dibandingkan dengan MAC  
 18 yang Alice kirimkan sama, maka Bob bisa yakin dengan pasti bahwa pesan yang dia terima  
 19 berasal dari Alice dan pesan tersebut asli, tidak diubah oleh pihak lain. Terdapat beberapa  
 20 modifikasi dari *message authentication code* antara lain adalah *nested MAC*, *hashed MAC*,  
 21 dan CMAC.

### 22 2.3.2 Otentikasi Entitas (*Entity Authentication*)

23 Otentikasi entitas merupakan sebuah teknik yang dirancang untuk memastikan kebenaran  
 24 identitas seseorang. Entitas yang dimaksud disini bisa berupa orang, pengguna (*user*), atau  
 25 sebuah *server*. Entitas yang identitasnya perlu dibuktikan kebenarannya disebut penuntut  
 26 (*claimant*) dan entitas yang bertindak untuk memastikan kebenaran identitas dari penuntut  
 27 disebut pemeriksa (*verifier*). Ketika Bob hendak memastikan kebenaran identitas dari Alice,  
 28 Bob adalah pemeriksa dan Alice adalah penuntut.



1 Ada dua perbedaan antara otentikasi pesan dan otentikasi entitas:

- 2 1. Otentikasi pesan atau otentikasi sumber data tidak terjadi secara langsung sedangkan  
3 otentikasi entitas terjadi secara langsung.
- 4 2. Otentikasi pesan hanya mengotentikasi satu pesan saja, ketika pesan berikutnya di-  
5 kirimkan maka proses otentikasi pesan akan dilakukan kembali sedangkan otentikasi  
6 entitas akan terjadi selama satu durasi dalam sebuah sesi.

7 Dalam otentikasi entitas, penuntut harus bisa membuktikan kebenaran identitas dirinya  
8 kepada pemeriksa. Ada beberapa cara pembuktian yang bisa dilakukan oleh penuntut, yaitu:

- 9 • Sesuatu yang diketahui (*something known*). Hal ini diketahui oleh penuntut dan juga  
10 pemeriksa. Contohnya antara lain adalah *password*, nomor PIN, kunci rahasia, dan  
11 kunci pribadi.
- 12 • Sesuatu yang dimiliki (*something possessed*). Hal ini dimiliki oleh penuntut dan bisa  
13 memastikan kebenaran identitas dari penuntut. Contohnya antara lain adalah paspor,  
14 KTP, kartu kredit, dan SIM.
- 15 • Sesuatu yang melekat (*something inherent*). Hal ini menempel atau sebagai bagian  
16 dari penuntut. Contohnya antara lain adalah sidik jari, suara, tanda tangan, pola  
17 retina, dan tulisan tangan.

18 Beberapa teknik dalam otentikasi entitas antara lain, yaitu *password*, *zero-knowledge*, *challenge-*  
19 *response*, dan biometrik.

### 20 2.3.3 Password

21 Salah satu teknik otentikasi entitas yang paling sederhana dan mudah adalah otentikasi  
22 menggunakan *password* atau kata kunci rahasia. *Password* ini adalah sesuatu yang diketahui  
23 hanya oleh penuntut. *Password* digunakan saat penuntut (selanjutnya akan dinamakan  
24 pengguna) perlu untuk mengakses sistem untuk menggunakan sumber daya dari sistem.  
25 Setiap pengguna akan diberikan *username* yang sifatnya publik dan *password* yang sifatnya  
26 rahasia atau pribadi. Ada dua skema otentikasi menggunakan *password* ini, yaitu *password*  
27 tetap (*fixed password*) dan *password* satu-kali (*one-time password*).

#### 28 Password Satu-Kali

29 *Password* satu-kali adalah *password* yang digunakan hanya satu kali. Jadi, setiap kali peng-  
30 guna akan mengakses sistem untuk setiap sesi waktu yang berbeda, maka *password* yang  
31 digunakan akan selalu berbeda-beda. Beberapa teknik yang digunakan dalam *password*  
32 satu-kali antara lain menggunakan *list*, *sequentially updated password*, dan *password* satu-  
33 kali *lamport*.

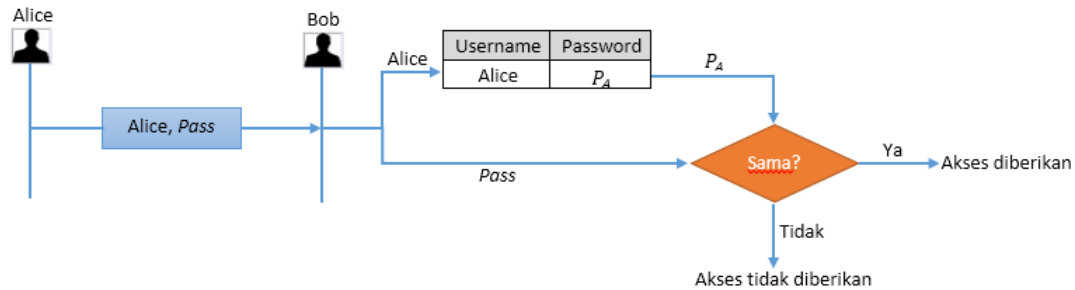
#### 34 Password Tetap

35 *Password* tetap (*fixed password*) adalah *password* yang digunakan berulang-ulang kali setiap  
36 saat pengguna akan mengakses sistem. *Password* ini akan selalu sama untuk setiap kali  
37 pengguna akan mengakses sistem. Ada beberapa skema dari *password* tetap ini.

- 38 • Skema 1

39 Dalam skema ini, sistem menyimpan setiap *password* dalam sebuah tabel basis data  
40 yang diurutkan berdasarkan nama pengguna (*username*). Saat pengguna akan meng-  
41 akses sistem, maka pengguna akan memasukkan *username* dan *password* dalam bentuk  
42 *plaintext*. Kemudian, sistem akan mencari *password* berdasarkan *username*. Jika *pass-*  
43 *word* yang dimasukkan pengguna sesuai dengan *password* yang ada dalam tabel basis

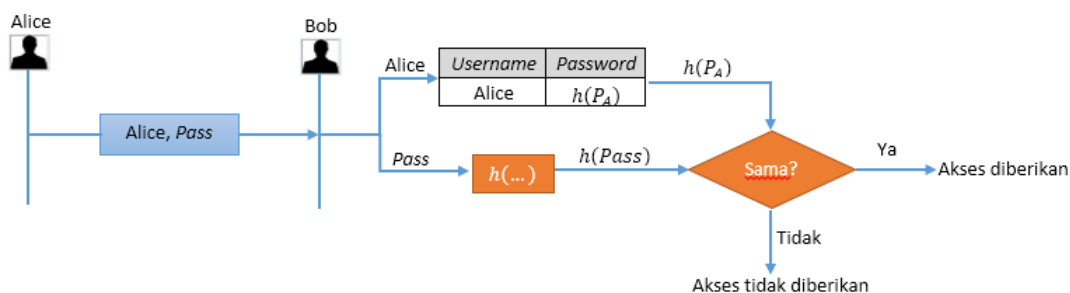
data maka pengguna diberikan akses masuk ke dalam sistem, jika tidak sesuai maka pengguna tidak diberikan akses masuk ke dalam sistem. Kekurangan dari skema ini adalah karena *password* disimpan dalam tabel basis data maka setiap orang yang memiliki akses ke dalam basis data dapat mengetahui setiap *password* yang disimpan dan tentu saja hal ini melanggar salah satu layanan dari kriptografi yaitu, kerahasiaan data (*data confidentiality*).



Gambar 2.24: *Username dan Password*

### • Skema 2

Skema yang lebih aman dibandingkan skema diatas adalah skema penyimpanan *password* yang menggunakan fungsi *hash*. Sistem tidak lagi menyimpan *password* dengan bentuk *plaintext* dalam tabel basis data, namun sistem menyimpan *password* dalam bentuk hashnya dalam tabel basis data, sehingga saat sistem dibobol maka penyerang tidak dapat mengetahui *password* secara langsung karena fungsi *hash* adalah fungsi satu arah dan mustahil untuk mengembalikan ke *plaintext*nya. Selain itu, orang-orang yang memiliki akses ke dalam basis data tidak bisa mengetahui *password* dari setiap *username* yang disimpan dalam tabel. Ketika *password* dibuat, sistem akan menghitung nilai *hash* dari *password* dan menyimpan nilai hash tersebut dalam tabel basis data. Saat pengguna akan mengakses sistem, pengguna memasukkan *username* dan *password* dalam bentuk *plaintext* kemudian sistem akan menghitung nilai hash dari *password* tersebut dan menyesuaikan dengan nilai *hash* dari *password* yang bersangkutan dalam tabel basis data. Jika sesuai, maka pengguna diberikan ijin masuk ke dalam sistem, jika tidak maka pengguna tidak diberikan ijin masuk.

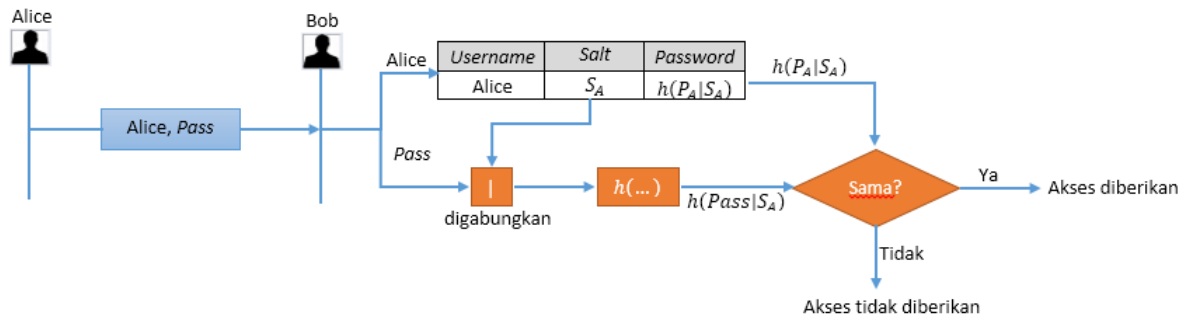


Gambar 2.25: *Password hashing*

### • Skema 3

Skema ketiga menggunakan teknik *salting*. Ketika *password* dibuat, sebuah *string* acak, disebut *salt*, akan ditambahkan pada *password*. Setelah ditambahkan, *password* yang bersangkutan akan dihitung nilai hashnya. Sistem akan menyimpan *username*, *salt*, dan *hash* atau *digest* dari *password* dalam tabel basis data. Ketika pengguna akan mengakses sistem, sistem akan mengambil *salt* berdasarkan *username* yang dimasukkan kemudian menambahkan *salt* tersebut pada *password* yang dimasukkan dan

- 1 menghitung nilai *hash*nya kemudian menyesuaikan dengan yang ada pada tabel basis data. Jika sesuai, maka akses diberikan, jika tidak maka akses akan ditolak.



Gambar 2.26: Password salting

2

## 3 2.4 Secret Sharing

- 4 Enkripsi digunakan untuk menjaga keamanan informasi dan setiap proses enkripsi disertai  
5 oleh sebuah kunci rahasia atau *password*. *Password* ini tentu saja harus selalu diingat oleh  
6 manusia atau disimpan dalam memori komputer. Namun, hal ini tidak sepenuhnya aman  
7 karena bisa saja manusia lupa atau terjadi kerusakan komputer atau bencana alam sehingga  
8 *password* ini hilang dan informasi yang dienkripsi tidak akan bisa diakses atau dikembalikan.

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

- 12 Shamir mendasarkan metode *secret sharing* dalam sebuah masalah sebagai berikut:

- 13 ”*Sebelas orang ilmuwan mengerjakan sebuah proyek rahasia. Mereka menyimpan dokumen*  
14 *dalam sebuah kabinet dimana kabinet ini hanya akan bisa dibuka jika enam atau lebih*  
15 *ilmuwan hadir. Berapa banyak kunci minimal yang dibutuhkan untuk membuka kabinet?*  
16 *Berapa banyak kunci yang harus dimiliki oleh masing-masing ilmuwan?*”

- 17 Tentu saja jawabannya minimal sebanyak 462 kunci untuk membuka kabinet dan mini-  
18 mal 252 kunci yang harus dimiliki oleh setiap ilmuwan. Angka ini sangat tidak masuk akal  
19 dan menyulitkan apalagi jika banyak ilmunannya bertambah. Dalam skema *secret sharing*  
20 shamir, suatu data  $D$  akan dibagi menjadi  $n$  buah *share* dengan ketentuan sebagai berikut:

- 21 • Jika bagian yang ada sebanyak  $k$  bagian atau lebih akan membuat  $D$  mudah untuk  
22 dibentuk kembali.
- 23 • Jika bagian yang ada hanya sebanyak  $k-1$  atau kurang maka  $D$  tidak akan dapat  
24 dibentuk kembali.

- 25 Skema diatas dinamakan skema  $threshold(k,n)$ .

### 26 2.4.1 Skema Threshold( $k,n$ )

- 27 Enkripsi dilakukan untuk melindungi kerahasiaan sebuah data atau informasi, namun ka-  
28 rena setiap proses enkripsi memerlukan kunci maka diperlukan cara lain untuk melindungi  
29 kerahasiaan kunci tersebut. Salah satu cara yang paling aman dalam melindungi kunci  
30 kriptografi adalah dengan menyimpannya dalam sebuah lokasi yang aman dan tersembunyi  
31 (komputer, otak manusia, atau lemari besi).

- 32 Namun cara ini tidak sepenuhnya aman karena bisa saja terjadi bencana alam, kema-  
33 tian, penipuan dan sabotase menyebabkan kunci yang disimpan menjadi rusak atau hilang  
34 sehingga data yang dienkripsi menjadi tidak bisa diakses dan hilang sepenuhnya. Solusi

lainnya adalah dengan memperbanyak kunci yang ada dan menyimpannya di tempat yang berbeda-beda, satu di memori komputer, satu diingat oleh manusia, dan yang lainnya disimpan di tempat yang berbeda-beda. Namun, cara ini juga tidak aman karena masih tidak terhindar dari bencana alam, *human error*, dan malapetaka lainnya.

Untuk mengatasi permasalahan ini maka dibutuhkan sebuah mekanisme atau skema, skema tersebut dinamakan skema *threshold*( $k, n$ ). Skema *threshold*( $k, n$ ) didasarkan pada interpolasi polinomial: diberikan  $k$  titik pada bidang kartesius  $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$  untuk setiap  $x_i$  maka hanya akan ada 1 saja nilai  $q(x_i)$  dalam derajat  $k - 1$  dimana  $q(x_i) = y_i$  untuk seluruh  $x_i$ . Misalkan diasumsikan bahwa  $D$  adalah representasi berupa angka dari kunci yang akan dipecah menjadi beberapa bagian atau *share* menggunakan skema *threshold*( $k, n$ ), kemudian untuk membagi  $D$  menjadi  $D_i$  *share*, dipilih  $k - 1$  derajat polinomial untuk membentuk fungsi  $q(x)$ :

$$q(x) = a_0 + a_1 * x + a_2 * x^2 + \dots + a_{k-1} * x^{k-1}$$

$$a_0 = D$$

Nilai  $a_1$  sampai  $a_{k-1}$  dipilih secara acak kemudian dihitung nilai dari  $D_1$  sampai  $D_n$  dimana  $n$  adalah banyaknya *share*:

$$D_1 = q(1), D_2 = q(2), \dots, D_i = q(i), \dots, D_n = q(n)$$

Untuk setiap nilai dari  $D_n$  yang diketahui, maka bisa dicari nilai koefisien dari fungsi  $q(x)$  dengan interpolasi atau penyelesaian sistem persamaan linear, dari situ bisa diperoleh nilai dari  $q(0)$  sehingga nilai  $D$  nanti bisa diketahui. Namun,  $D$  bisa dihitung jika  $k$  atau lebih *share* diketahui, jika hanya  $k - 1$  atau kurang *share* yang diketahui, maka nilai  $D$  tidak bisa dihitung sehingga data tidak bisa dikembalikan.

Namun, dengan nilai  $k$  yang tinggi juga akan sulit untuk mendapatkan data, karena banyak *share* yang dikumpulkan. Dengan mengurangi nilai  $k$ , akan mempermudah untuk mendapatkan kembali data sehingga mengatasi akibat atau hilangnya beberapa *share*. Pemilihan  $k$  dan  $n$  yang tepat dapat menjaga kerahasiaan data.

## 2.5 Probabilitas

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

Seluruh kemungkinan keluaran yang akan terjadi dalam probabilitas disebut ruang sampel sedangkan masing-masing kemungkinan yang dapat terjadi dalam ruang sampel dinamakan elemen atau anggota dari ruang sampel. Ruang sampel dilambangkan dengan huruf  $S$  dan elemen dilambangkan dengan huruf  $x_i$ . Dalam notasi matematikanya:

$$S = x_1, x_2, x_3, \dots, x_n$$

Sedangkan probabilitas akan kemungkinan bahwa kejadian  $x_i$  akan terjadi dilambangkan dengan  $P(x_i)$ . Dalam rumus matematikanya:

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

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

## 2.6 Distribusi Binom

Setiap eksperimen atau percobaan yang dilakukan secara berkali-kali pasti memiliki dua keluaran, yaitu sukses atau gagal. Untuk setiap keluaran yang diperoleh (baik sukses maupun gagal) bisa ditetapkan sebagai sukses. Proses ini dinamakan proses bernouli dan setiap eksperimen yang dilakukan untuk setiap proses bernouli dinamakan percobaan bernouli. Ada beberapa syarat sebuah eksperimen bisa dinamakan percobaan bernouli:

1. Eksperimen harus diulang sebanyak  $n$  kali.
2. Setiap keluaran dari perulangan bisa dianggap sukses atau gagal.
3. Probabilitas bahwa keluarannya sukses,  $p$ , harus selalu sama untuk setiap kali perulangan.
4. Setiap perulangan tidak dipengaruhi dengan perulangan sebelumnya atau independen.

Maka, untuk banyak perulangan yang sukses,  $X$ , dalam  $n$  perulangan percobaan bernouli dinamakan variabel acak binom sedangkan untuk probabilitas dari variabel acak ini dinamakan distribusi binom dilambangkan dengan  $P(x, n, p)$ , dimana  $p$  merupakan probabilitas keluaran sukses dan  $1 - p$ , yaitu  $q$  merupakan probabilitas keluaran gagal. Karena percobaan dilakukan sebanyak  $n$  kali maka probabilitas keseluruhannya menjadi  $p^n q^{(n-x)}$ , kemudian karena sebanyak  $x$  kejadian sukses yang diambil dari keseluruhan ruang sampel  $n$ , maka probabilitas  $p$  dan  $q$  akan dikalikan dengan kombinasi  $n$  dan  $x$ , dituliskan menjadi  $\binom{n}{x}$ . Jadi, dalam notasi matematikanya:

$$P(x, n, p) = \binom{n}{x} p^x q^{(n-x)} \quad x = 0, 1, 2, \dots, n$$

## 2.7 Entropi

Entropi adalah rata-rata banyak informasi yang dimiliki oleh sebuah pesan. Pesan yang dimaksud disini adalah kejadian yang diharapkan atau elemen dari sebuah ruang sampel atau kejadian. Maka dari itu, entropi bisa dijadikan alat ukur dari tingkat ketidakpastian yang dimiliki oleh sebuah pesan atau sumber informasi. Misalkan sebuah pesan  $X$  terdiri dari huruf-huruf. Untuk setiap huruf  $x$  yang ada dalam pesan  $X$  memiliki banyak kemunculan  $n$  dan  $X$  terdiri dari huruf-huruf sebanyak  $N$ . Kemudian untuk setiap huruf yang bukan  $x$ ,  $y$ , memiliki banyak kemunculan  $m$ . Maka, nilai entropi yang dimiliki oleh pesan  $X$  untuk huruf  $x$ ,  $H(x)$ :

$$H(x) = -\frac{n}{N} \log_2\left(\frac{n}{N}\right) - \frac{m}{N} \log_2\left(\frac{m}{N}\right)$$

atau untuk setiap huruf  $x$  dalam pesan  $X$ , probabilitas kemunculannya adalah  $p$  dan untuk setiap huruf bukan  $x$  dalam pesan  $X$  probabilitas kemunculannya adalah  $q$ , sehingga rumus nilai entropinya menjadi:

$$H(x) = -p \log_2 p - q \log_2 q$$

dimana  $q = 1 - p$  dan menggunakan logaritma basis 2 karena probabilitas kemunculannya biner, yaitu huruf  $x$  dan huruf bukan  $x$ .



## BAB 3

### ANALISIS

Pada bab ini akan dibahas analisis terhadap teori-teori yang telah dibahas sebelumnya. Analisis akan meliputi studi kasus untuk algoritma *secret sharing* shamir yang telah dibahas sebelumnya, algoritma *secret sharing* shamir yang dikembangkan, dan perancangan perangkat lunak.

#### 3.1 Studi Kasus

Bagian ini akan berisi studi kasus mengenai algoritma *secret sharing* shamir yang telah dibahas sebelumnya dan algoritma *secret sharing* shamir yang telah dikembangkan.

##### 3.1.1 *Secret Sharing* Shamir

Pada bagian ini akan dijelaskan proses pembangunan *share* untuk *secret sharing* shamir pada sebuah data  $S$ , proses pembangunan kembali  $S$  dari *share-share* yang ada, dan proses penyelesaian persamaan linear untuk pembangunan kembali  $S$ . Untuk contoh kasus ini, diasumsikan jenis data yang digunakan adalah angka positif, dipilih  $S = 1234$ .

##### Proses pembangunan *share*

Langkah yang perlu dilakukan pertama adalah memilih banyak *share*  $n$  yang diinginkan dan banyak minimal *share* yang diperlukan untuk mengembalikan  $S$ , yaitu  $k$ . Untuk kasus ini, akan dipilih  $n = 8$  dan  $k = 3$ .

Langkah selanjutnya adalah memilih  $k - 1$  angka acak yang nanti akan digunakan sebagai koefisien fungsi polinomial  $f(x)$ . Karena pada kasus ini  $k = 3$  maka ada 2 angka acak yang dipilih, misalkan 237 dan 55. Maka fungsi  $f(x)$  untuk menghitung nilai setiap *share*:

$$f(x) = 1234 + 237x + 55x^2$$

Kemudian, akan dihitung nilai dari setiap  $f(x_i)$  dari  $f(1)$  sampai  $f(8)$  karena  $n = 8$ :

$$\begin{aligned}f(1) &= 1234 + 237 * 1 + 55 * 1 * 1 = 1526 \\f(2) &= 1234 + 237 * 2 + 55 * 2 * 2 = 1928 \\f(3) &= 1234 + 237 * 3 + 55 * 3 * 3 = 2440 \\f(4) &= 1234 + 237 * 4 + 55 * 4 * 4 = 3062 \\f(5) &= 1234 + 237 * 5 + 55 * 5 * 5 = 3794 \\f(6) &= 1234 + 237 * 6 + 55 * 6 * 6 = 4636 \\f(7) &= 1234 + 237 * 7 + 55 * 7 * 7 = 5588 \\f(8) &= 1234 + 237 * 8 + 55 * 8 * 8 = 6650\end{aligned}$$

Maka diperoleh nilai untuk setiap  $S_1$  sampai  $S_8$ .

$$S_1 = 1526, S_2 = 1928, S_3 = 2440, S_4 = 3062, S_5 = 3794, S_6 = 4636, S_7 = 5588, S_8 = 6650$$

### 1 Proses pembangunan kembali (rekonstruksi $S$ )

2 Karena pada kasus ini  $k = 3$  maka untuk mengembalikan  $S$  maka hanya dibutuhkan 3  
3 *share* saja. Misalkan *share* yang dipilih adalah  $S_2$ ,  $S_4$ , dan  $S_5$ .

Langkah selanjutnya adalah membentuk rumus dasar dari fungsi  $f(x)$ . Karena pada kasus ini,  $k = 3$ , maka rumus dasar dari  $f(x)$ :

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

Setelah rumus dasar dari fungsi  $f(x)$  dibentuk, langkah selanjutnya adalah menghitung nilai masing-masing fungsi berdasarkan *share* yang diketahui, dalam kasus ini adalah  $S_2$ ,  $S_4$ , dan  $S_5$ , maka nilai masing-masing fungsi  $f(x)$ :

$$f(2) = c + 2b + 4a = 1928$$

$$f(4) = c + 2b + 16a = 3062$$

$$f(5) = c + 5b + 25a = 3794$$

4 Langkah selanjutnya adalah menyelesaikan persamaan linear di atas, sehingga nanti bi-  
5 sa diperoleh nilai  $c$  dimana seperti yang diketahui nilai  $c$  merupakan  $S$  karena  $S$  adalah  
6 konstanta tanpa koefisien dari  $f(x)$  ( $f(0) = c$ ).

### 7 Proses penyelesaian persamaan linear untuk pembangunan kembali 8 (rekonstruksi $S$ )

Pada bagian ini akan dijelaskan proses penyelesaian persamaan linear menggunakan eliminasi gauss. Pada kasus ini, persamaan linear yang diperoleh:

$$c + 2b + 4a = 1928$$

$$c + 2b + 16a = 3062$$

$$c + 5b + 25a = 3794$$

9 Langkah awal yang perlu dilakukan dalam eliminasi gauss adalah transformasi persamaan  
10 linear ke matriks. Maka, dari persamaan linear di atas matriksnya:

$$\begin{bmatrix} 1 & 2 & 4 & 1928 \\ 1 & 4 & 16 & 3062 \\ 1 & 5 & 25 & 3794 \end{bmatrix}$$

11 Langkah selanjutnya adalah proses yang dinamakan operasi baris, yaitu operasi aritma-  
12 tika (tambah, kurang, kali dan bagi) pada setiap baris dari matriks. Proses ini dilakukan  
13 sampai diperoleh bentuk matriks segitiga atas, yaitu matriks bujur sangkar yang semua  
14 elemen di bawah diagonal utamanya 0. Berikut langkah-langkah proses untuk memperoleh  
15 matriks segitiga atas. Sebelumnya, untuk akan diberi label untuk masing-masing baris, baris  
16 pertama,  $L_1$ , baris kedua,  $L_2$ , dan baris ketiga,  $L_3$ .

17 Langkah pertama dari operasi baris untuk membangun matriks segitiga atas adalah  
18 mengurangi  $L_2$  dan  $L_3$  dengan  $L_1$  agar setiap elemen kolom pertama di bawah  $L_1$  nilainya  
19 menjadi 0 (nol).

$$L_2 - L_1$$

$$L_3 - L_1$$



1 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 1928 \\ 0 & 2 & 12 & 1134 \\ 0 & 3 & 21 & 1866 \end{bmatrix}$$

2 Kemudian, untuk  $L_2$  dan  $L_3$  akan disederhanakan nilainya, dengan cara membagi  $L_2$   
 3 dengan 2 dan  $L_3$  dengan 3.

$$\frac{1}{2}L_2$$

$$\frac{1}{3}L_3$$

4 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 1928 \\ 0 & 1 & 6 & 567 \\ 0 & 1 & 7 & 622 \end{bmatrix}$$

5 Kemudian,  $L_3$  akan dikurangi oleh  $L_2$  sehingga, bisa diperoleh matriks segitiga atas.

$$L_3 - L_2$$

6 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 1928 \\ 0 & 1 & 6 & 567 \\ 0 & 0 & 1 & 55 \end{bmatrix}$$

7 Langkah selanjutnya adalah menghitung nilai konstanta untuk masing-masing koefisien  
 8 dengan cara melakukan substitusi balik. Setiap koefisien pada persamaan diasosiasikan  
 9 dengan kolom pada matriks, kolom pertama untuk  $c$ , kolom kedua untuk  $b$ , dan kolom  
 10 ketiga untuk  $a$ . Maka, dari substitusi balik ini diperoleh:

11 Untuk nilai  $a$  diperoleh dari  $L_3$ .

$$a = 55$$

12 Kemudian, untuk nilai  $b$  diperoleh dari  $L_2$  dengan mensubstitusi nilai  $a$ .

$$b + 6a = 567$$

$$b + 6 * 55 = 567$$

$$b = 567 - 330$$

$$b = 237$$

- 1 Substitusi balik terakhir adalah untuk mendapatkan nilai  $c$  dengan mensubstitusi nilai  $a$   
 2 dan  $b$  pada  $L_1$ , dimana nilai  $c$  adalah  $S$  karena seperti yang diketahui,  $S$  adalah konstanta  
 3 bebas.

$$c + 2b + 4a = 1928$$

$$c + 2 * 237 + 4 * 55 = 1925$$

$$c = 1928 - 474 - 220$$

$$c = 1234$$

### 4 3.1.2 Pengembangan Algoritma *Secret Sharing* Shamir

- 5 Jika pada contoh kasus sebelumnya adalah algoritma *secret sharing* shamir pada jenis data  
 6 angka. Pada bagian ini akan dijelaskan pengembangan dari algoritma *secret sharing* shamir  
 7 untuk jenis data tulisan atau kalimat. Sama seperti contoh kasus sebelumnya, pada bagian  
 8 ini ada 3 proses, yaitu proses pembangunan pembangunan *share* untuk data  $S$ , proses rekon-  
 9 struksi  $S$  dari *share-share*, dan proses penyelesaian persamaan linear untuk mengembalikan  
 10 nilai  $S$ . Untuk contoh kasus ini, data  $S$  bentuknya berupa tulisan atau kalimat.

$$S = \textit{secret}$$

#### 11 Proses pembangunan *share*

- 12 Langkah yang perlu dilakukan pertama adalah memilih banyak *share*  $n$  yang diinginkan  
 13 dan banyak minimal *share* yang diperlukan untuk mengembalikan  $S$ , yaitu  $k$ . Untuk kasus  
 14 ini, akan dipilih  $n = 8$  dan  $k = 4$ .

- 15 Karena  $S$  merupakan tulisan atau kalimat, sebelum menentukan angka acak untuk kon-  
 16 stanta setiap koefisien, setiap karakter dari  $S$  (termasuk spasi) harus diubah menjadi angka  
 17 berdasarkan nilai ASCII. Jadi, untuk masing-masing karakter dalam  $S$ .

$$'s' = 115$$

$$'e' = 101$$

$$'c' = 99$$

$$'r' = 114$$

$$'e' = 101$$

$$'t' = 116$$

- 18 Langkah berikutnya adalah memilih  $k - 1$  angka acak yang nanti akan digunakan sebagai  
 19 koefisien fungsi polinomial  $f(x)$ . Karena  $S$  merupakan tulisan atau kalimat, maka perlu  
 20 dipilih angka acak untuk setiap karakter dari  $S$ . Dalam contoh kasus ini, ada  $k - 1$  angka  
 21 acak untuk masing-masing karakter dari  $S$  dan karena  $k = 4$ , maka untuk masing-masing  
 22 karakter terdapat 3 angka acak.

- 23 •  $'s'$ : 43, 45, dan 27

	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
1	230	267	204	252	303	313
2	597	933	527	868	847	954
3	1378	2423	1224	2322	1925	2303
4	2735	5061	2451	4974	3729	4624
5	4830	9171	4364	9184	6451	8181
6	7825	15077	7119	15312	10283	13238
7	11882	23103	10872	23718	15417	20059
8	17163	33573	15779	34762	22045	28908

Tabel 3.1: *Share dari Data S*

- $'e'$ : 24, 88, dan 54
- $'c'$ : 48, 31, dan 26
- $'r'$ : 19, 59, dan 60
- $'e'$ : 95, 75, dan 32
- $'t'$ : 63, 90, dan 44

Perlu diketahui, pemilihan angka dari masing-masing karakter  $S$  dilakukan secara acak dan tidak memiliki kaitan antara satu karakter dengan karakter lainnya. Langkah selanjutnya adalah membangun fungsi polinomial  $f(x)$  yang akan digunakan untuk menghitung nilai-nilai *share*.

$$f_1(x) = 115 + 43x + 45x^2 + 27x^3 \quad (3.1)$$

$$f_2(x) = 101 + 24x + 88x^2 + 54x^3 \quad (3.2)$$

$$f_3(x) = 99 + 48x + 31x^2 + 26x^3 \quad (3.3)$$

$$f_4(x) = 114 + 19x + 59x^2 + 60x^3 \quad (3.4)$$

$$f_5(x) = 101 + 95x + 75x^2 + 32x^3 \quad (3.5)$$

$$f_6(x) = 116 + 63x + 90x^2 + 44x^3 \quad (3.6)$$

Langkah selanjutnya adalah menghitung nilai setiap *share* untuk masing-masing fungsi. Setiap fungsi  $f(x)$  memiliki 8 *share* karena  $n = 8$ .

Dengan begitu, diperoleh 48 *share* untuk data  $S$ . Selanjutnya diberi label untuk masing-masing *share*:

$$S_{11} = f_1(1), S_{12} = f_1(2), S_{13} = f_1(3), \dots, S_{68} = f_6(8) \quad (3.7)$$

### Proses pembangunan kembali dan penyelesaian persamaan linear (rekonstruksi $S$ )

Untuk mengembalikan  $S$  dari *share-share* yang ada maka proses pembangunan kembali harus dilakukan berulang-ulang untuk setiap karakter dari  $S$ , karena  $S$  adalah data tulisan atau kalimat. Pada contoh kasus ini,  $k$  yang dipilih  $k = 4$  karena itu cukup 4 *share* saja yang diketahui.

Langkah selanjutnya adalah membentuk rumus dasar dari fungsi  $f(x)$ . Karena pada kasus ini,  $k = 4$ , maka rumus dasar dari  $f(x)$ :

$$f(x) = d + cx + bx^2 + ax^3$$

- 1 Selanjutnya, proses pembangunan kembali dimulai dari karakter pertama dari  $S$  sampai  
 2 karakter keenam karena pada kasus ini  $S$  terdiri dari 6 karakter.

### 3 Karakter pertama

- 4 Misalkan *share* yang diketahui untuk karakter pertama adalah  $S_{11}$ ,  $S_{12}$ ,  $S_{14}$ , dan  $S_{16}$ .  
 5 Langkah selanjutnya adalah menghitung nilai dari fungsi  $f(x)$  untuk masing-masing *share*  
 6 yang diketahui.

$$f_1(1) = d + c + b + a = 230 \quad (3.8)$$

$$f_1(2) = d + 2c + 4b + 8a = 597 \quad (3.9)$$

$$f_1(4) = d + 4c + 16b + 64a = 2735 \quad (3.10)$$

$$f_1(6) = d + 6c + 36b + 216a = 7825 \quad (3.11)$$

- 7 Kemudian, untuk proses penyelesaian persamaan linear, setiap fungsi diubah menjadi  
 8 matriks.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 230 \\ 1 & 2 & 4 & 8 & 597 \\ 1 & 4 & 16 & 64 & 2735 \\ 1 & 6 & 36 & 216 & 7825 \end{bmatrix}$$

- 9 Untuk setiap barisnya, akan diberi label  $L_1$ ,  $L_2$ ,  $L_3$ , dan  $L_4$ . Langkah selanjutnya adalah  
 10 operasi baris untuk memperoleh matriks segitiga atas.

$$L_4 - L_1$$

$$L_3 - L_1$$

$$L_2 - L_1$$

- 11 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 230 \\ 0 & 1 & 3 & 7 & 367 \\ 0 & 3 & 15 & 63 & 2505 \\ 0 & 5 & 35 & 215 & 7595 \end{bmatrix}$$

Operasi baris berikutnya.

$$\frac{1}{3}L_3 - \frac{1}{5}L_4$$

- 12 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 230 \\ 0 & 1 & 3 & 7 & 367 \\ 0 & 3 & 15 & 63 & 2505 \\ 0 & 1 & 7 & 43 & 1519 \end{bmatrix}$$

Operasi baris berikutnya.

$$L_4 - L_2$$

$$L_3 - L_2$$

1 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 230 \\ 0 & 1 & 3 & 7 & 367 \\ 0 & 0 & 2 & 14 & 468 \\ 0 & 0 & 4 & 36 & 1152 \end{bmatrix}$$

Operasi baris berikutnya untuk memperoleh matriks segitiga atas.

$$\frac{1}{2}L_2$$

$$L_4 - 2L_2$$

2 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 230 \\ 0 & 1 & 3 & 7 & 367 \\ 0 & 0 & 1 & 7 & 234 \\ 0 & 0 & 0 & 8 & 216 \end{bmatrix}$$

3 Setelah memperoleh matriks segitiga atas, dilakukan proses substitusi balik untuk men-  
4 dapatkan konstanta masing-masing koefisien. Maka, hasil akhir untuk masing-masing kon-  
5 stanta koefisiennya:

$$8a = 216$$

$$a = 27$$

$$b + 7a = 234$$

$$b + 7 * 27 = 234$$

$$b = 234 - 189$$

$$b = 45$$

$$\begin{aligned}
c + 3b + 7a &= 367 \\
c + 3 * 45 + 7 * 27 &= 367 \\
c &= 367 - 135 - 189 \\
c &= 43
\end{aligned}$$

$$\begin{aligned}
d + c + b + a &= 230 \\
d + 43 + 45 + 27 &= 230 \\
d &= 115
\end{aligned}$$

1 Karena  $d$  adalah konstanta bebas pada fungsi  $f(x)$ , maka nilai  $d$  merupakan *secret* untuk  
2 karakter pertama dari data  $S$ . Kemudian, nilai  $d$  ini akan diubah dari nilai ASCII ke  
3 karakter. Maka, karakter pertama adalah 's'.

#### 4 Karakter kedua

5 Misalkan *share* yang diketahui untuk karakter kedua adalah  $S_{23}$ ,  $S_{24}$ ,  $S_{27}$ , dan  $S_{28}$ .  
6 Langkah selanjutnya adalah menghitung nilai dari fungsi  $f(x)$  untuk masing-masing *share*  
7 yang diketahui.

$$f_2(3) = d + 3c + 9b + 27a = 2423 \quad (3.12)$$

$$f_2(4) = d + 4c + 16b + 64a = 5061 \quad (3.13)$$

$$f_2(7) = d + 7c + 49b + 343a = 23103 \quad (3.14)$$

$$f_2(8) = d + 8c + 64b + 512a = 33573 \quad (3.15)$$

8 Kemudian, untuk proses penyelesaian persamaan linear, setiap fungsi diubah menjadi  
9 matriks.

$$\begin{bmatrix}
1 & 3 & 9 & 27 & 2423 \\
1 & 4 & 16 & 64 & 5061 \\
1 & 7 & 49 & 343 & 23103 \\
1 & 8 & 64 & 512 & 33573
\end{bmatrix}$$

10 Untuk setiap barisnya, akan diberi label  $L_1$ ,  $L_2$ ,  $L_3$ , dan  $L_4$ . Langkah selanjutnya adalah  
11 operasi baris untuk memperoleh matriks segitiga atas.

$$\begin{aligned}
L_4 - L_1 \\
L_3 - L_1 \\
L_2 - L_1
\end{aligned}$$

12 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 3 & 9 & 27 & 2423 \\ 0 & 1 & 7 & 37 & 2638 \\ 0 & 4 & 40 & 316 & 20680 \\ 0 & 5 & 55 & 485 & 31150 \end{bmatrix}$$

Operasi baris berikutnya.

$$L_4 - 5L_2$$

$$L_3 - 4L_2$$

1 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 3 & 9 & 27 & 2423 \\ 0 & 1 & 7 & 37 & 2638 \\ 0 & 6 & 12 & 168 & 10128 \\ 0 & 5 & 20 & 300 & 17960 \end{bmatrix}$$

Operasi baris berikutnya untuk memperoleh matriks segitiga atas.

$$L_4 - \frac{20}{12}L_3$$

2 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 3 & 9 & 27 & 2423 \\ 0 & 1 & 3 & 37 & 2638 \\ 0 & 0 & 12 & 168 & 10128 \\ 0 & 0 & 0 & 20 & 1080 \end{bmatrix}$$

3 Setelah memperoleh matriks segitiga atas, dilakukan proses substitusi balik untuk men-  
 4 dapatkan konstanta masing-masing koefisien. Maka, hasil akhir untuk masing-masing kon-  
 5 stanta koefisiennya:

$$20a = 1080$$

$$a = 54$$

$$12b + 168a = 10128$$

$$b + 14a = 844$$

$$b = 844 - 14 * 54$$

$$b = 88$$

$$\begin{aligned}
c + 7b + 37a &= 2638 \\
c + 7 * 88 + 37 * 54 &= 2638 \\
c &= 24
\end{aligned}$$

$$\begin{aligned}
d + 3c + 9b + 27a &= 2423 \\
d + 3 * 24 + 9 * 88 + 27 * 54 &= 2423 \\
d &= 101
\end{aligned}$$

1 Karena  $d$  adalah konstanta bebas pada fungsi  $f(x)$ , maka nilai  $d$  merupakan *secret* untuk  
2 karakter pertama dari data  $S$ . Kemudian, nilai  $d$  ini akan diubah dari nilai ASCII ke  
3 karakter. Maka, karakter pertama adalah ' $e$ '.

#### 4 Karakter ketiga

5 Misalkan *share* yang diketahui untuk karakter ketiga adalah  $S_{32}$ ,  $S_{33}$ ,  $S_{35}$ , dan  $S_{36}$ .  
6 Langkah selanjutnya adalah menghitung nilai dari fungsi  $f(x)$  untuk masing-masing *share*  
7 yang diketahui.

$$f_3(2) = d + 2c + 4b + 8a = 527 \quad (3.16)$$

$$f_3(3) = d + 3c + 9b + 27a = 1224 \quad (3.17)$$

$$f_3(5) = d + 5c + 25b + 125a = 4364 \quad (3.18)$$

$$f_3(6) = d + 6c + 36b + 216a = 7119 \quad (3.19)$$

8 Kemudian, untuk proses penyelesaian persamaan linear, setiap fungsi diubah menjadi  
9 matriks.

$$\begin{bmatrix}
1 & 2 & 4 & 8 & 527 \\
1 & 3 & 9 & 27 & 1224 \\
1 & 5 & 25 & 125 & 4364 \\
1 & 6 & 36 & 216 & 7119
\end{bmatrix}$$

10 Untuk setiap barisnya, akan diberi label  $L_1$ ,  $L_2$ ,  $L_3$ , dan  $L_4$ . Langkah selanjutnya adalah  
11 operasi baris untuk memperoleh matriks segitiga atas.

$$L_4 - L_1$$

$$L_3 - L_1$$

$$L_2 - L_1$$

12 Maka, matriksnya menjadi



$$\begin{bmatrix} 1 & 2 & 4 & 8 & 527 \\ 0 & 1 & 5 & 19 & 697 \\ 0 & 3 & 21 & 117 & 3837 \\ 0 & 4 & 32 & 208 & 6592 \end{bmatrix}$$

Operasi baris berikutnya.

$$L_3 - 3L_2$$

$$L_4 - 4L_2$$

1 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 8 & 527 \\ 0 & 1 & 5 & 19 & 697 \\ 0 & 0 & 6 & 60 & 1746 \\ 0 & 0 & 12 & 132 & 3804 \end{bmatrix}$$

Operasi baris berikutnya.

$$\frac{1}{6}L_3 - \frac{1}{12}L_4$$

2 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 8 & 527 \\ 0 & 1 & 5 & 19 & 697 \\ 0 & 0 & 1 & 10 & 291 \\ 0 & 0 & 2 & 11 & 317 \end{bmatrix}$$

Operasi baris berikutnya untuk memperoleh matriks segitiga atas.

$$L_4 - 2L_3$$

3 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 8 & 527 \\ 0 & 1 & 5 & 19 & 697 \\ 0 & 0 & 1 & 10 & 291 \\ 0 & 0 & 0 & 1 & 26 \end{bmatrix}$$

1 Setelah memperoleh matriks segitiga atas, dilakukan proses substitusi balik untuk men-  
 2 dapatkan konstanta masing-masing koefesien. Maka, hasil akhir untuk masing-masing kon-  
 3 stanta koefesiennya:

$$a = 26$$

$$b + 10a = 291$$

$$b + 10 * 26 = 291$$

$$b = 31$$

$$c + 5b + 19a = 697$$

$$c + 5 * 31 + 19 * 26 = 697$$

$$c = 48$$

$$d + 2c + 4b + 8a = 527$$

$$d + 2 * 48 + 4 * 31 + 8 * 26 = 230$$

$$d = 99$$

4 Karena  $d$  adalah konstanta bebas pada fungsi  $f(x)$ , maka nilai  $d$  merupakan *secret* untuk  
 5 karakter pertama dari data  $S$ . Kemudian, nilai  $d$  ini akan diubah dari nilai ASCII ke  
 6 karakter. Maka, karakter pertama adalah ' $c$ '.

### 7 Karakter keempat

8 Misalkan *share* yang diketahui untuk karakter keempat adalah  $S_{41}$ ,  $S_{44}$ ,  $S_{47}$ , dan  $S_{48}$ .  
 9 Langkah selanjutnya adalah menghitung nilai dari fungsi  $f(x)$  untuk masing-masing *share*  
 10 yang diketahui.

$$f_4(1) = d + c + b + a = 252 \quad (3.20)$$

$$f_4(4) = d + 4c + 16b + 64a = 4974 \quad (3.21)$$

$$f_4(7) = d + 7c + 49b + 343a = 23718 \quad (3.22)$$

$$f_4(8) = d + 8c + 64b + 512a = 34762 \quad (3.23)$$

11 Kemudian, untuk proses penyelesaian persamaan linear, setiap fungsi diubah menjadi  
 12 matriks.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 252 \\ 1 & 4 & 16 & 64 & 4974 \\ 1 & 7 & 49 & 343 & 23718 \\ 1 & 8 & 64 & 512 & 34762 \end{bmatrix}$$

13 Untuk setiap barisnya, akan diberi label  $L_1$ ,  $L_2$ ,  $L_3$ , dan  $L_4$ . Langkah selanjutnya adalah  
 14 operasi baris untuk memperoleh matriks segitiga atas.

$$L_4 - L_1$$

$$L_3 - L_1$$

$$L_2 - L_1$$

1 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 252 \\ 0 & 3 & 15 & 63 & 4722 \\ 0 & 6 & 48 & 342 & 23466 \\ 0 & 7 & 63 & 511 & 34510 \end{bmatrix}$$

Operasi baris berikutnya.

$$\begin{aligned} & \frac{1}{3}L_2 \\ & \frac{1}{6}L_3 - \frac{1}{3}L_2 \\ & \frac{1}{7}L_4 - \frac{1}{3}L_2 \end{aligned}$$

2 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 252 \\ 0 & 1 & 5 & 21 & 1574 \\ 0 & 0 & 3 & 36 & 2337 \\ 0 & 0 & 4 & 52 & 3356 \end{bmatrix}$$

Operasi baris berikutnya untuk memperoleh matriks segitiga atas.

$$\begin{aligned} & L_4 - \frac{4}{3}L_2 \\ & \frac{1}{3}L_3 \end{aligned}$$

3 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 252 \\ 0 & 1 & 5 & 21 & 1574 \\ 0 & 0 & 1 & 12 & 779 \\ 0 & 0 & 0 & 4 & 240 \end{bmatrix}$$

4 Setelah memperoleh matriks segitiga atas, dilakukan proses substitusi balik untuk men-  
5 dapatkan konstanta masing-masing koefisien. Maka, hasil akhir untuk masing-masing kon-  
6 stanta koefesiennya:

$$4a = 240$$

$$a = 60$$

$$b + 12a = 779$$

$$b + 12 * 60 = 779$$

$$b = 59$$

$$c + 5b + 21a = 1574$$

$$c + 5 * 59 + 12 * 60 = 1574$$

$$c = 19$$

$$d + c + b + a = 252$$

$$d + 19 + 59 + 60 = 252$$

$$d = 114$$

1 Karena  $d$  adalah konstanta bebas pada fungsi  $f(x)$ , maka nilai  $d$  merupakan *secret* untuk  
 2 karakter pertama dari data  $S$ . Kemudian, nilai  $d$  ini akan diubah dari nilai ASCII ke  
 3 karakter. Maka, karakter pertama adalah ' $r$ '.

#### 4 Karakter kelima

5 Misalkan *share* yang diketahui untuk karakter kelima adalah  $S_{52}$ ,  $S_{56}$ ,  $S_{57}$ , dan  $S_{58}$ .  
 6 Langkah selanjutnya adalah menghitung nilai dari fungsi  $f(x)$  untuk masing-masing *share*  
 7 yang diketahui.

$$f_5(2) = d + 2c + 4b + 8a = 847 \quad (3.24)$$

$$f_5(6) = d + 6c + 36b + 216a = 10283 \quad (3.25)$$

$$f_5(7) = d + 7c + 49b + 343a = 15417 \quad (3.26)$$

$$f_5(8) = d + 8c + 64b + 512a = 22045 \quad (3.27)$$

8 Kemudian, untuk proses penyelesaian persamaan linear, setiap fungsi diubah menjadi  
 9 matriks.

$$\begin{bmatrix} 1 & 2 & 4 & 8 & 847 \\ 1 & 6 & 36 & 216 & 10283 \\ 1 & 7 & 49 & 343 & 15417 \\ 1 & 8 & 64 & 512 & 22045 \end{bmatrix}$$

10 Untuk setiap barisnya, akan diberi label  $L_1$ ,  $L_2$ ,  $L_3$ , dan  $L_4$ . Langkah selanjutnya adalah  
 11 operasi baris untuk memperoleh matriks segitiga atas.

$$L_4 - L_1$$

$$L_3 - L_1$$

$$L_2 - L_1$$

1 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 8 & 847 \\ 0 & 4 & 32 & 208 & 9436 \\ 0 & 5 & 45 & 335 & 14570 \\ 0 & 6 & 60 & 504 & 21198 \end{bmatrix}$$

Operasi baris berikutnya.

$$\begin{aligned} & \frac{1}{4}L_2 \\ & \frac{1}{5}L_3 - \frac{1}{4}L_2 \\ & \frac{1}{6}L_4 - \frac{1}{4}L_2 \end{aligned}$$

2 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 4 & 8 & 847 \\ 0 & 1 & 8 & 52 & 2359 \\ 0 & 0 & 1 & 15 & 555 \\ 0 & 0 & 2 & 32 & 1174 \end{bmatrix}$$

Operasi baris berikutnya untuk memperoleh matriks segitiga atas.

$$L_4 - 2L_3$$

3 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 2 & 4 & 8 & 847 \\ 0 & 1 & 8 & 52 & 2359 \\ 0 & 0 & 1 & 13 & 555 \\ 0 & 0 & 0 & 2 & 64 \end{bmatrix}$$

4 Setelah memperoleh matriks segitiga atas, dilakukan proses substitusi balik untuk men-  
5 dapatkan konstanta masing-masing koefisien. Maka, hasil akhir untuk masing-masing kon-  
6 stanta koefisiennya:

$$2a = 64$$

$$a = 32$$

$$b + 15a = 555$$

$$b + 15 * 32 = 555$$

$$b = 75$$

$$c + 8b + 52a = 2359$$

$$c + 15 * 32 + 52 * 60 = 2359$$

$$c = 95$$

$$d + 2c + 4b + 8a = 847$$

$$d + 2 * 95 + 4 * 75 + 8 * 32 = 847$$

$$d = 101$$

1 Karena  $d$  adalah konstanta bebas pada fungsi  $f(x)$ , maka nilai  $d$  merupakan *secret* untuk  
 2 karakter pertama dari data  $S$ . Kemudian, nilai  $d$  ini akan diubah dari nilai ASCII ke  
 3 karakter. Maka, karakter pertama adalah 'e'.

#### 4 Karakter keenam

5 Misalkan *share* yang diketahui untuk karakter keenam adalah  $S_{61}$ ,  $S_{63}$ ,  $S_{66}$ , dan  $S_{68}$ .  
 6 Langkah selanjutnya adalah menghitung nilai dari fungsi  $f(x)$  untuk masing-masing *share*  
 7 yang diketahui.

$$f_6(1) = d + c + b + a = 313 \quad (3.28)$$

$$f_6(3) = d + 3c + 9b + 27a = 2303 \quad (3.29)$$

$$f_6(6) = d + 6c + 36b + 216a = 13238 \quad (3.30)$$

$$f_6(8) = d + 8c + 64b + 512a = 28908 \quad (3.31)$$

8 Kemudian, untuk proses penyelesaian persamaan linear, setiap fungsi diubah menjadi  
 9 matriks.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 313 \\ 1 & 3 & 9 & 27 & 2303 \\ 1 & 6 & 36 & 216 & 13238 \\ 1 & 8 & 64 & 512 & 28908 \end{bmatrix}$$

10 Untuk setiap barisnya, akan diberi label  $L_1$ ,  $L_2$ ,  $L_3$ , dan  $L_4$ . Langkah selanjutnya adalah  
 11 operasi baris untuk memperoleh matriks segitiga atas.

$$L_4 - L_1$$

$$L_3 - L_1$$

$$L_2 - L_1$$

1 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 313 \\ 0 & 2 & 8 & 26 & 1990 \\ 0 & 5 & 35 & 215 & 12925 \\ 0 & 7 & 63 & 511 & 28595 \end{bmatrix}$$

Operasi baris berikutnya.

$$\begin{aligned} & \frac{1}{2}L_2 \\ & \frac{1}{5}L_3 - \frac{1}{2}L_2 \\ & \frac{1}{7}L_4 - \frac{1}{2}L_2 \end{aligned}$$

2 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 313 \\ 0 & 1 & 4 & 13 & 995 \\ 0 & 0 & 3 & 30 & 1590 \\ 0 & 0 & 5 & 60 & 3090 \end{bmatrix}$$

Operasi baris berikutnya untuk memperoleh matriks segitiga atas.

$$\begin{aligned} & \frac{1}{3}L_3 \\ & \frac{1}{5}L_4 - \frac{1}{3}L_3 \end{aligned}$$

3 Maka, matriksnya menjadi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 313 \\ 0 & 1 & 4 & 13 & 995 \\ 0 & 0 & 1 & 10 & 530 \\ 0 & 0 & 0 & 2 & 88 \end{bmatrix}$$

4 Setelah memperoleh matriks segitiga atas, dilakukan proses substitusi balik untuk men-  
5 dapatkan konstanta masing-masing koefisien. Maka, hasil akhir untuk masing-masing kon-  
6 stanta koefesiennya:

$$2a = 88$$

$$a = 44$$

$$b + 10a = 530$$

$$b = 530 - 44 * 10$$

$$b = 90$$

$$c + 4b + 13a = 995$$

$$c + 4 * 90 + 13 * 44 = 995$$

$$c = 63$$

$$d + c + b + a = 313$$

$$d + 63 + 90 + 44 = 252$$

$$d = 116$$

1 Karena  $d$  adalah konstanta bebas pada fungsi  $f(x)$ , maka nilai  $d$  merupakan *secret* untuk  
 2 karakter pertama dari data  $S$ . Kemudian, nilai  $d$  ini akan diubah dari nilai ASCII ke  
 3 karakter. Maka, karakter pertama adalah ' $t$ '.

4 Hasil akhir pembangunan kembali  $S$  diperoleh dengan menyatukan keenam karakter yang  
 5 sudah dibangun kembali dari *share-share* yang diketahui, yaitu menjadi '*secret*', dimana  
 6 '*secret*' merupakan data awal dari  $S$ .

$$S = secret$$

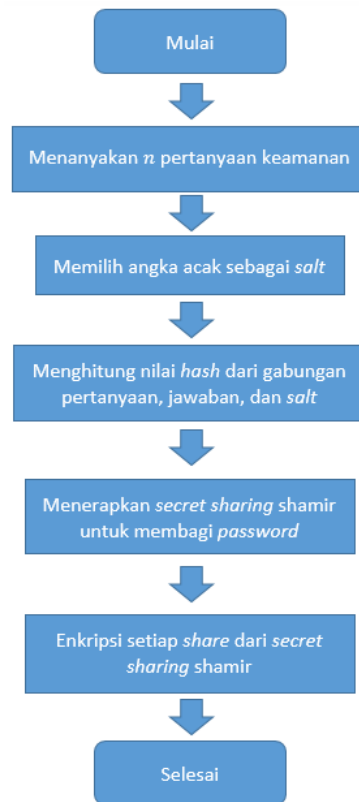
## 7 3.2 Perancangan Perangkat Lunak

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

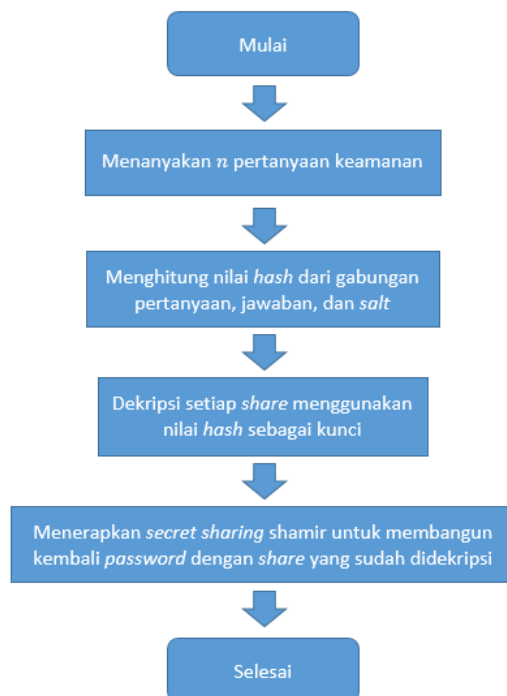
### 10 3.2.1 Alur Proses

11 Algoritma *secret sharing* shamir yang sudah dijelaskan pada bagian sebelumnya akan di-  
 12 terapkan sebagai perlindungan *password* dengan menggunakan pertanyaan keamanan yang  
 13 sifatnya personal. Proses ini akan dibagi menjadi 2 bagian, yaitu proses pembangunan *share*  
 14 dari pesan rahasia dan proses pembangunan kembali atau rekonstruksi pesan rahasia da-  
 15 ri *share-share* yang ada. Dalam alur proses ini diasumsikan bahwa  $n$  dan  $k$  sudah dipilih  
 16 dengan baik dan optimal dan pesan rahasia disini adalah *password*. Berikut alur proses  
 17 pembangunan *share* dari *password*.



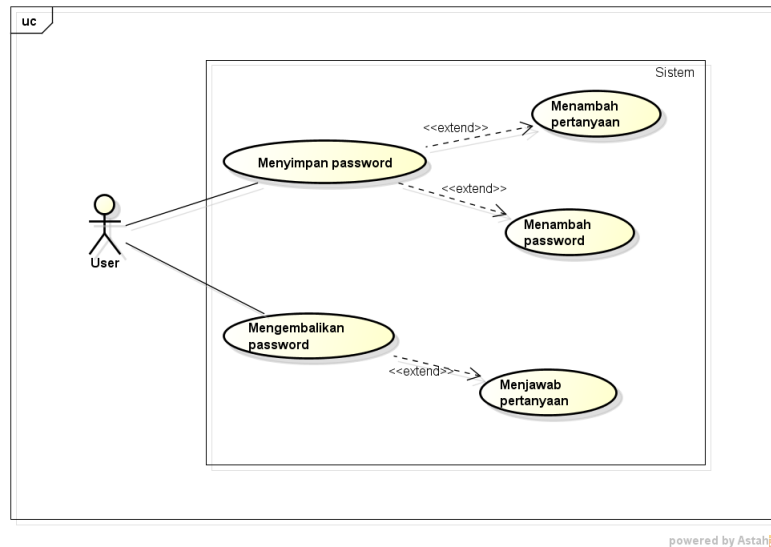
Gambar 3.1: Proses pembangunan *share* dari *password*

- 1 Kemudian, untuk alur proses pembangunan kembali atau rekonstruksi *password* dari
- 2 *share-share* yang adalah sebagai berikut.

Gambar 3.2: Proses pembangunan kembali atau rekonstruksi *password*

### 3.2.2 Diagram Use Case

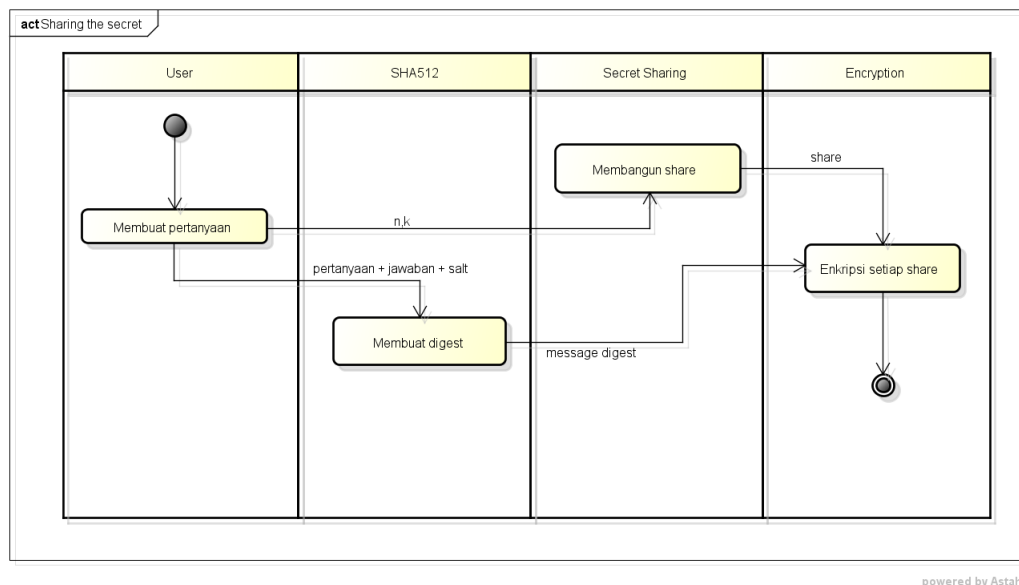
Perangkat lunak yang dibangun akan memiliki 2 fitur utama, yaitu menyimpan *password* beserta pertanyaan keamanan yang sifatnya personal dan mengembalikan *password*. Saat menyimpan *password*, pengguna akan diminta untuk menambahkan pertanyaan keamanan yang sifatnya personal dan saat mengembalikan *password*, pengguna akan diminta untuk menjawab pertanyaan keamanan yang sudah disimpan saat menyimpan *password*. Diagram *use case* di bawah ini menunjukkan kedua fitur utama.



Gambar 3.3: Diagram *use case* dari perangkat lunak

### 3.2.3 Diagram Aktivitas

Perangkat lunak yang dibangun memiliki 2 proses, yaitu menyimpan *password* atau *secret* dan mengembalikan *password* atau *secret*. Diagram aktivitas di bawah ini menunjukkan proses menyimpan *password* atau *secret*.



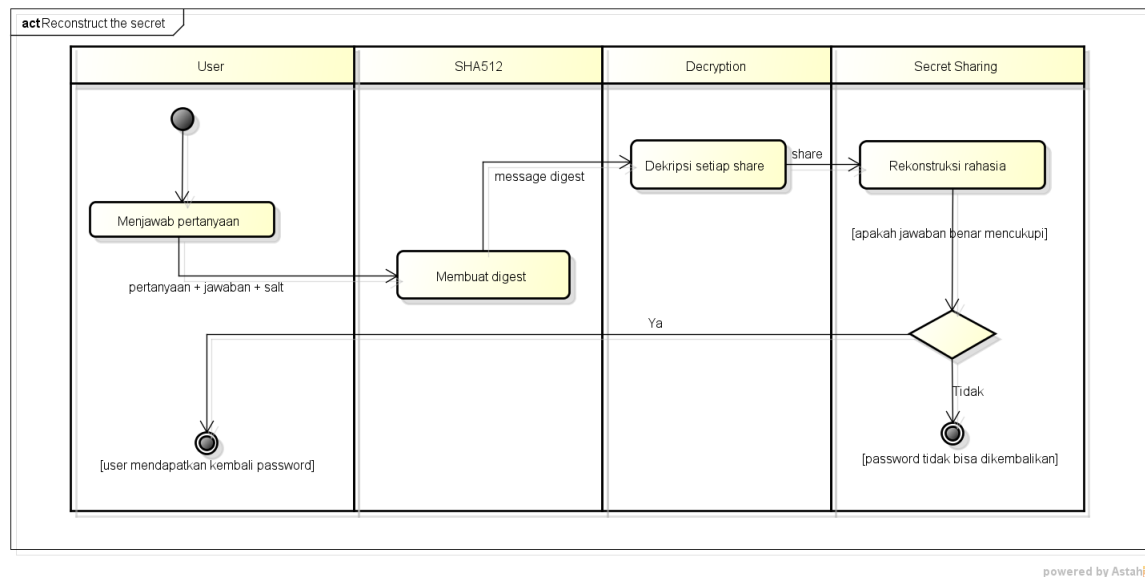
Gambar 3.4: Diagram aktivitas untuk menyimpan *password*

Dalam proses menyimpan *password*, awalnya *user* harus terlebih dahulu menentukan banyak pertanyaan keamanan yang hendak digunakan (*n*) dan banyak minimal pertanyaan-

an 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 proses mengembalikan *password*.



Gambar 3.5: Diagram aktivitas untuk mengembalikan *password*

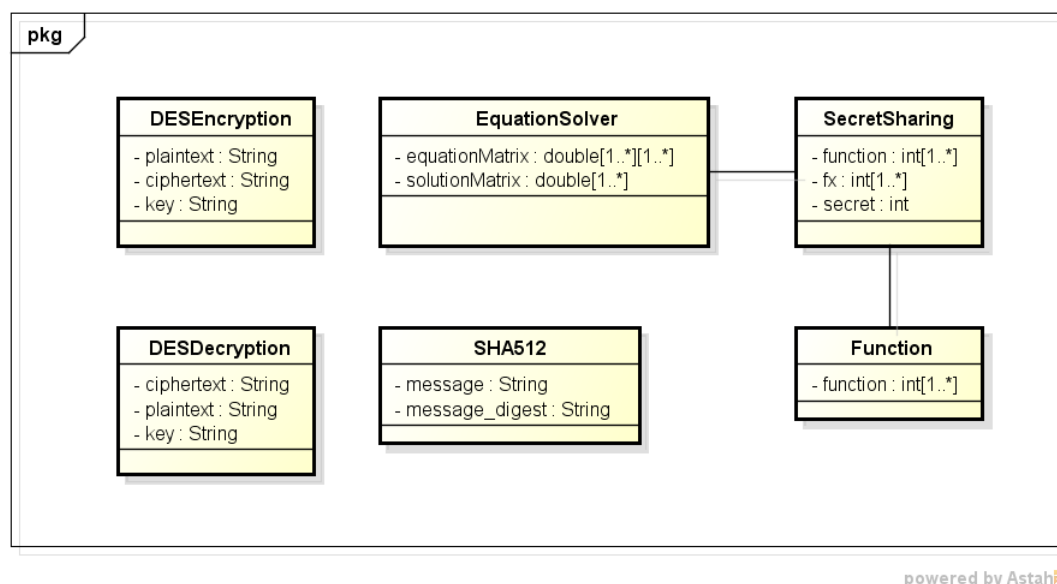
Dalam proses untuk mengembalikan *password*, *user* akan diminta untuk menjawab beberapa pertanyaan keamanan yang sudah dipilih saat *user* menyimpan *password*. Selanjutnya adalah proses yang sama saat menyimpan *password*, yaitu menghitung nilai *hash* dari pertanyaan keamanan yang sudah dijawab oleh *user*. Langkah selanjutnya adalah mendekripsi setiap *share* dengan menggunakan kunci nilai *hash*.

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

### 3.2.4 Diagram Kelas

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

Bagian *engine* merupakan sekumpulan kelas *Java*, sedangkan bagian antarmuka akan terdiri dari sekumpulan *Java Server Page* atau JSP. Pada bagian ini akan dijelaskan bagian *engine* saja.

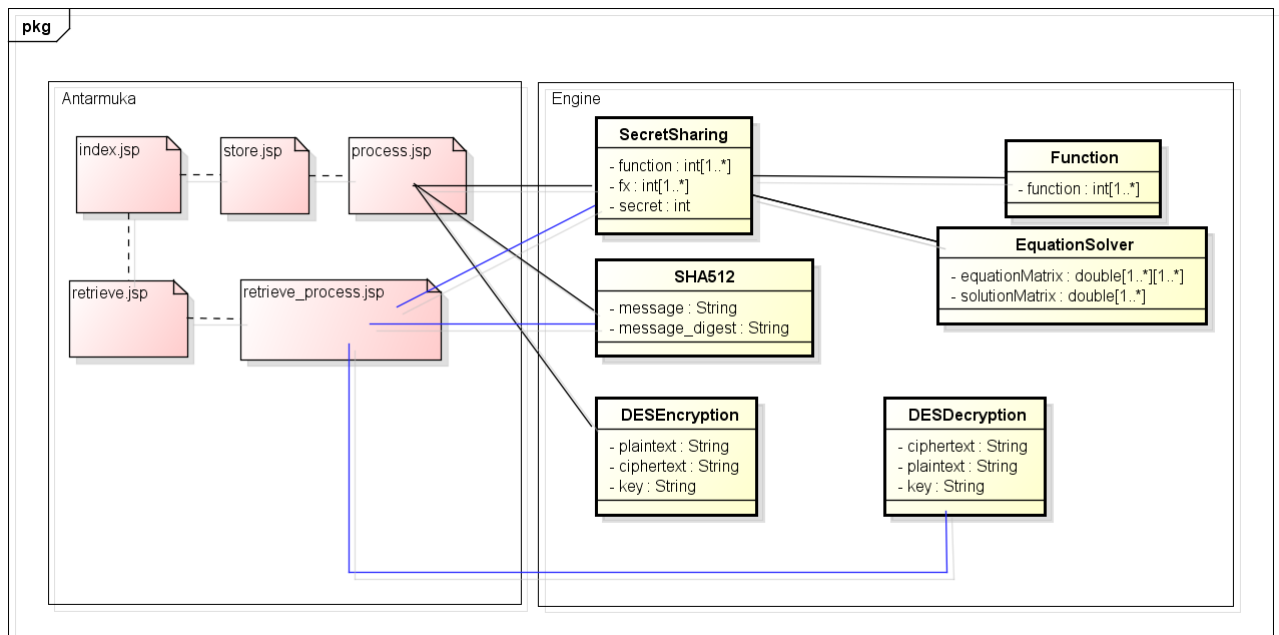
Gambar 3.6: Diagram kelas *engine*

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 3.2.5 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.



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.



## BAB 4

### PERANCANGAN

Pada bab ini akan dibahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak akan mencakup algoritma, perancangan antarmuka, diagram kelas lengkap, dan perancangan berorientasi objek.

#### 4.1 Algoritma

Pada bagian ini akan berisi mengenai algoritma yang digunakan oleh perangkat lunak dalam menyimpan *password* (*sharing the secret*) and mengembalikan *password* (*reconstructing the secret*). Diasumsikan bahwa sekumpulan pertanyaan keamanan yang logis dengan jawabannya serta  $n$  (banyak pertanyaan keamanan) dan  $k$  (banyak minimal jawaban benar yang harus dijawab untuk mengembalikan *password*) sudah ditentukan.

##### Algoritma untuk menyimpan *password* $p$

- (1) Membuat pertanyaan  $q_1, q_2, \dots, q_n$ .
- (2) Menjawab setiap pertanyaan  $q_1, q_2, \dots, q_n$  untuk menghasilkan jawaban  $a_1, a_2, \dots, a_n$ .
- (3) Menghitung nilai hash *dari* gabungan pertanyaan, jawaban, dan angka acak dinamakan *salt*:  $h_1 = H(q_1 + a_1 + r_s), \dots, h_n = H(q_n + a_n + r_s)$ .
- (4) Membagi  $p$  menjadi beberapa karakter, kemudian setiap karakter akan diubah menjadi nilai ASCII:  $c_1, c_2, \dots, c_m$ .
- (5) Untuk setiap karakter  $c_1, c_2, \dots, c_m$ , dengan menggunakan skema *threshold*( $k, n$ ) membagi setiap karakter  $c_1, c_2, \dots, c_m$  menjadi  $n$  share  $s_{11}, s_{12}, \dots, s_{mn}$ .
- (6) Mengenkripsi setiap *share*  $s$  dengan nilai *hash* sebagai kunci:  $E_{h_1}(s_{11}) = c_{11}, E_{h_2}(s_{12}) = c_{12}, E_{h_1}(s_{21}) = c_{21}, \dots, E_{h_n}(s_{mn}) = c_{mn}$ .

Nilai *salt*  $r_s$  sebenarnya tidak terlalu dibutuhkan untuk keamanan. Karena walaupun ada kemungkinan bahwa untuk 2 atau lebih kasus nilai *hash*  $h_i$  bisa sama tetapi  $s_{mn}$  pasti akan berbeda sehingga  $c_{mn}$  pasti akan berbeda juga. Tapi, untuk lebih lagi menjamin keamanan maka nilai *salt*  $r_s$  tetap akan digunakan.

##### Algoritma untuk mengembalikan *password* $p$

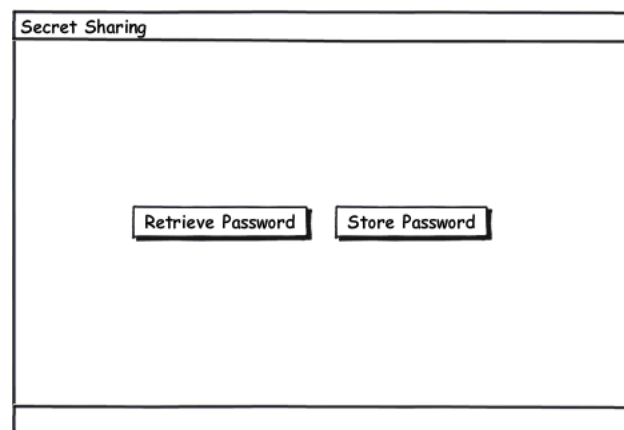
- (1) Menjawab setiap pertanyaan  $q_1, q_2, \dots, q_n$  yang sama untuk menghasilkan jawaban  $a'_1, a'_2, \dots, a'_n$ .
- (2) Menghitung nilai hash *dari* gabungan pertanyaan, jawaban, dan *salt*:  $h'_1 = H(q_1 + a'_1 + r_s), \dots, h'_n = H(q_n + a'_n + r_s)$ .
- (3) Mendekripsi setiap ciphertext  $c_{11}, c_{12}, \dots, c_{mn}$  dengan nilai hash sebagai kunci:  $D_{h'_1}(c_{11}) = s'_{11}, D_{h'_2}(c_{12}) = s'_{12}, D_{h'_1}(c_{21}) = s'_{21}, \dots, D_{h'_n}(c_{mn}) = s'_{mn}$ .

- (4) Dengan menggunakan skema  $threshold(k, n)$ , untuk setiap share  $s$ , jika pertanyaan yang dijawab benar banyaknya sesuai atau lebih dari  $k$ , maka  $p$  bisa direkonstruksi, jika hanya  $k - 1$  atau kurang pertanyaan yang dijawab benar, maka  $p$  tidak bisa direkonstruksi.

## 4.2 Perancangan Antarmuka

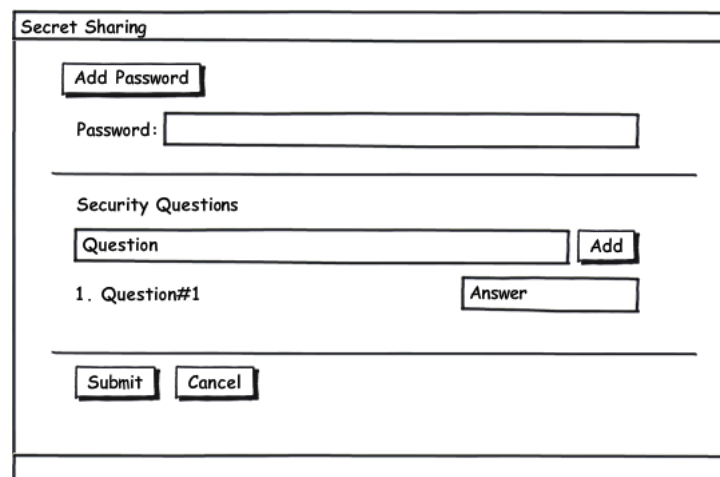
Perangkat lunak yang dikembangkan akan memiliki 3 tampilan utama, tampilan untuk menyimpan *password*, tampilan untuk mengembalikan *password*, dan tampilan untuk memilih menyimpan *password* atau mengembalikan *password*.

Gambar 4.1 menunjukkan tampilan awal yang akan dimunculkan pertama kali untuk memilih menyimpan *password* atau mengembalikan *password*.



Gambar 4.1: Perancangan Tampilan Awal

Tampilan utama ini cukup sederhana. Dalam tampilan utama pada Gambar 4.1, hanya terdapat 2 pilihan, yaitu *store password* untuk menyimpan *password* dan *retrieve password* untuk mengembalikan *password*. Selanjutnya, jika pengguna memilih *store password*, maka akan ditampilkan halaman *store password*.



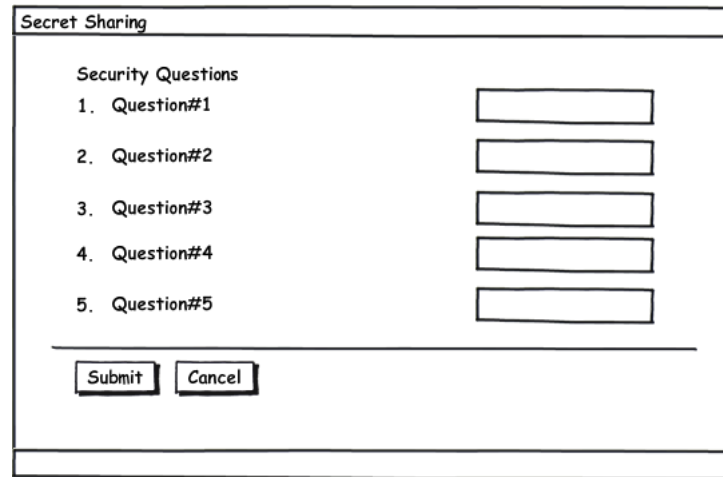
Gambar 4.2: Perancangan Tampilan Menyimpan *Password*

Pada tampilan menyimpan *password* di Gambar 4.2, tombol "Add Password" berfungsi untuk menambah *text box password*, pada bagian ini pengguna bisa mengisi *password* yang akan disimpan. Bagian "Security Questions" berisi pertanyaan keamanan yang dibuat oleh pengguna. Setelah pengguna mengisi pertanyaan personal pada *text box* di bagian "Security Questions" dan menekan tombol "Add", akan muncul pertanyaan yang sudah dibuat, kemudian pengguna harus mengisi jawaban dari pertanyaan keamanan yang sudah dibuat.



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

5 Berikutnya adalah tampilan untuk mengembalikan *password*. Gambar 4.3 menunjukkan  
6 tampilan untuk mengembalikan *password*.

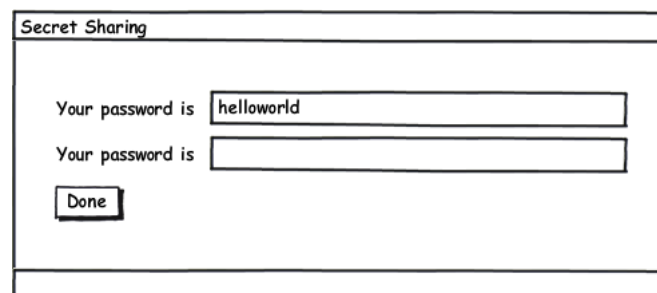


The image shows a window titled "Secret Sharing". Inside, there is a section labeled "Security Questions". Below this label, there are five numbered questions: "1. Question#1", "2. Question#2", "3. Question#3", "4. Question#4", and "5. Question#5". To the right of each question is an empty rectangular input field. At the bottom of the window, there are two buttons: "Submit" and "Cancel".

Gambar 4.3: Perancangan Tampilan Mengembalikan *Password*

7 Pada bagian untuk mengembalikan *password*, tampilannya cukup sederhana dan penggu-  
8 na hanya cukup memasukkan setiap jawaban dari pertanyaan keamanan yang sudah dibuat  
9 sebelumnya di bagian penyimpanan *password*. Pada bagian ini, pengguna bebas untuk  
10 memilih mengisi setiap pertanyaan atau tidak menjawab pertanyaan keamanan. Setelah se-  
11 luruh pertanyaan sudah dijawab, pengguna dapat menekan tombol "*Submit*" yang kemudian  
12 akan menunjukkan *password* pengguna.

13 Gambar 4.4 menunjukkan tampilan sesudah pengguna menekan tombol "*Submit*" pada  
14 bagian di Gambar 4.3.



The image shows the same "Secret Sharing" window after the user has submitted answers. The "Security Questions" section is no longer visible. Instead, there are two lines of text: "Your password is" followed by a text box containing the word "helloworld". Below this, there is another "Your password is" followed by an empty text box. At the bottom left, there is a button labeled "Done".

Gambar 4.4: Perancangan Tampilan Mengembalikan *Password*

15 Jika banyak pertanyaan keamanan yang dijawab benar oleh pengguna sesuai dengan  
16 minimal banyak pertanyaan keamanan yang dijawab benar maka pengguna bisa melihat  
17 *password* yang sudah disimpan. Tapi, jika banyak pertanyaan keamanan yang dijawab benar  
18 oleh pengguna kurang dari minimal banyak pertanyaan keamanan yang harus dijawab benar  
19 maka pengguna tidak bisa melihat *password* yang sudah disimpan.

## 20 4.3 Diagram Kelas Rinci

21 Diagram kelas rinci digunakan sebagai gambaran umum untuk setiap kelas yang ada dalam  
22 perangkat lunak yang dibangun serta keterkaitan setiap kelas. Diagram kelas rinci dapat  
23 dilihat pada Gambar 4.5. Ada perbedaan antara diagram kelas pada Gambar 4.5 dengan

- 1 kelas diagram pada Bab 3. Pada diagram kelas rinci ditambahkan beberapa atribut dan
- 2 fungsi sesuai dengan kebutuhan dari masing-masing kelas.

## 3 4.4 Deskripsi Kelas dan Fungsi

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

### 6 4.4.1 SHA512

7 Kelas SHA512 merupakan kelas yang mengimplementasikan *Secure hashing algorithm 512*  
 8 (SHA-512). Cara kerja algoritma dapat dilihat pada bagian 2.2.1. Kelas ini memiliki 8  
 9 atribut, yaitu *INITIALS*, *CONSTANTS*, *originalMessage*, *lengthMessage*, *paddingMessage*,  
 10 *message*, *messageBlock*, dan *digest*. Atribut *INITIALS* menyimpan 8 konstanta awal  
 11 SHA512, dapat dilihat pada gambar 2.18. Atribut *CONSTANTS* menyimpan 80 konstanta  
 12 yang digunakan untuk 80 ronde SHA512.

13 Atribut *originalMessage* menyimpan *message* dalam bentuk *string*. Atribut *lengthMes-*  
 14 *sage* menyimpan informasi dari panjang atribut *originalMessage* dalam bentuk biner. Atri-  
 15 but *paddingMessage* menyimpan *padding* dari *originalMessage* dalam bentuk biner. Atri-  
 16 but *message* menyimpan gabungan dari atribut *originalMessage*, *lengthMessage*, dan *pa-*  
 17 *ddingMessage* dalam bentuk biner. Atribut *messageBlock* menyimpan *array* yang berisi  
 18 *message*, dibagi per 1024-bit. Atribut *digest* menyimpan nilai *hash* dari atribut *message*.

19 Kelas SHA512 memiliki 15 fungsi, yaitu fungsi *setMessage*, fungsi *reset*, fungsi *getDigest*,  
 20 fungsi *createDigest*, fungsi *wordExpansion*, fungsi *round*, fungsi *rotShift*, fungsi *rotR*, fungsi  
 21 *shL*, fungsi *majority*, fungsi *conditional*, fungsi *rotate*, fungsi *longValue*, fungsi *initialize*, dan  
 22 fungsi *pad*.

23 Fungsi *setMessage* yang berguna untuk mengatur isi dari atribut *message*. Fungsi *reset*  
 24 berguna untuk mengatur ulang seluruh nilai atribut kecuali untuk atribut *INITIALS* dan  
 25 *CONSTANTS*. Fungsi *initialize* berguna untuk melakukan inisialisasi awal sebelum meng-  
 26 hitung nilai hash, cara kerja fungsi ini bisa dilihat pada bagian Persiapan *Message* dalam  
 27 2.2.1.

28 Fungsi *getDigest* berguna untuk mengembalikan nilai atribut *digest*. Fungsi *rotShift*,  
 29 *rotR*, *shL*, *majority*, *conditional*, dan *rotate* merupakan fungsi dasar dari SHA512 yang  
 30 mengoperasikan *bit-bit*, rumus fungsi ini dapat dilihat pada bagian Ekspansi *Word* dalam  
 31 2.2.1 dan Ronde dalam 2.2.1. Fungsi *round* merupakan fungsi yang mengimplementasikan  
 32 satu ronde dalam SHA512, satu ronde dalam SHA512 dapat dilihat pada bagian Ronde  
 33 dalam 2.2.1.

34 Fungsi *wordsExpansion* merupakan fungsi ekspansi *word* dalam SHA512, cara kerja fung-  
 35 si ini dapat dilihat pada bagian Ekspansi *Word* dalam 2.2.1. Fungsi *createDigest* adalah  
 36 fungsi utama dari kelas SHA512 ini, fungsi ini akan menjalankan 80 ronde dari SHA512 dan  
 37 menghasilkan nilai *hash* dari atribut *message*. Fungsi *longValue* berguna untuk mengubah  
 38 nilai *string* menjadi *long*. Fungsi *pad* berguna menambahkan *padding* pada *string* supaya  
 39 bisa diubah menjadi nilai *long* dengan fungsi *longValue*.

### 40 4.4.2 Function

41 Kelas *Function* merupakan kelas yang merepresentasikan sebuah fungsi polinomial  $f(x)$ .  
 42 Kelas ini memiliki 1 atribut, yaitu atribut *function* yang menyimpan konstanta setiap koe-  
 43 fisien dari fungsi polinomial  $f(x)$ . Kelas ini memiliki 1 fungsi, yaitu fungsi *countFunction*.  
 44 Fungsi *countFunction* berguna untuk menghitung nilai dari titik  $x$  dalam fungsi polinomial  
 45  $f(x)$ .

#### 4.4.3 *EquationSolver*

Kelas *EquationSolver* merupakan kelas yang berguna untuk menyelesaikan persamaan linear. Kelas ini memiliki 2 atribut, yaitu *equationMatrix* dan *solutionMatrix*. Kelas *EquationSolver* memiliki 2 fungsi, yaitu fungsi *reset*, dan fungsi *solve*. Atribut *equationMatrix* menyimpan nilai konstanta setiap koefisien dari matriks persamaan linear. Atribut *solutionMatrix* menyimpan matriks solusi dari persamaan linear. Fungsi *reset* berguna untuk mengembalikan seluruh nilai atribut ke nilai awal. Fungsi *solve* adalah implementasi dari algoritma eliminasi Gauss-Jordan untuk menyelesaikan persamaan linear.

#### 4.4.4 *SecretSharing*

Kelas *SecretSharing* merupakan kelas yang berperan untuk menjalankan algoritma *secret sharing* shamir. Kelas ini memiliki 4 atribut, yaitu *function*, *fx*, *secret*, dan *solver*. Atribut *function* menyimpan konstanta koefisien dari fungsi polinomial  $f(x)$ . Atribut *fx* menyimpan setiap nilai  $x$  dari fungsi polinomial  $f(x)$ . Atribut *secret* menyimpan nilai *secret* dari *share-share* yang akan dibangun. Atribut *solver* menyimpan objek dari kelas *EquationSolver*.

Kelas *SecretSharing* memiliki 4 fungsi, yaitu fungsi *getFunction*, fungsi *getFx*, fungsi *split*, dan fungsi *reconstruct*. Fungsi *getFunction* berguna untuk mengembalikan nilai dari atribut *function*. Fungsi *getFx* berguna untuk mengembalikan nilai atribut *fx*. Fungsi *split* berguna untuk membangun *share-share*. Fungsi *reconstruct* berguna untuk mengembalikan nilai *secret* dari *share-share* yang dimiliki.

#### 4.4.5 *DESEncryption*

Kelas *DESEncryption* merupakan kelas yang mengimplementasikan proses enkripsi algoritma *data encryption standard* (DES). Kelas *DESEncryption* memiliki 22 atribut, yaitu *PC1*, *PC2*, *IP*, *exp*, *s1*, *s2*, *s3*, *s4*, *s5*, *s6*, *s7*, *s8*, *P*, *IP1*, *strMsg*, *strKey*, *msg*, *key*, *roundKey*, *cipher*, *sBox*, dan *msgBlock*. Atribut *PC1* dan *PC2* menyimpan matriks permutasi untuk pembangunan kunci ronde. Atribut *IP* menyimpan matriks inisialisasi awal. Atribut *exp* menyimpan matriks ekspansi *P-box*, penjelasan mengenai *P-box* dapat dilihat pada bagian Kompresi *P-box* dalam 2.1.2. Atribut *s1*, *s2*, *s3*, *s4*, *s5*, *s6*, *s7*, dan *s8* menyimpan matriks *S-box*. Atribut *P* menyimpan matriks permutasi untuk setiap ronde dalam DES. Atribut *IP1* menyimpan matriks permutasi akhir.

Atribut *strMsg* menyimpan *plaintext* yang akan dienkripsi. Atribut *strKey* menyimpan kunci rahasia untuk enkripsi. Atribut *msg* menyimpan *plaintext* dalam bentuk biner. Atribut *key* menyimpan kunci rahasia dalam bentuk biner. Atribut *roundKey* menyimpan kunci untuk setiap ronde. Atribut *cipher* menyimpan *ciphertext* hasil enkripsi. Atribut *sBox* menyimpan setiap matriks *S-box*. Atribut *msgBlock* menyimpan *plaintext* dalam bentuk biner yang sudah dibagi menjadi beberapa blok biner berukuran 64-bit.

Kelas *DESEncryption* memiliki 13 fungsi, yaitu fungsi *setMessage*, fungsi *setKey*, fungsi *getCipherText*, fungsi *reset*, fungsi *permute*, fungsi *xor*, fungsi *leftShift*, fungsi *initialize*, fungsi *initialPermutation*, fungsi *createSubKey*, fungsi *feistelCipher*, fungsi *round*, dan fungsi *encrypt*. Fungsi *setMessage* untuk mengatur nilai atribut *strMsg*. Fungsi *setKey* untuk mengatur nilai atribut *strKey*. Fungsi *getCipherText* untuk mengembalikan nilai dari atribut *cipher*. Fungsi *reset* untuk mengatur ulang nilai dari atribut *strMsg*, *strKey*, *msg*, *key*, *roundKey*, *cipher*, dan *msgBlock*. Fungsi *permute* merupakan fungsi dasar dalam DES, penjelasan mengenai fungsi ini dapat dilihat pada Gambar 2.10.

Fungsi *xor* berguna untuk melakukan operasi *bit xor*. Fungsi *leftShift* berguna untuk menggeser *bit* ke arah kiri sebanyak  $n$ -bit. Fungsi *initialize* berguna untuk memroses *plaintext* dan membangun kunci ronde sebelum dilakukan proses enkripsi. Fungsi *initialPermutation* berguna untuk melakukan permutasi awal. Fungsi *createSubKey* berguna untuk membangun kunci untuk setiap ronde. Fungsi *feistelCipher* merupakan implementasi dari sandi feistel. Fungsi *round* adalah menjalankan satu ronde dari DES. Fungsi *encrypt* merupakan fungsi utama untuk enkripsi menggunakan DES.

#### 4.4.6 *DESDecryption*

Kelas *DESDecryption* merupakan kelas yang mengimplementasikan proses dekripsi algoritma *data encryption standard* (DES). Kelas *DESEncryption* memiliki 18 atribut, yaitu *PC1*, *PC2*, *IP*, *exp*, *s1*, *s2*, *s3*, *s4*, *s5*, *s6*, *s7*, *s8*, *P*, *IP1*, *key*, *roundKey*, *cipher*, *sBox*, dan *msgBlock*. Atribut *PC1* dan *PC2* menyimpan matriks permutasi untuk pembangunan kunci ronde. Atribut *IP* menyimpan matriks inisialisasi awal. Atribut *exp* menyimpan matriks ekspansi *P-box*, penjelasan mengenai *P-box* dapat dilihat pada bagian Kompresi *P-box* dalam 2.1.2. Atribut *s1*, *s2*, *s3*, *s4*, *s5*, *s6*, *s7*, dan *s8* menyimpan matriks *S-box*. Atribut *P* menyimpan matriks permutasi untuk setiap ronde dalam DES. Atribut *IP1* menyimpan matriks permutasi akhir.

Atribut *key* menyimpan kunci rahasia untuk dekripsi dalam bentuk biner. Atribut *roundKey* menyimpan kunci untuk setiap ronde. Atribut *cipher* menyimpan *ciphertext* yang akan didekripsi. Atribut *sBox* menyimpan setiap matriks *S-box*. Kelas *DESEncryption* memiliki 12 fungsi, yaitu fungsi *setCipher*, fungsi *setKey*, fungsi *reset*, fungsi *permute*, fungsi *xor*, fungsi *leftShift*, fungsi *binToStr*, fungsi *initialPermutation*, fungsi *createSubKey*, fungsi *feistelCipher*, fungsi *round*, dan fungsi *decrypt*.

Fungsi *setCipher* untuk mengatur nilai atribut *cipher*. Fungsi *setKey* untuk mengatur nilai atribut *key*. Fungsi *reset* untuk mengatur ulang nilai dari atribut *cipher*, *key*, dan *roundKey*. Fungsi *permute* merupakan fungsi dasar dalam DES, penjelasan mengenai fungsi ini dapat dilihat pada Gambar 2.10. Fungsi *xor* berguna untuk melakukan operasi *bit xor*. Fungsi *leftShift* berguna untuk menggeser *bit* ke arah kiri sebanyak *n-bit*.

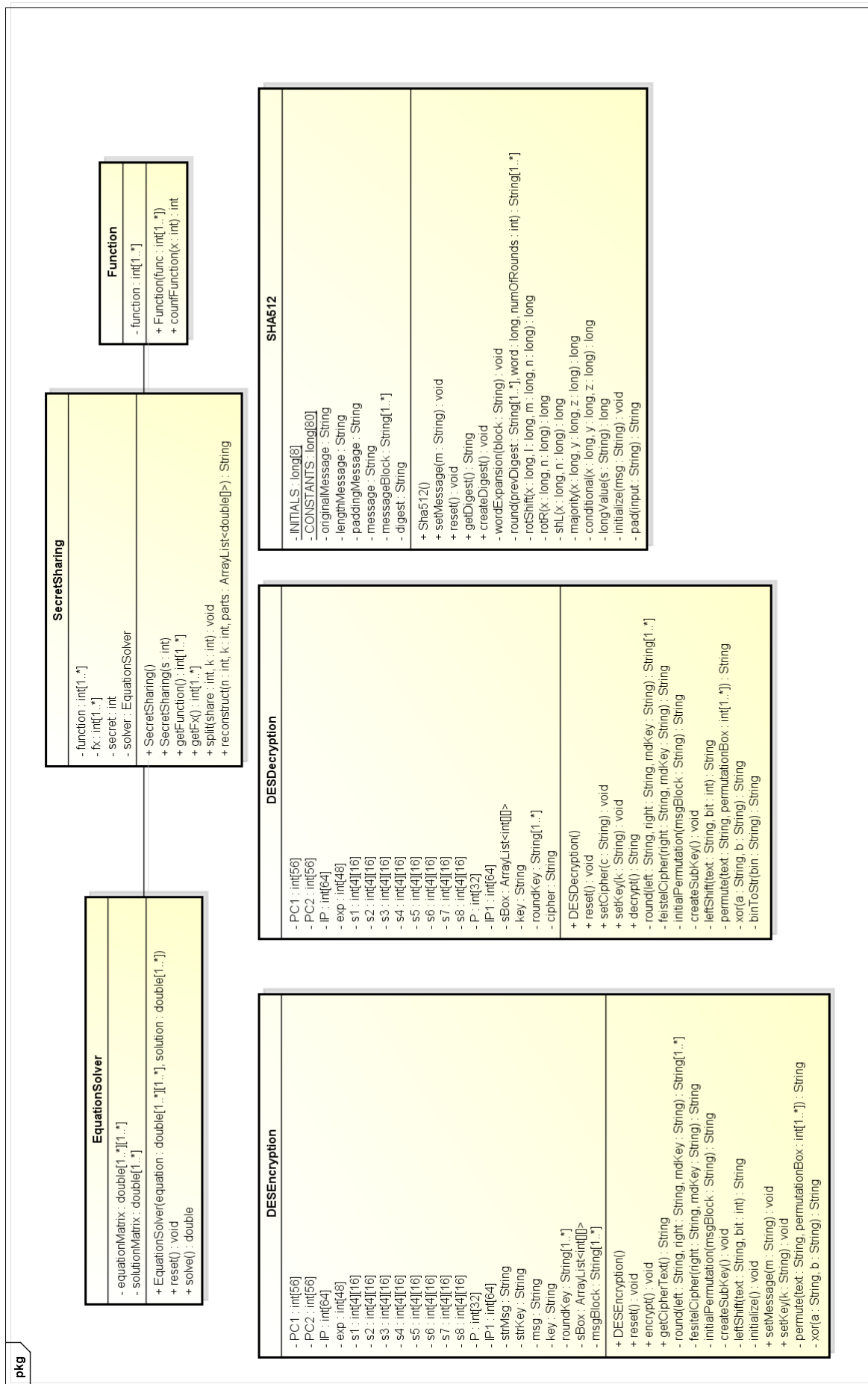
Fungsi *initialPermutation* berguna untuk melakukan permutasi awal. Fungsi *createSubKey* berguna untuk membangun kunci untuk setiap ronde. Fungsi *feistelCipher* merupakan implementasi dari sandi feistel. Fungsi *round* adalah menjalankan satu ronde dari DES. Fungsi *decrypt* menjalankan 16 ronde *inverse* dalam DES.

#### 4.4.7 *DataReader*

Kelas *DataReader* merupakan kelas yang berperan untuk menerima masukan berupa berkas teks. Kelas ini memiliki 2 atribut, yaitu *filename* dan *content*. Atribut *filename* menyimpan nama file dari berkas teks yang akan dibaca oleh perangkat lunak. Atribut *content* menyimpan isi dari berkas teks yang dibaca oleh perangkat lunak. Kelas *DataReader* memiliki 2 fungsi, yaitu fungsi *get* dan fungsi *read*. Fungsi *get* berguna untuk mengembalikan nilai dari atribut *content*. Fungsi *read* berguna untuk membaca berkas teks lalu menyimpannya ke dalam atribut *content*.

#### 4.4.8 *DataWriter*

Kelas *DataWriter* merupakan kelas yang berperan untuk menulis keluaran ke dalam berkas teks. Kelas *DataWriter* memiliki 2 atribut, yaitu *filename* dan *content*. Atribut *filename* menyimpan nama berkas teks keluaran akan ditulis. Atribut *content* menyimpan hasil keluaran dari perangkat lunak yang akan ditulis ke dalam berkas teks. Kelas *DataWriter* memiliki 1 fungsi, yaitu fungsi *write*. Fungsi *write* berguna untuk menulis keluaran ke dalam berkas teks.



Gambar 4.5: Diagram Kelas Rinci



3

## BAB 5

4

## IMPLEMENTASI DAN PENGUJIAN

- 1 Pada bab ini akan berisi mengenai implementasi perangkat lunak dan pengujian dari kualitas
- 1 pertanyaan keamanan yang dibuat.





## DAFTAR REFERENSI



1001

**LAMPIRAN A**

1002

**THE PROGRAM**