# PROPOSAL

**IMPLEMENTASI ALGORITMA SHA-512 DAN *VIGENERE***

***CIPHER* UNTUK PENGAMANAN FILE DARI SERANGAN**

**PEMBAJAKAN (*MAN-IN-THE-MIDDLE DAN SSH BRUTE FORCE ATTACK*)**

**Diajukan Untuk Memenuhi**

**Salah Satu Syarat Memperoleh Derajat Sarjana Teknik**



**NUR HAZMILA**

**E1E1 15 037**

# JURUSAN TEKNIK INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS HALUOLEO KENDARI

## 2020

# DAFTAR ISI

## HALAMAN JUDUL ........................................................................................ i LEMBAR PENGESAHAN.............................................................................. ii

**PERNYATAAN................................................................................................. iv INTISARI.......................................................................................................... v *ABSTRACT........................................................................................................* vi HALAMAN PERSEMBAHAN.......................................................................**

## KATA PENGANTAR....................................................................................... vii DAFTAR ISI...................................................................................................... ix DAFTAR GAMBAR........................................................................................ xii DAFTAR TABEL............................................................................................. xiv

**BAB I PENDAHULUAN** ................................................................................. 1

1.1 Latar Belakang.................................................................................. 1

1.2 Rumusan Masalah............................................................................. 3

1.3 Batasan Masalah................................................................................ 4

1.4 Tujuan Penelitian............................................................................... 4

1.5 Manfaat Penelitian............................................................................. 5

1.6 Sistematika Penulisan........................................................................ 5

1.7 Tinjauan Pustaka............................................................................... 6

## BAB II LANDASAN TEORI .......................................................................... 9

2.1 Jaringan Komputer............................................................................ 9

2.2 Topologi jaringan Komputer............................................................. 10

2.3 Internet.............................................................................................. 11

2.4 E-mail ............................................................................................... 12

2.5 Pengertian File Teks ......................................................................... 13

2.6 Keamanan Data dan Jaringan............................................................ 13

2.7 *Man In The Middle Attack*................................................................. 14

2.8 SSH Brute Force Attack*....................................................................* 15

2.9 Kriptografi......................................................................................... 15

2.9.1 Defenisi Kriptografi ................................................................ 15

2.9.2 Terminologi dan konsep dasar kriptografi................................16

2.9.3 Sistem Kriptografi ................................................................... 17

2.9.4 Tujuan Kriptografi .................................................................. 18

2.10 Kode ASCII dan UNCODE............................................................ 19

2.11 Algoritma........................................................................................ 20

2.11.1 Pengertian Algoritma............................................................. 20

2.11.2 Jenis-jenis Algoritma Kriptografi.......................................... .20

2.11.2.1 Algoritma kriptografi simetris (konvesional) ............20

2.11.2.2 Algoritma kriptografi Asimetri (kunci publik).......... .21

2.12 *Secured Hash Algorithm* 512.......................................................... 22

2.13 Algoritma Klasik............................................................................. 25

2.14 Bahasa Pemrograman...................................................................... 26

2.14.1 Pengenalan Java................................................................... 26

2.14.2 *NetBeans*.............................................................................. 27

2.15 *UML (Unified Modelling Language*)*..............................................* 27

2.17 *Use Case Diagram*.......................................................................... 29

2.18 *Sequence Diagram*.......................................................................... 30

2.19 *Activity Diagram*............................................................................. 31

2.20 *Linux Ubuntu*................................................................................... 32

## BAB III METODOLOGI PENELITIAN ...................................................... 34

3.1 Metode Pengumpulan Data............................................................... 34

3.2 Metode Pengembangan Perangkat Lunak......................................... 34

3.3 Waktu Penelitian............................................................................... 36

3.4 Tempat Penelitian.............................................................................. 36

4.1 Analisis Sistem.................................................................................. 37

4.1.1 Analisis Masalah...................................................................... 37

4.1.2 Analisis Kebutuhan Fungsional............................................... 37

4.1.2.1 Analisis Kebutuhan Input............................................. 38

4.1.2.2 Analisis Kebutuhan Output.......................................... 38

4.1.3 Analisis Kebutuhan *Nonfungsional*.......................................... 38

4.1.3.1 Analisis Kebutuhan Perangkat Keras........................... 39

4.1.3.2 Analisis Kebutuhan Perangkat Lunak.......................... 39

4.2 Perancangan Sistem.......................................................................... 39

4.2.1 *Use Case Diagram*................................................................... 40

4.2.2 Activity *Diagram*..................................................................... 41

4.2.3 Sequence *Diagram*................................................................... 43

4.3 Rancangan *Interface* ......................................................................... 45

4.3.1 Halaman Utama Sistem............................................................ 45

4.3.2 Form Memasukkan Kunci Enkripsi File.................................. 46

4.3.3 Form Memasukkan Kunci Dekripsi File.................................. 46

4.4 Rancangan Topologi Jaringan........................................................... 47

**DAFTAR PUSTAKA……………………………………………………...........**

# DAFTAR GAMBAR

## Gambar 2.1 Topologi Star............................................................................... 11 Gambar 2.2 Proses Enkripsi dan Dekripsi.................................................... 16 Gambar 2.3. Skema Kriptografi Simetris...................................................... 21 Gambar 2.4 Skema Kriptografi Asimetris..................................................... 21 Gambar 2.5 Gambar 2.5 Operasi fungsi hash SHA-512.............................. 23 Gambar 2.6 Contoh *Sequence Diagram*........................................................ 30 Gambar 4.1 Rancangan *Use case Diagram* Sistem....................................... 40 Gambar 4.2 *Activity Diagram* Enkripsi File................................................... 41 Gambar 4.3 *Activity Diagram* Dekripsi File................................................... 42 Gambar 4.4 *Sequence Diagram* Enkripsi File............................................... 43 Gambar 4.5 *Sequence Diagram* Dekripsi File................................................ 44 Gambar 4.6 Tampilan Halaman Utama........................................................ 45 Gambar 4.7 Tampilan Form Memasukkan Kunci Enkripsi File................ 46 Gambar 4.8 Tampilan Form Memasukkan Kunci Dekripsi File............... 46 Gambar 4.9 Topologi Jaringan....................................................................... 47

# DAFTAR TABEL

## Tabel 2.1 Algoritma *Vigenere Chiper*.............................................................. 26 Tabel 2.2 Normalisasi *Hexadecimal*................................................................. 26 Tabel 2.3 Simbol-simbol *Use Case* Diagram................................................... 28 Tabel 2.4 Simbol-Simbol *Activity* Diagram.................................................... 31 Tabel 3.1 Waktu Penelitian.............................................................................. 36 Tabel 4.1 Spesifikasi perangkat keras............................................................. 39 Tabel 4.2 Spesifikasi perangkat lunak............................................................ 39

xix

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Perkembangan teknologi komputer dan interkoneksi melalui jaringan telah mendunia mulai dari perusahaan besar hingga penggunaan pribadi membuat lalu lintas data semakin meningkat. Seiring dengan perkembangan dunia IT yang semakin pesat, ada sebagian pihak yang ingin mengetahui data orang lain (*Hacker*) yang membuat kerahasiaan suatu data terancam keamanannya.

Keamanan menjadi faktor penting dalam proses pengiriman data melalui jaringan. Hal tersebut dikarenakan dalam proses pengiriman data akan melalui banyak *server* sebelum sampai ke tujuan. Untuk menjamin hal tersebut diperlukan suatu sistem agar informasi atau data yang dikirim dari satu pihak kepihak lain dapat sampai dengan utuh dan selamat ketujuan. Sama halnya data yang di simpan dapat terjamin keamanannya sehingga tidak dapat dibuka atau dibaca oleh pihak yang tidak diinginkan. Keamanan menjadi hal yang sangat penting bagi semua pengguna namun tidak dapat dipungkiri bahwa jaminan keamanan masih lemah hingga saat ini.

Pada 6 bulan awal tahun 2018, ada 4,5 miliar data dicuri berdasarkan perusahaan Gemalto, jumlah ini meningkat sebanyak 113% dari tahun lalu, akan tetapi kasus pencurian data mengalami penurunan dari 1.162 kasus pada tahun lalu menjadi 945 kasus di tahun ini. Data yang diretas mencapai 6,9 juta data per harinya menurut laporan dari total 14,6 miliar sejak tahun 2013. Media sosial menjadi target terbesar para pihak ketiga yang tidak bertanggung jawab yaitu pencurian sebesar 56,11%, kemudian diikuti dengan data-data instansi pemerintah sebesar 26,62% yang sudah pasti sangat penting. Meskipun demikian hanya sebagian kecil dari data-data tersebut terlindungi dengan enkripsi yaitu hanya sekitar 4% saja (Agustino Setyo Wardani, 2018).

Ada beberapa bentuk serangan pada komunikasi jaringan di antaranya adalah serangan *Man-In-The-Middle* dan *SSH Brute Force.* Dalam serangan *Man-In-The*

1

*Middle* melibatkan dua titik akhir (korban) dan pihak ketiga (penyerang). Penyerang memiliki akses pada saluran komunikasi antara dua titik akhir dan bisa memanipulasi pesan mereka (Risqiwati, D., Irawan, E. A., 2018). Dalam skenario ini penyerang menempatkan dirinya di tengah-tengah dua perangkat yang saling berkomunikasi sehingga si penyerang bisa mengetahui, membaca, mancuri bahkan memanipulasi data yang ada.

*SSH Brute Force* adalah suatu bentuk serangan yang membuat pengguna jaringan yang sama dapat mengakses komputer *server* dari jarak jauh. *SSH brute force attack* dapat mencuri dan memanipulasi isi komputer korban dengan cara mengambil alih *remot control terminal server*, sehingga sipenyerang dapat mengambil, menghapus, memanipulasi data dengan leluasa.

Salah satu solusi menyelesaikan masalah diatas adalah melakukan proses pengamanan *file* (enkripsi dan dekripsi) terhadap data yang akan dikirimkan. Enkripsi dilakukan untuk mengubah data asli menjadi data rahasia. Sedangkan dekripsi dilakukan oleh penerima untuk mengetahui isi data asli dengan cara mengubah data rahasia menjadi data asli. Proses ini disebut dengan teknik kriptografi dimana selama proses pengiriman data, data yang dikirim harus bersifat rahasia dan data asli hanya diketahui oleh pengirim dan penerima dengan menggunakan kunci rahasia.

SHA-512 (*Secured Hash Algorithm-* 512) adalah algoritma kriptografi yang menggunakan fungsi hash satu arah yang diciptakan oleh **Ron Rivest**. Algoritma ini merupakan pengembangan dari algoritma-algoritma sebelumnya yaitu algoritma SHA-0, SHA-1, SHA 256, dan algoritma SHA-384. Cara kerja kriptografi algoritma SHA 512 adalah menerima *input* sembarang dan menghasilkan *Message Diggest* yang memiliki panjang 512 bit.

Algoritma kriptografi klasik digunakan sejak sebelum era komputerisasi dan kebanyakan menggunakan teknik kunci simetris. Metode menyembunyikan pesannya adalah dengan teknik substitusi atau transposisi atau keduanya. Teknik substitusi adalah menggantikan karakter dalam *plaintext* menjadi karakter lain yang hasilnya adalah *ciphertext*. Sedangkan transposisi adalah teknik mengubah *plaintext* menjadi *ciphertext* dengan cara permutasi karakter (Sumandri, 2017).

Salah satu contoh algoritma kriptograsi sederhana adalah algoritma *Vigenere Cipher.* Dalam pengembangannya, *Vigenere Cipher* dinilai sebagai salah satu algoritma yang cepat, tersusun dengan rapi, sederhana, dan panjang kunci digunakan sebagai variabel dalam hal menggunakan sandi pada *file* yang telah dienkripsi.

Berdasarkan pemaparan diatas maka penulis melakukan penelitian yang berjudul “*Implementasi Agoritma SHA-512 dan Vigenere Cipher Untuk Pengamanan File Dari Serangan Pembajakan (Man-In-The-Middle dan SSH Brute Force Attack)*”. Dengan menggunakan Algoritma SHA-512 sebagai metode verifikasi kunci diharapakan dapat meningkatkan perlindungan kunci karena kunci asli akan dituliskan pada *file* terenkripsi sebanyak 128 *byte* pertama, dan akan melakukan verifikasi kunci asli pada proses dekripsi dengan 128 *byte* pertama pada *file* terenkripsi tersebut. Sedangkan untuk isi *file* dapat diproses dengan menggunakan algoritma sederhana *Vigenere Cipher* berdasarkan kunci yang digunakan, dimana panjang kunci akan diulang secara periodik sebanyak jumlah *plaintext,* hingga semua isi *file* dapat terenkripsi. Untuk menguji keamanan hasil enkripsi file penulis meng-implementasikan serangan *Man-In-The-Middle* dan *SSH Brute Force Attack*.

## 1.2 Rumusan masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, rumusan masalah yang akan dipaparkan yaitu :

1. Apakah SHA-512 mampu melakukan proses verifikasi kunci pada sistem yang akan dibangun?
2. Bagaimana implementasi algoritma SHA-512 sebagai metode verifikasi kunci dan algoritma *Vigenere Cipher* untuk menghasilkan *cipherfile* dan *plainfile* kembali pada sistem yang akan dibangun?
3. Bagaimana performasi sistem aplikasi yang telah dirancang terhadap keamanan *file*?

## 1.3 Batasan Masalah

Hal – hal yang menjadi batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Objek yang dijadikan sebagai data penelitian adalah *file* yang berekstensi *\*.txt, \*.doc/docx, dan \*.pdf.*
2. Bahasa Pemrograman yang digunakan adalah bahasa pemrograman java.
3. Algoritma SHA-512 diimplementasikan sebagai metode verifikasi kunci
4. Algoritma sederhana *Vigenere cipher* digunakan untuk enkripsi dan dekripsi isi *file* berdasarkan kunci yang digunakan.
5. Aplikasi dibangun hanya untuk mengamankan *file* dengan proses enkripsi dan dekripsi *file* terhadap serangan pembajakan (*Man-In-The-Middle* dan *SSH Brute Force Attack*).
6. Sistem yang akan dibangun berbasis desktop
7. Metode serangan/pembajakan diimplementasikan hanya untuk menguji keamanan data hasil enkripsi.

## 1.4 Tujuan penelitian

Berdasarkan permasalahan yang ada, adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Untuk mengetahui kemampuan algoritma SHA-512 sebagai metode verifikasi kunci dalam proses enkripsi dan dekripsi *file* pada sistem yang dibangun.
2. Untuk mengimplementasikan algoritma SHA-512 dan algoritma sederhana *Vigenere Cipher* sebagai sistem keamanan file dari serangan pembajakan (*Man-In-The-Middle* dan *SSH Brute Force Attack*).
3. Untuk mengetahui performasi sistem aplikasi yang telah dirancang terhadap keamanan *file*.

## 1.5 Manfaat Penelitian

Adapun Manfaat yang diharapkan penulis dalam penelitian ini adalah sebagai berikut:

1. Dapat mengimplementasikan algoritma SHA-512 sebagai verifikasi kunci dan algoritma sederhana *Vigenere cipher* pada java sebagai sistem keamanan *file*
2. Dapat mengetahui performasi system aplikasi yang telah dirancang terhadap keamanan *file*.

## 1.6 Sistematika Penulisan

Tugas akhir ini terdiri dari enam bab, masing-masing merupakan rangkaian sistematis dalam pengkajian materi berdasarkan rumusan masalah yang telah dikemukakan pada BAB I dengan sistematika sebagai berikut:

## BAB I PENDAHULUAN

Bagian ini berisi pendahuluan yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan dan tinjauan pustaka.

## BAB II LANDASAN TEORI

Bagian ini berisi materi yang berhubungan dengan tema penelitian baik yang bersumber dari media cetak maupun media elektronik meliputi pengertian Internet, pengertian jaringan computer, Topologi jaringan, kriptografi, algoritma, keamanan data dan jaringan, file teks, algoritma kriptografi SHA-512, *Vigenere Cipher*, pengenalan *Man-In-The-Middle dan SSH Brute Force Attack*, pemodelan UML (*Unified Modeling Language*), *use case diagram*, *activity diagram*, *java Script*, *linux ubuntu*.

## BAB III METODOLOGI PENELITIAN

Bagian ini memuat prosedur dan pengumpulan data, prosedur pengembangan perangkat lunak serta waktu dan tempat penelitian.

## BAB IV ANALISIS DAN PERANCANGAN

Pada bagian ini diuraikan analisis sistem yang akan dibuat dan kebutuhan sistem yang meliputi kebutuhan fungsional dan kebutuhan non fungsional sistem. Rancangan sistem meliputi rancangan arsitektur sistem, rancangan proses, rancangan data dan rancangan antarmuka pengguna.

## BAB V IMPLEMENTASI DAN PEMBAHASAN

Bagian ini menyajikan implementasi serta hasil penelitian dan pembahasan. Bagian implementasi menguraikan tentang implementasi secara detail dan terurut, bagian hasil penelitian dan pembahasan hasil dari implementasi disertai dengan penjelasan yang terkait.

## BAB VI PENUTUP

Bagian ini memuat kesimpulan dari penelitian yang telah dilakukan serta saran untuk penelitian selanjutnya.

### 1.7 Tinjauan Pustaka

Beberapa penelitian tentang implementasi *algoritma SHA (SHA 512)* terhadap pengkodean data adalah sebagai berikut :

Pada tahun 2017, penelitian juga dilakukan oleh S.M. Mohidul Islam, dkk., dengan judul “*Bit Adjusting Image Steganography in Blue Channel using AES and Secured Hash Function*”. Pada penelitian tersebut penulis mengusulkan teknik steganografi menggunakan *Advanced Encryption Standard* (AES) dengan fungsi hash aman di saluran biru gambar. Sistem penyematan dilakukan dengan sistem penyesuaian *bit* dinamis dalam saluran biru gambar RGB. Ini menanamkan *bit* pesan ke lebih dalam ke intensitas gambar yang sangat sulit untuk semua jenis manipulasi peretas. Sebelum menanamkan teks dienkripsi menggunakan AES dengan fungsi hash. Untuk ekstraksi *bit* teks sandi ditemukan dari intensitas gambar menggunakan algoritma ekstraksi penyesuaian *bit* dan kemudian didekripsi oleh AES dengan fungsi hash yang sama untuk mendapatkan teks rahasia yang sebenarnya. Adapun kesimpulan dari penelitian ini adalah Pendekatan yang diusulkan lebih baik dalam nilai *Pick Signal to Noise Ratio* (PSNR) dan lebih sedikit kesalahan histogram antara gambar stego dan gambar sampul daripada beberapa sistem yang ada.

Pada tahun 2018, penelitian dilakukan juga oleh Alfry Aristo Jansen Sinla E, dkk., dengan judul “Rancang Bangun Kriptosistem Untuk Pengamanan DataData Digital”. Penelitian tersebut bertujuan untuk merancang dan membangun suatu aplikasi keamanan data/kriptosistem dan mengimplementasikan algoritma *Rivest Shamir Adleman* (RSA) dan fungsi hash SHA-512 ke dalam kriptosistem tersebut. Data digital yang digunakan adalah *file* bertipe dokumen *(.doc/.docx dan .pdf)*, file bertipe teks *(.txt)*, dan file bertipe *audio (.mp3 dan .wav).* Hasil dari penelitian ini adalah kriptosistem dapat melakukan proses enkripsi dan dekripsi dengan baik, sehingga memenuhi aspek-aspek keamanan dari kriptografi yaitu kerahasiaan, integritas data, otentikasi dan penyangkalan.

Pada tahun 2019 penelitian juga dilakukan oleh Faizal Achmad, dkk., yang berjudul “*Implementasi Algoritme Kriptografi Two Fish dan SHA-512 Sebagai Aplikasi Pengamanan Informasi File Digital*”. Penelitian tersebut untuk mengamankan informasi rahasia yang berbentuk *file* serta implementasi fungsi SHA 512 sebagai kontrol akses pengaman *password Login* pengguna. Adapun kesimpulan dalam penelitian ini adalah Aplikasi “*Two Fish File Encryptor*” dapat melakukan enkripsi semua tipe *file*, sehingga informasi pada *file* tidak diketahui oleh pihak yang tidak berwenang.

Pada tahun 2019 penelitian juga dilakukan oleh Candra Irawan, dkk., dengan judul “*Implementasi Algoritma Autokey Cipher dan AES-128 pada Enkripsi file*”. Dalam penelitian tersebut, peneliti memanfaatkan algoritma *Autokey cipher* digabung dengan *Advanced Encryption Standard* (AES-128), kemudian untuk menambah tingkat keamanan, kunci yang digunakan untuk enkripsi dan dekripsi *file* memanfaatkan fungsi hash SHA-512. Dengan dimanfaatkannya algoritma-algoritma tersebut, algoritma tidak akan mudah dipecahkan oleh kriptanalis. *File* yang sudah terenkripsi akan membuat pihak ketiga tidak dapat langsung membaca *file* karena *bit-bit fil*e sudah teracak, dengan demikian data yang tersimpan di dunia maya baik itu komputer pribadi, *cloud*, atau data yang dibagikan melalui suatu aplikasi akan terlindungi. Adapun *file* yang menjadi bahan penelitian dalam penelitian tersebut adalah mengenkripsi berbagai *file* teks, gambar, musik, maupun video. Adapun kesimpulan dari penelitian tersebut adalah Peneliti berhasil membangun *software* bernama *VertoCrypt* berbasis desktop untuk mengenkripsi *file* sehingga *file* dapat terlindungi dengan mengimplementasikan metode SHA-512 untuk verifikasi kunci serta *Autokey cipher* dan AES-128 untuk enkripsi *file*.

Pada Tahun 2020, penelitian dilakukan oleh Anju A James, dengan judul “*A Novel High-Speed IoT Based Crypto Lock Using AES-128 and SHA-512”*.

Dalam penelitian tersebut, sipeneliti memanfaatkan algoritma SHA-512 dan AES128 untuk merancang kunci *crypto* berbasis IoT baru untuk bangunan cerdas atau rumah pintar. Kombinasi AES-128 dan SHA-512 diperkenalkan untuk sistem manajemen kunci pintu yang sangat aman. kunci *crypto* akan diminta untuk mengatur kata sandi dari sistem kunci menggunakan tombol. Kemudian memasukkan kata sandi, parameter keamanan akan diaktifkan dan servomotor akan mengunci pintu. Klien memasukkan kata sandi yang tepat dan pintu akan dibuka menggunakan rotasi servomotor. Jika orang yang tidak berwenang menebak kata sandi sebagai upaya yang salah, alarm keamanan akan diaktifkan dan pemberitahuan akan dikirim ke orang yang berwenang. Jika seorang pencuri mencoba memasuki ruangan menggunakan pintu belakang tanpa menggunakan kata sandi, sensor PIR mendeteksi entri yang tidak sah dan mengaktifkan alarm keamanan, dan kemudian juga pemberitahuan diberikan kepada pengguna.

**BAB II LANDASAN TEORI**

### 2.1 Jaringan Komputer

Jaringan komputer (*computer network*) adalah hubungan dua buah komputer atau lebih yang bertujuan untuk melakukan pertukaran data dengan mudah. Diantaranya berbagi perangkat lunak (*software)* dan perangkat keras (*hardware)*, bahkan berbagi kekuatan pemrosesan data sehingga mempersingkat waktu pengerjaan dan meningkatkan efisiensi kerja (Fauzi, A., & Suartana, I., 2018 ).

Ada banyak tipe jaringan komputer, berikut tipe jaringan berdasarkan ruang lingkupnya.

1. LAN (*Local Area Network*): LAN adalah sebuah Jaringan komputer yang cakupan areanya kecil, seperti sebuah rumah, kantor, atau sekolah. Karakteristik khusus dari LAN yang membedakan dengan jenis jaringan yang lain adalah transfer data yang lebih besar, namun cakupan area yang lebih sempit. Teknologi yang dipakai untuk membuat LAN ada beberapa macam, ada ARCNET dan Token Ring, namun yang lazim digunakan sekarang adalah Ethernet dan kabel UTP.
2. MAN (*Metropolitan Area Network*): MAN adalah jaringan komputer yang cakupan luasnya mencapai satu kota atau lebih. Sebuah MAN biasanya menghubungkan antara beberapa LAN lokal dengan menggunakan teknologi backbone, seperti fiber optik, dan beberapa penyedia layanan internet yang sering di sebut ISP (*Internet Service Provider*).
3. WAN (*Wide Area Network*): WAN adalah jaringan komputer yang cakupannya cukup luas, seperti antar regional atau antar negara. Salah satu contoh koneksi dalam jaringan WAN adalah *Leased Line*, merupakan koneksi *point-to-point* antara 2 komputer atau *Local Area Network* (LAN) dengan area yang cukup jauh (antar negara) (Nugroho, B., 2013).

9

### 2.2 Topologi Jaringan Komputer

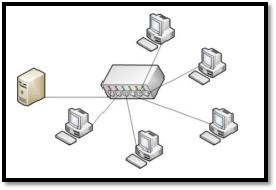
Topologi dapat diartikan sebagai *layout* atau arsitektur jaringan dari komputer. Topologi merupakan suatu aturan bagaimana jaringan komputer (*node*) secara fisik, topologi berkaitan dengan komponen-komponen jaringan seperti *server*, *workstation*, *switch*, *router* dan lainnya yang saling berkomunikasi melalui media transmisi data (Sofana, I., 2013).

Topologi jaringan memiliki dua kategori, yaitu *phisycal topology* (topologi fisik) dan *logical topology* (topologi logika). Topologi fisik berkaitan dengan bentuk jaringan, sedangkan topologi logika berkaitan dengan bagaimana data mengalir di dalam topologi fisik. Topologi sendiri umumnya terdiri dari beberapa bagian, yaitu:

### 1. Topologi *Bus* 2. Topologi *Ring* 3. Topologi *Tree* 4. Topologi *Mesh* 5. Topologi *Star*

Menurut Iwan Sofana (2013) topologi *star* adalah topologi yang sering digunakan dan lebih mudah dikonfigurasi. Topologi ini membuat tiap perangkatnya di hubungkan pada perangkat penghubung (sentral). Topologi *Star* ini merupakan salah satu topologi yang sering dijumpai dalam sebuah jaringan komputer, apabila mempunyai tiga atau lebih komputer dan sebuah perangkat penghubung, dengan menggunakan *hub*, *switch*, atau *router*, maka ketiga komputer bisa saling terkoneksi. Karakteristik topologi *star* antara lain adalah:

1. Kabel yang digunakan berjenis *coaxial*, UTP, dan STP.
2. Menggunakan penghubung berupa *hub*, *switch* dan *router*.
3. Jika salah satu segmen kabel putus pada satu atau lebih *node*, maka hanya segmen itu saja yang akan lumpuh sementara jaringan dapat tetap berfungsi.
4. Pengiriman data menggunakan metode CSMA/CD (*Carrier Sense Multiple Access*/*Collision Detection*) *baseband*.
5. Data mengalir pada sebuah kabel secara bolak-balik.
6. Sering terjadi banjir data dan *collision* (tabrakan data) sehingga dapat menurunkan performa jaringan, terutama pada penghubung hub.
7. Relatif lebih mahal dibandingkan dengan topologi *bus*, namun proses instalansi mudah dan cocok diimplementasikan pada jarinngan kecil maupun besar.



**Gambar 2.1 Topologi *Star* (Sofana, I., 2013)**

#### 2.3 Internet

*Internet* berasal dari kata *Interconnection Networking* dan mempunyai arti hubungan komputer dengan berbagai tipe yang mempunyai arti hubungan komputer dengan berbagai tipe yang membentuk sistem jaringan mencangkup seluruh dunia (jaringan computer global) dengan melalui jalur telekomunikasi seperti telepon, radiolnk, satelit dan lainnya sehingga didalamnya menghubungkan situs-situs akademik, pemerintahan, komersial, organisasi, maupun perorangan.

*Internet* mengacu pada istilah jaringan bukan aplikasi. *Internet* tidak memiliki manfaat apa.bila tidak ada aplikasi yang sesuai dan bisa digunakan untuk berbagai keperluan. Setiap aplikasi berjalan diatas *protocol* tertentu. Istilah “**protokol**” di *internet* mengacu pada satu set aturan yang mengatur bagaimana sebuah aplikasi berkomunikasi dalam satu jaringan (Alam , 2016 ).

#### 2.4 E-mail

*Electronic mail* (E-mail) berfungsi sebagai sarana untuk mengirim surat atau pesan melalui jaringan Intenet, dengan email kita hanya membutuhkan beberapa menit agar surat/pesan dapat sampai tujuan, tidak perlu menunggu berhari-hari seperti mengirim surat/pesan biasa (Pos) dan dengan E-mail isi surat/pesan dapat kita isi dengan konten gambar/suara dan video (Supratman, L. P., dan Wahyudin, A., 2017).

E-mail bukan hanya untuk mengirim surat/pesan, saat ini E-mail dapat saling terhubung dengan internet seperti mendaftar *facebook, twitter, blogger* dan lainlain. Sebuah pesan elektronis terdiri dari isi, alamat pengirim, dan alamat-alamat yang dituju. Sistem E-mail yang beroperasi diatas jaringan berbasis pada model *store and forward* (Supratman, L. P., dan Wahyudin, A., 2017).

Sistem ini mengaplikasikan sebuah sistem *server* E-mail yang menerima, meneruskan, mengirimkan, serta menyimpan pesan pesan *user*, dimana *user* hanya perlu untuk mengkoneksikan pc mereka kedalam jaringan. E-mail dapat dianalogikan dengan kotak surat yang ada di kantor POS sedangkan *server* E-mail dapat diibaratkan sebagai kantor POS. Dengan analog ini sebuah mail *server* dapat memiliki banyak *account email* yang ada didalamnya.

Salah satu layanan E-mail yang sering digunakan adalah *Google Email* (Gmail). Berdasarkan data yang baru diumumkan di *twitter*, jumlah pengguna aktifnya kini telah mencapai 1,5 milyar. Gmail memulai layanannya di *desktop* pada April 2004 sebagai versi beta terbatas. Titel beta tersebut berakhir pada 2009. Versi java tersedia untuk pengguna ponsel pada 2006. Pada 2012, Gmail memiliki 425 juta pengguna aktif, kemudian jumlahnya naik dua kali lipat menjadi 900 juta pada 2015. Seiring waktu jumlah pengguna akif mengalami peningkatan.

Dengan menggunakan E-mail milik Google ini, pengguna dapat mengakses Email dalam bentuk surat web HTTPS, *protocol* POP3 ataau IMAP4. Gmail menyedikan kapasitas penyimpanan sebanyak lebih dari 7538 *megabyte* dan terus bertambah. Jumlah ini lebih dari jumlah yang disediakan situs lain seperti *Yahoo!* dan *Hotmail*. Hal ini berarti para pengguna dapat menyimpan sampai ribuan surat elektronik. Sampai saat ini, Gmail merupakan Email dengan kapasitas terbanyak (Supratman, L. P., dan Wahyudin, A., 2017).

#### 2.5 Pengertian *File* Teks

Teks adalah permukaan dari fenomena karya sastra. Teks adalah kata-kata yang membentuk karya dan yang disusun dengan cara sedemikian rupa untuk membelokkan arti yang tetap dan seunik mungkin. Karena teks merupakan tenunan yang dijalin, teks sebagai jaringan, yang secara konstitutif berhubungan dengan tulisan, maka teks mempunyai fugsi menjaga tetapnya dan permanennya inkripsi yang ditulis agar ingatan terbantu (Manullang, D. I., 2018).

*File* adalah entitas dari data yang disimpan didalam sistem *file* yang dapat diakses dan diatur oleh pengguna. Sebuah *file* memiliki nama yang unik dalam direktori dimana ia berada. Alamat direktori dimana suatu berkas ditempatkan diistilahkan dengan **path**. Sebuah *file* berisi aliran data (data stream) yang berisi sekumpulan data yang disebut dengan *properties* yang berisi informasi mengenai *file* yang bersangkutan seperti informasi mengenai kapan sebuah berkas dibuat (Manullang, D. I., 2018).

*File Teks* merupakan salah satu bentuk *file* yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca (Manullang, D. I., 2018).

#### 2.6 Keamanan Data dan Jaringan

Secara umum data dikategorikan menjadi dua, yaitu data yang bersifat rahasia dan data yang tidak bersifat rahasia. Data yang tidak bersifat rahasia biasanya tidak akan terlalu diperhatikan.Data yang perlu diperhatikan adalah data yang bersifat rahasia, dimana setiap informasi yang ada didalamnya akan sangat berharga bagi pihak yang membutuhkan karena data tersebut dapat dengan mudah digandakan. Untuk mendapatkan informasi didalamnya, biasanya dilakukan berbagai cara yang tidak sah (D. Nurnaningsih *and* A. A. Permana,2018).

Keamanan jaringan adalah data-data yang berada pada perangkat keras dan perangkat lunak dalam sistem jaringan dilindungi dari tindakan-tindakan yang bersifat jahat atau merusak, modifikasi dan hal-hal yang bersifat membocorkan data ke pihak lain, untuk memastikan sistem akan berjalan secara konsisten dan handal tanpa adanya gangguan pada sistem tersebut (Fauzi, A., & Suartana, 2018).

#### 2.7 *Man-In-The-Middle Attack*

Dalam kriptografi, penyerangan dengan metode *Man-In-The-Middle* (*MITM*) adalah sebuah bentuk penyadapan dimana sang penyerang membuat sebuah koneksi yang independen, antara korban dan mengirimkan pesan diantara para korban yang mengira mereka,sedang berkomunikasi pada sebuah koneksi *privat* dimana sebenarnya semua percakapan tersebut diatur oleh sang penyerang. Pada metode ini, sang penyerang diharuskan untuk bisa menyadap sebuah pesan yang dikomunikasikan antara kedua korban dan memasukan pesan baru. Penyerangan ini hanya bisa sukses jika dan hanya jika sang penyerang bisa menyamar menjadi setiap *endpoint* dari korban dengan persetujuan yang lainnya. Jadi dengan serangan MITM ini Attacker tidak hanya bersifat pasif, tetapi juga aktif mengubah komunikasi yang terjadi. Salah satu contoh serangan *MITM (Man-*

*In-The-Middle)* adalah *ARP Spoofiing* atau *poisoning* dan *DNS Spoofing* (Alam,

2016).*.*

*ARP (Address Resolution Protocol*) *poisoning* atau *ARP Spoofing* ini adalah suatu teknik menyerang pada jaringan komputer lokal baik dengan media kabel atau *wireless*, yang memungkinkan penyerang bisa mengendus *frame* data pada jaringan lokal dan atau melakukan modifikasi *traffic* atau bahkan menghentikan *traffic. ARP spoofing* merupakan konsep dari serangan penyadapan diantara terhadap dua mesin yang sedang berkomunikasi atau yang disebut dengan *MITM (Man in The Middle Attack).* Prinsip serangan *ARP poisoning* ini memanfaatkan kelemahan pada teknologi jaringan komputer itu sendiri yang menggunakan *arp broadcast. ARP* berada pada layer 2, dimana alamat pada layer dua adalah *MAC address.* Misalnya sebuah *host (contoh: PC*) yang terhubung pada sebuah *LAN* ingin menghubungi *host* lain pada *LAN* tersebut, maka dia membutuhkan inforamsi *MAC address* dari *host* tujuan ( Fauzi, A., & Suartana, 2018).

#### 2.8 *SSh Brute Force Attack*

*SSh Brute Force Attack* secara harfiah mempunyai arti serangan SSH secara paksa, dimana mempunyai arti sebuah aktifitas untuk terus menggali sebuah informasi secara terus menerus. Layanan SSH dapat memungkinkan pengguna jaringan untuk mengakses sebuah komputer dari jarak jauh (M.Aguk dan

Anggraini, 2018).

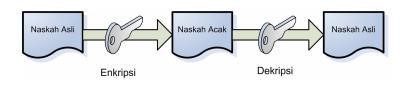
Enkripsi yang digunakan oleh SSH menyediakan kerahasiaan dan integritas data melalui jaringan yang tidak aman seperti Internet. SSH menggunakan kriptografi kunci publik untuk mengautentikasi komputer remote dan membiarkan komputer *remote* untuk mengautentikasi pengguna, jika perlu. SSH biasanya digunakan untuk login ke mesin *remote* dan mengeksekusi berbagai perintah. SSH dapat digunakan untuk mentransfer *file* melalui SFTP atau SCP. SSH menggunakan *client-server* model. Standar TCP port 22 telah ditetapkan sebagai jalur untuk *server SSH*. Sebuah klien SSH biasanya digunakan untuk membangun koneksi ke SSH daemon supaya dapat dikendalikan *via remote* (Sakti, B., & Aziz, A, 2013).

#### 2.9 Kriptografi

##### 2.9.1 Defenisi kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan berita [*bruce schneier – applied cryptography*]. *Crypto* berarti “*secret*” (rahasia) dan “*graphy*” berarti “*writing*” (tulisan). Sebuah algoritma kriptografik (*cryptographic algorithm*), disebut ***ciphe*r**, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Biasanya kedua persamaan matematik (untuk enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat (Fauzi, A, 2019).

Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. Proses enkripsi dilakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma dianggap sesuatu yang tidak baik. Rahasia terletak di beberapa parameter yang digunakan, jadi kunci ditentukan oleh parameter. Parameter yang menentukan kunci dekripsi itulah yang harus dirahasiakan (parameter menjadi ekuivalen dengan kunci) (Suanda,R., 2018).



## Gambar 2.2 Proses Enkripsi dan Dekripsi (Suanda,R., 2018)

Dapat dilihat dari gambar 2.2 merupakan proses dari enkripsi dan dekripsi suatu file teks. Pesan asli disebut *plaintext*, dan pesan menyamar disebut *ciphertext*. Pesan terakhir, dikemas dan dikirim, disebut kriptogram. Proses transformasi *plaintext* menjadi *ciphertext* disebut enkripsi atau *enciphering*. Proses kebalikan dari mengubah *ciphertex*t menjadi *plaintext*, yang dicapai oleh penerima yang memiliki pengetahuan untuk menghapus menyamar, disebut dekripsi atau mengartikan. Siapapun yang terlibat dalam kriptografi disebut *kriptografe*r (Suanda,R., 2018).

### 2.9.2 Terminologi dan konsep dasar kriptografi

Didalam kriptografi akan sering ditemukan beberapa istilah atau *terminology*. Berikut merupakan istilah penting untuk diketahui :

1. Plainteks dan cipherteks

Pesan merupakan data atau informasi yang dapat dibaca maknanya. Bentuk pesan yang tidak disandikan disebut plainteks atau dapat dikenal juga sebagai *cleartext*. Dan bentuk pesan yang telah tersandikan disebut cipherteks. Cipherteks harus dapat dikembalikan menjadi plainteks semula agar pesan yang diterima bisa dibaca (Sandyansyah, 2018).

1. Pengirim dan penerima

Komunikasi data melibatkan pertukaran pesan antara dua entitas. Pengirim adalah entitas yang mengirim pesan kepada entitas lainnya. Penerima adalah entitas yang menerima pesan. Entitas disini dapat berupa orang, mesin (komputer), kartu kredit, dan sebagainya (Sandyansyah, 2018).

1. Enkripsi dan dekripsi

Proses mengubah plainteks menjadi cipherteks disebut enkripsi atau *enciphering*. Dan proses mengembalikan cipherteks menjadi plainteks, dimana penerima mengetahui cara untuk menghapus penyamaran pesan disebut deskripsi atau *deciphering* (Sandyansyah, 2018).

1. Kriptanalis

Kriptografi berkembang sedemikian rupa sehingga melahirkan bidang yang berlawanan yaitu kriptanalisis. Kriptanalisis (*cryptanalysis*) adalah ilmu dan seni untuk memecahkan cipherteks menjadi plainteks tanpa mengetahui kunci yang digunakan. Pelakunya disebut kriptanalis. Jika seorang kriptografer (*cryptographer*) mentransformasikan plainteks menjadi cipherteks dengan suatu algoritma dan kunci maka sebaliknya seorang kriptanalis berusaha untuk memecahkan cipherteks tersebut untuk menemukan plainteks atau kunci (Sandyansyah, 2018).

### 2.9.3 Sistem kriptografi

Secara umum istilah kriptografi yang sering digunakan adalah :

1. **Pesan**, yaitu data atau informasi yang dapat dibaca dan dimengerti maknanya, pesan sering juga disebut dengan *plaintext* atau teks jelas *(cleartext*). *Plaintext* merupakan suatu pesan bermakna yang akan diproses menggunakan algoritma kriptografi.
2. **Enkripsi**, yaitu proses penyandian *plaintext* menjadi *ciphertext* atau disebut *enciphering*. Enkripsi dilakukan dengan tujuan agar *plaintext* tersebut tidak dibaca oleh pihak yang tidak memiliki otoritas (wewenang).
3. ***Ciphertext*,** yaitu pesan yang telah tersandi. Pesan dalam bentuk *ciphertext* tidak dapat dibaca karena berisi karakter-karakter yang tidak memiliki makna setelah melalui enkripsi.
4. **Dekripsi,** yaitu proses pengembalian *ciphertext* menjadi *plaintext* semula atau disebut *deciphering*. Dekripsi dilakukan ketika pesan telah sampai kepada pihak yang dituju.
5. **Kunci** (***key***) adalah parameter yang digunakan untuk transformasi enkripsi dan dekripsi. Kunci dapat juga berupa *string* atau deteran bilangan. Keamanan suatu algoritma kriptografi biasanya bergantung pada kerahasiaan penyebaran kunci (*key*).
6. **Kriptosistem (*Cryptosystem*)**, yaitu perangkat keras atau implementasi perangkat lunak kriptografi yang diperlukan atau mentransformasi sebuah pesan asli menjadi *ciphertext* atau sebaliknya (Suanda,R., 2018).

### 2.9.4 Tujuan kriptografi

Menurut (Munir, 2006, h. 9), ada empat tujuan mendasar dari ilmu kriptograf i ini yang juga merupakan aspek keamanan informasi yaitu :

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/ mengupas informasi yang telah disandi.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. untuk menjaga integritas data, system harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
3. *Autentikas*i, adalah berhubungan dengan identifikasi atau pengenalan baik secara kesatuan system atau informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman dan lain-lain.
4. *No-Repudiasi* atau penyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi (Fauzi, A, 2019).

### 2.10 Kode ASCII dan UNICODE

Kode Standar Amerika untuk pertukaran informasi atau ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti *Hex* dan *Unicode* tetapi ASCII lebih bersifat *universal*, contohnya 124 adalah untuk karakter “|”. Kode ini selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan *biner* sebanyak 7 *bit*. Namun, ASCII disimpan sebagai sandi 8 *bit* dengan menambakan satu angka 0 sebagai *bit significant* paling tinggi. *Bit* tambahan ini sering digunakan untuk uji prioritas. Karakter *control* pada ASCII dibedakan menjadi 5 kelompok sesuai dengan penggunaan yaitu berturut-turut meliputi *logical communication*, *Device control, Information separator, Code extention, dan physical communication*. *Code* ASCII ini banyak dijumpai pada papan ketik (keyboard) computer atau instrument-instrument digital.

Jumlah kode ASCII adalah 255 kode. Kode ASCII 0..127 merupakan kode ASCII untuk manipulasi teks; sedangkan kode ASCII 128. 255 merupakan kode ASCII untuk manipulasi grafik. Kode ASCII sendiri dapat dikelompokkan lagi kedalam beberapa bagian:

1. Kode yang tidak terlihat simbolnya seperti Kode 10 (Line Feed), 13(Carriage Return), 8 (Tab), 32 (Space).
2. Kode yang terlihat simbolnya seperti abjad (A..Z), numerik (0..9), karakter khusus (~!@#$%^&\*()\_+?:”{})
3. Kode yang tidak ada di *keyboard* namun dapat ditampilkan. Kode ini umumnya untuk kode-kode grafik.

Dalam pengkodean kode ASCII memanfaatkan 8 *bit*. Pada saat ini kode ASCII telah tergantikan oleh kode UniCode (*Universal Code*). UniCode dalam pengkodeannya memanfaatkan 16-*bit* sehingga memungkinkan untuk menyimpan kode-kode lainnya seperti kode bahasa Jepang, Cina, Thailand dan sebagainya (Irawan, R., Ilhamsyah, Brianorman,Y.,2015).

### 2.11 Algoritma

#### 2.11.1 Pengertian Algoritma

Algoritma adalah urutan langkah-langkah logis dalam penyelesaian masalah yang disusun secara sistematis dan logis. Kata logis merupakan kata kunci dalam algoritma. Langkah-langkah dalam algoritma harus logis dan harus dapat ditentukan bernilai salah atau benar (Sandyansyah, 2018).

Dalam beberapa konteks, algoritma adalah spesifikasi urutan langkah untuk melakukan pekerjaan tertentu. Pertimbangan dalam pemilihan algoritma adalah pertama, algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli sebagus apapun algoritma, kalau memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik (Sandyansyah, 2018)

#### 2.11.2 Jenis-jenis algoritma kriptorafi

Berdasarkan jenis kunci, algoritma kriptografi dikelompokan menjadi dua bagian yaitu algoritma kriptografi simetri (konvesional) / *Symetric Cryptosystem*, dan algoritma kriptografi asimetri (kunci publik) / *Assymetric Cryptosystem*.

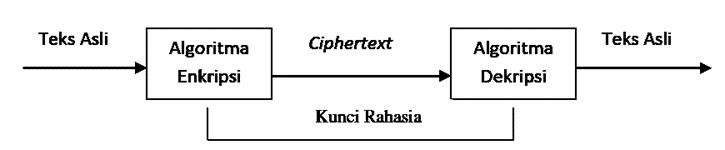
##### 2.11.2.1 Algoritma kriptografi simetris (konvensional)

Algoritma simetris adalah algoritma dimana kunci enkripsi dapat dihitung dari kunci dekripsi dan sebaliknya. Algoritma ini juga disebut algoritma kuncirahasia, algoritma sama-kunci, algoritma satu-kunci, yang mengharuskan pengirim dan penerima menyetujui suatu kunci sebelum mereka dapat berkomunikasi dengan aman (Suanda, R., 2018).

Dalam kebanyakan algoritma simetri, kunci enkripsi dan kunci dekripsi adalah sama. Algoritma ini juga disebut algoritma kunci-rahasia, algoritma samakunci, algoritma satu-kunci, yang mengharuskan pengirim dan penerima menyetujui suatu kunci sebelum mereka dapat berkomunikasi dengan aman. Keamanan algoritma simetri terletak pada kuncinya, membocorkan kunci berarti membiarkan siapa saja bisa mengenkripsi dan mendekripsi pesan. Selama komunikasi tetap rahasia, maka kunci harus tetap dirahasiakan (Schneier, 1996).

Yang termasuk algoritma kunci simetri adalah OTP (*One Time Pad*), RC2, RC4, RC5, *Twofish*, *Skipjack*, dan lainnya (Sandyansyah, 2018).

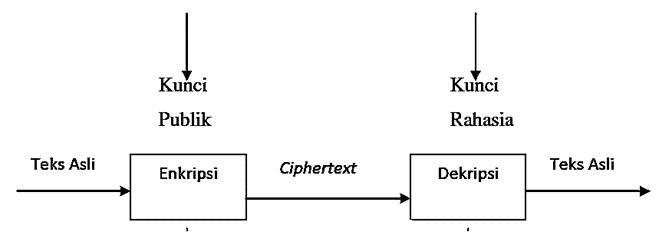
kriptografi simetri dapat dilihat pada gambar 2.3.



**Gambar 2.3. Skema Kriptografi Simetris (suanda, R., 2018)**

##### 2.11.2.2 Algoritma kriptografi asimetris (kunci publik)

Sekitar pertengahan tahun 1970an, muncul konsep baru dalam kriptografi yaitu kriptografi *public-key* (*asimetri*). Seseorang yang mengetahui cara mengenkripsi naskah / teks asli (*plaintext*) belum tentu mengetahui juga cara mendekripsi naskah/teks acak (*ciphertext*) yang dihasilkan. Demikian juga seseorang yang mengetahui cara mendekripsi naskah / teks acak (*ciphertext*) belum tentu mengetahui juga cara mengenkripsi naskah/teks asli (*plaintext*) untuk menghasilkan naskah/teks acak (*ciphertext*) tersebut. Enkripsi dan dekripsi dalam kriptografi *public-key* menggunakan sepasang kunci yaitu kunci public (*publickey*) dan kunci privat (*private-key*). Naskah/teks yang telah dienkripsi menggunakan kunci privat hanya dapat didekripsi menggunakan kunci publik dan naskah yang dapat didekripsi menggunakan kunci publik dapat dipastikan telah dienkripsi menggunakan kunci privat. Sebaliknya, naskah yang telah dienkripsi menggunakan kunci publik hanya dapat didekripsi menggunakan kunci privat (Sandyansyah, 2018). Skema kriptografi asimetri dapat dilihat pada gambar 2.4.



**Gambar 2.4 Skema Kriptografi Asimetris (Suanda, R., 2018)**

### 2.12 *Secured Hash Algorithm* (SHA-512)

Fungsi hash merupakan salah satu bagian dari kriptografi. Fungsi hash adalah fungsi yang dapat memberikan nilai terhadap data masukan. Fungsi hash menerima masukan string yang panjangnya sembarang, kemudian mentransformasikannya menjadi string keluaran yang panjangnya tetap (umumnya berukuran jauh lebih kecil daripada ukuran semula). *String* keluaran tersebut disebut juga dengan nilai hash. Umumnya, fungsi hash ini digunakan untuk keperluan autentikasi dan integritas data.

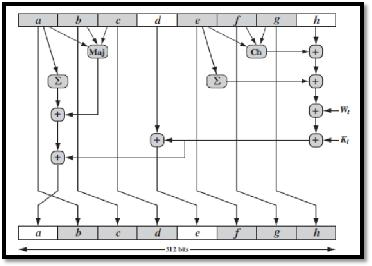
SHA-1 adalah fungsi hash pertama yang ditawarkan oleh NIST. SHA-1 merupakan perbaikan dari SHA-0. SHA-1 adalah fungsi hash 160 bit, yang mengartikan bahwa keluarannya, atau yang disebut juga *digest*, memiliki panjang 160 bit. SHA-1 masih aman untuk digunakan, tetapi orang dapat juga menggunakan algoritma SHA-2.

SHA-2 adalah nama informal untuk algoritma SHA putaran kedua yang didesain oleh NIST. Termasuk juga SHA-224, SHA-256, SHA-384, dan SHA-512. Nilai 224, 256, 384, atau 512 menyatakan panjang *digest* yang dikeluarkan. Jadi fungsi SHA-512 merupakan variasi dari SHA-2 atau perkembangan dari SHA-1, yang membedakan dari masing-masing standar adalah ukuran blok yang dipakai (Setiawan, 2010).

Fungsi yang terjadi pada algoritma SHA-2 mirip dengan yang terjadi pada SHA-1. Proses yang terjadi pada SHA-512 adalah:

1. Sebelum Pemrosesan :
   1. Mengubah *string* masukan menjadi kumpulan *bit* 0 dan 1.
   2. Penambahan k *bit* ‘0’, dimana k adalah nilai minimum > = 0 sehingga pesan yang memiliki kelipatan 1024 bit.
   3. Tambahkan panjang pesan, dalam *bi*t, sebagai 64-*bit integer*.
2. Pada saat pemrosesan
   1. Memiliki 8 buah penyangga yang berukuran 64 *bit*.
   2. Memiliki 80 buah konstanta yang berukuran 64 *bit*.
   3. Bagi kumpulan *bit* setiap 1024 *bit*.
   4. Pada tiap kumpulan 1024 *bit* tersebut, bagi tiap *bi*t tersebut menjadi 16 bagian.
   5. Perpanjang jadi 80 bagian, dengan menggunakan fungsi *Gamma0* dan *Gamma1.*

Adapun operaasi yang terjadi pada tiap putaran dapat dilihat pada gambar 2.5 di bawah ini :



**Gambar 2.5 Operasi fungsi hash SHA-512 (Sakti, B., & Aziz, A, 2013)**

Nilai A sampai H adalah 8 buah penyangga yang telah ditentukan sebelumnya. Fungsi Maj yang ada pada gambar diatas adalah fungsi yang menerima 3 buah variabel dengan aturan melakukan operasi : ( ( A ∨ B ) ∧ C ) ∨ (A ∧ B) ) (Setiawan,w., 2011).

Sedangkan fungsi Sigma0 adalah fungsi yang melakukan operasi : ( ROR (x,28) ⊕ ROR (x,34) ⊕ ROR (x,39) ). Yang dimaksud dengan ROR adalah rotasi *bit* yang ada ke kanan. Sehingga, yang terjadi pada fungsi *Sigma0* adalah rotasikan x ke kanan sebanyak 2 kali , kemudian dilakukan operasi xor dengan x yang dirotasi ke kanan sebanyak 13 kali dan xor dengan x yang dirotasi kanan sebanyak 22 kali. Fungsi Sigma1 juga memiliki operasi yang mirip, yaitu : ( ROR (x,14) ⊕ ROR (x,18) ⊕ ROR (x,41) ) (Setiawan,W., 2011).

Fungsi *Gamma0* dan *Gamma1* adalah fungsi yang dilakukan untuk memperbanyak jumlah potongan pada tiap bagian menjadi 8 0 bagian dari 16 bagian. Fungsi ini mirip dengan fungsi sebelumnya, yaitu fungsi *Sigma0* dan *Sigma1*. Fungsi *Gamma0* memiliki persamaan ( ROR (x,1) ⊕ ROR (x,8) ⊕ R (x,7) ). Fungsi R berbeda dengan fungsi rotasi kanan. Fungsi R merupakan fungsi untuk melakukan shift bit ke kanan sebanyak yang telah ditentukan. Sedangkan, fungsi *Gamma1* memiliki persamaan (ROR (x,19) ⊕ ROR (x,61) ⊕ R (x,6))

(Setiawan,W., 2011).

Pada gambar terlihat ada Wx dan Kx. Yang dimaksud dengan nilai Kx adalah nilai konstanta yang telah ditentukan pada tiap putaran. Sedangkan, nilai Wx adalah nilai dari tulisan yang akan di-hash yang sudah ditambah bitnya dan diperpanjang ukurannya sampai 80 buah potongan. Untuk mendapatkan nilai W ke 16 sampai dengan 79 (jika nilai W pertama adalah nilai W ke 0), operasi yang dilakukan adalah melakukan operasi penambahan antara *Gamma1* dari W[i-2] dengan W[i-7] dan juga ditambah dengan hasil penjumlahan antara *Gamma0* dari W[i-15] dan W[i-16] (Setiawan,W., 2011).

Kemudian, terdapat juga fungsi Ch. Fungsi ini melakukan operasi (G ⊕ (E ∧ (F ⊕ G))). Setelah melakukan operasi-operasi yang ada, nilai selanjutnya dapat ditentukan. Penentuan nilai selanjutnya dilakukan dari H sampai ke A. Nilai H ditentukan dari nilai G, nilai G ditentukan dari nilai F, nilai F ditentukan dari nilai E, nilai E ditentukan dari penjumlahan dari D dan operasi penjumlahan, nilai D ditentukan dari nilai C, nilai C ditentukan dari nilai B,nilai B ditentukan dari nilai A,dan nilai A diperoleh dari penjumlahan (Setiawan,W., 2011).

Fungsi hash diperoleh dengan menggabungkan nilai-nilai hasil penjumlahan sebelumnya. Fungsi hash SHA512 melakukan operasi hash yang sama dengan operasi SHA-2 pada umumnya. Yang menjadi perbedaan dengan algoritma hash SHA-2 yang lainnya adalah angkanya memiliki panjang 64 *bit*, sedangkan SHA256 hanya memiliki panjang 32 *bit*. Selain itu, pada SHA-512 terdapat 80 buah putaran, sedangkan SHA-256 hanya memiliki 64 putaran. Pada SHA-512, nilai 8 buah penyangganya dan konstanta penambahnya diperpanjang mencapai 64 *bit*. *Operasi shift* dan rotasi yang dilakukan pada SHA-512 juga berbeda (Setiawan,W., 2011).

SHA-512 adalah varian dari SHA yang beroperasi pada delapan 64-*bit word*.

Pesan yang akan dihitung nilai hash-nya diproses sebagai berikut:

1. Dilakukan proses *padding* pada pesan dengan panjangnya sehingga hasilnya adalah kelipatan dari panjang 1024-*bit*.
2. Urai menjadi 1024-*bi*t blok pesan )



)

,



,

…

,



Blok pesan diproses satu kali pada satu waktu: Dimulai dengan nilai *fixed* initial hash H(i), dihitung secara berurutan:

). Dimana C ada fungsi kompresi SHA-512 dan



)

=



)

+



)

,

…

,



“+” berarti penjumlahan *word-wise* mod 264. H(N) adalah hash dari M

(Setiawan,W., 2011).

### 2.13 Algoritma Klasik

Algoritma kriptografi klasik digunakan sejak sebelum era komputerisasi dan kebanyakan menggunakan teknik kunci simetris. Metode menyembunyikan pesannya adalah dengan teknik substitusi atau transposisi atau keduanya. Teknik substitusi adalah menggantikan karakter dalam *plaintext* menjadi karakter lain yang hasilnya adalah *ciphertext.* Sedangkan transposisi adalah teknik mengubah *plaintext* menjadi *ciphertext* dengan cara permutasi karakter. Kombinasi keduanya secara kompleks adalah yang melatarbelakangi terbentuknya berbagai macam algoritma kriptografi modern. Contoh algoritma kriptografi klasik yaitu: *Caesar Cipher, Vigenere Cipher, dan Hill Cipher* (Sumandri, 2017).

*Vigenere cipher* merupakan salah satu algoritma klasik dengan Teknik substitusi. Nama *vigenere* diambil dari seorang yang bernama **Blaise de Vigenere**.

*Vigenere cipher* menggunakan suatu kunci yang memiliki panjang tertentu. Panjang kunci tersebut bisa lebih pendek ataupun sama dengan panjang plainteks. Jika panjang kunci kurang dari panjang plainteks, maka kunci yang tersebut akan diulang secara periodik hingga panjang kunci tersebut sama dengan panjang plainteksnya.

**Tabel 2.1 Algoritma *Vigenere Chiper***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|  | O | P | Q | R | S | T | U | V | W | X | Y | Z |  |

**Tabel 2.2 Normalisasi *Hexadecimal***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |

Algoritma enkripsi *vigenere cipher* :

*Ci* (*Pi*  *Ki*)mod26 (1)

Algoritma dekripsi *vigenere cipher* :

*Pi* (*Ci*  *Ki*)mod26 (2)

Dimana :

Ci = nilai desimal karakter *ciphertext* ke-i

Pi = nilai desimal karakter *plaintext* ke-i

Ki = nilai desimal karakter kunci ke-I (Sumandri, 2017).

### 2.14 Bahasa Pemrograman

Bahasa pemrograman merupakan prosedur penulisan. Ada tiga *record* dalam penulisan bahasa pemograman yaitu :

1. *Syntax adalah* aturan penulisan bahasa tersebut (tata bahasa).
2. *Semantic* adalah arti atau maksud yang terkandung di dalam *statement* tersebut.
3. Kebenaran logika adalah berhubungan dengan benar tidaknya urutan *statement* (Fridayanthie, C., 2016).

#### 2.14.1 Pengenalan Java

Java adalah sebuah teknologi yang diperkenalkan oleh **Sun Microsystem** pada pertengahan tahun 1990. Menurut defenisi **Sun**, java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada *Computer Standalone* ataupun pada lingkungan jaringan. Teknologi java memiliki tiga komponen penting, yaitu *Programming-Language spesification*, *Aplicationprogramming interface*, dan *Virtual-machine spesification* JDK *(java Development Kit*) (Bay Haqi, 2017).

Java adalah adalah bahasa pemrograman yang dapat dijalankan diberbagai komputer, termasuk telepon genggam. Bahasa pemrograman ini pertama kali dibuat oleh James Gosling saat masih bergabung di *Sun Microsystems*. Bahasa pemrograman ini merupakan pengembangan dari bahasa C++ karena banyak mengadopsi sintak C dan C++.

Saat ini Java merupakan bahasa pemrograman yang paling popular digunakan. Kelebihan Java dari bahasa pemrograman yang lain adalah bisa dijalankan diberbagai jenis sistem operasi sehingga dikenal juga bahasa pemrograman multiplatform, bersifat Pemrograman Berorientasi Objek (PBO), memiliki *library* yang lengkap (Fridayanthie, C., 2016).

#### 2.14.2 NetBeans

NetBeans adalah *Integrated Development Environment* (IDE), berbasiskan java dari Sun Microsystem yang berjalan diatas Swing. Swing sebuah teknologi java untuk mengembangkan aplikasi Desktop yang dapat berjalan diberbagai macam palatforms seperti Windows, Linux, Mac OS X, and Solaris (Bay Haqi,

2017).

IDE adalah lingkup pemrograman yang diintegrasikan kedalam aplikasi perangkat lunak yang menyediakan pembangun *Graphic User Interface* (GUI), suatu text atau kode editor, suatu *Compiler atau Interpreter* atau suatu *debugger*

(Bay Haqi, 2017).

Salah satu yang menjadi kelebihan NetBeans GUI Builder adalah yang telah disebutkan diatas, yaitu gratis. Selain itu NetBeans GUI Builder sangat kompatibel dengan Swing karena di kembangkan oleh Sun yang merupakan pengembang Swing (Bay Haqi, 2017).

### 2.15 UML (*Unified Modelling Language*)

*Unified Modelling Language* adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

*Unified Modelling Language* merupakan suatu alat untuk perancangan sistem pada objek. Dengan menggunakan alat bantu *Unified Modelling Language* kita dapat membuat sesuatu model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi ini dapat berjalan pada perangkat keras, sistem operasi dan jaringan internet apapun, serta ditulis dalam bahasa pemrograman (Zulkifli, 2018).

### 2.16 *Use Case Diagram*

*Use case* adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oeh sebuah aktor. *Use case* digunakan untuk membentuk tingkah laku benda dalam sebuah mode serta direalisasikan oleh sebuah *collaborator*, umumnya *use case* digambarkan dengan sebuah *elips* dengan garis yang *solid*, biasanya mengandung nama*. Use case* menggambarkan proses sistem (kebutuhan sistem dari sudut pandang *user*). Secara umum *use case* adalah pola perilaku sistem dan urutan transaksi yang berhubungan yang dilakukan oleh aktor. *Use Case* Diagram terdiri dari: *use case, aktor, relationship, sistem boundary boxes, packages* (Suendri, (2018). Adapun simbol-simbol dari *use case* diagram dapat dilihat pada table 2.3.

**Tabel 2.3 Simbol-simbol *Use Case Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| No | Gambar | Nama | Keterangan |
| 1. |  | *Actor* | Menspesifikasikan himpunan pesan yang pengguna mainkan ketika berinteraksi dengan *use case*. |
| 2. |  | *Use case* | Deskripsi dari urutan aksi-aksi yang ditampilkan system yang menghasilkan suatu hasil yang terukur bagi suatu *actor*. |

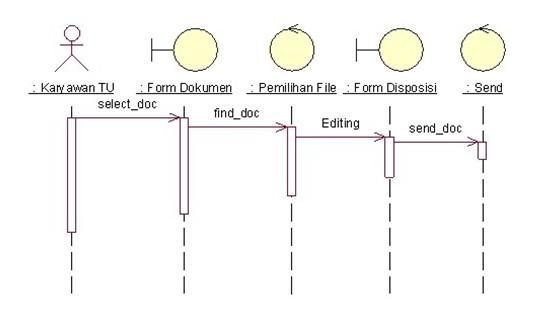
**Tabel 2.3 Simbol-simbol *Use Case Diagram (*Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| 3. |  | *Include* | Menspesifikasikan bahwa *use case*  secara *eksplisit* |
| 4. |  | *System* | Menspesifikasi paket yang menampilkan sistem -sistem secara terbatas |

### 2.17 *Sequence Diagram*

Suatu *Sequence* diagram adalah suatu penyajian perilaku yang tersusun sebagai rangkaian langkah-langkah percontohan dari waktu ke waktu. *Sequence diagram* digunakan untuk menggambarkan arus pekerjaan, pesan yang sampaikan dan bagaimana elemen-elemen di dalamnya bekerja sama dari waktu ke waktu untuk mencapai suatu hasil.

Masing–masing urutan elemen diatur di dalam suatu urutan horisontal, dengan pesan yang disampaikan dibelakang dan didepan diantara elemen-elemen. Seorang elemen aktor digunakan untuk menghadirkan pemakai yang memulai alur peristiwa/ kejadian. Elemen–elemen yang ditiru, seperti *boundary*, *control* dan *entity*, digunakan untuk menggambarkan layar, pengontrol, dan materi database, secara berturut-turut. Masing-masing elemen yang dihubungkan garis–garis batang disebut suatu lifeline, di mana jika unsur itu berpotensi mengambil bagian dalam interaksi itu (Suendri, (2018). Contoh *sequence* diagram dapat dilihat pada gambar 2.6 berikut.



**Gambar 2.6 Contoh *Sequence Diagram***

Suatu model *sequence diagram* adalah suatu pandangan yang dinamis menyangkut interaksi antar unsur-unsur model pada runtime. *Sequence diagram* biasanya digunakan sebagai model yang menjelaskan *scenario* kasus penggunaan. Dengan penciptaan suatu diagram urutan dengan object dan aktor dilibatkan di dalam kasus penggunaan, kamu dapat model urutan langkah-langkah pemakai dan sistem melakukan untuk melengkapi tugas yang diperlukan itu. Suatu Diagram urutan adalah sering dipasang secara langsung di bawah suatu kasus penggunaan yang ditunjuk. Hal ini menyimpan unsur bersama-sama, baik dalam model dan ketika dokumentasi diproduksi.untuk mengerjakan ini, klik kanan kasus penggunaan pada diagram dan memilih *add sequence* ((Suendri, (2018).

Contoh dibawah menunjukan beberapa unsur *sequence* :

1. Actor satu kejadian dari suatu aktor pada runtime
2. Menggunakan Kasus- Suatu kejadian suatu kasus penggunaan pada runtime
3. Obyek-suatu unsure standar yang tidak diketik
4. Batas-menghadirkan seorang alat penghubung layar pemakai atau *input/output* alat
5. Kesatuan-secara khas diterapkan sebagai unsure atau database.
6. Pengontrol-komponen aktf yang mengendalikan pekerjaan apa yang dilakukan, kapan dan bagaimana caranya (Suendri, (2018).

### 2.18 *Activity* Diagram

*Activity* Diagram adalah lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis. Memiliki struktur diagram yang mirip *flowchart* atau *data flow diagram* pada perancangan terstruktur. Memiliki pula manfaat yaitu apabila penulis membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan (Suendri, 2018). Berikut adalah symbol-simbol *activity diagram*:

**Tabel 2.4 Simbol-Simbol *Activity Diagram***

No

Symbol

Nama

Keterangan

1.

*Activity*

Memperlihatkan

bagaimana

masing-masing

kelas

antarmuka

saling

berinteraksi

satu

sama

lain

2.

*Action*

State

dari

sistem

yang

mencerminkan

eksekusi

dari

suatu

aksi.

3.



*Initial*

*Node*

Bagaimana

objek

dibentuk

atau

diawali

4.



*Activity*

*final*

*node*

Bagaimana

objek

dibentuk

atau

diakhiri

5.

*Line*

*Connector*

Digunakan

untuk

menghubungkan

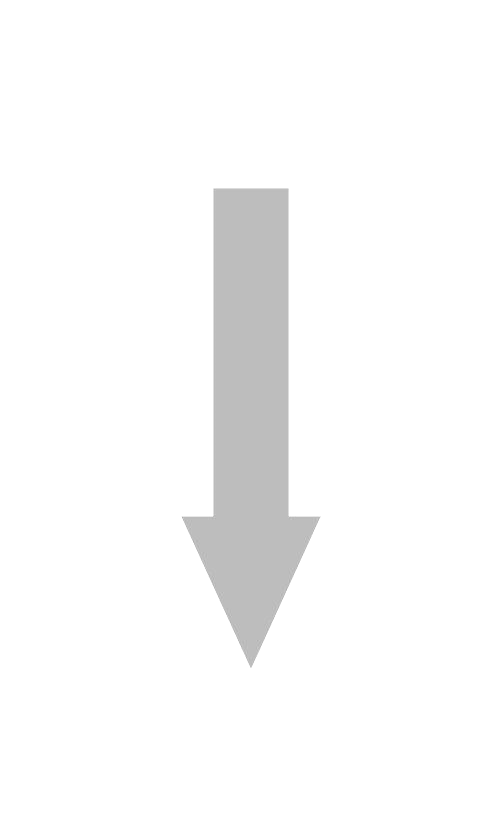
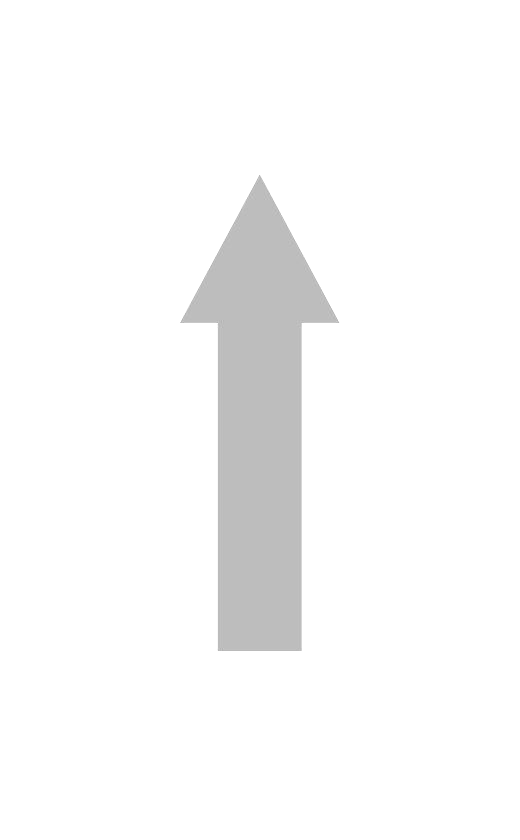
satu

simbol

dengan

simbol

lainnya.



### 2.19 Linux Ubuntu

Linux ubuntu adalah sistem operasi *Linux* adalah suatu sistem operasi komputer bertipe *unix* yang bersifat *open source*. Karena sifatnya *open source*, maka kode sumber *linux* dapat dimodifikasi, digunakan dan didistribusikan kembali secara bebas oleh siapapun. Linux pada awalnya dibuat oleh seorang Menspesifikasikan himpunan pesan yang pengguna mainkan ketika berinteraksi dengan *use case.*mahasiswa Finlandia yang bernama Linus Torvalds. Dulunya *linux* merupakan proyek hobi yang diinspirasikan dari Minix, yaitu sistem *unix* kecil yang dikembangkan oleh Andrew Tanenbaum.Linux versi 0.01 dikerjakan sekitar bulan Agustus 1991. Kemudian pada tanggal 5 Oktober 1991, Linus mengumumkan versi resmi *linux*, yaitu versi 0.02 yang hanya dapat menjalankan shell bash (GNU Bourne AgainShell) dan gcc (GNU C *Compiler*) (Megawati,

2012).

Saat ini *linux* adalah sistem *unix* yang sangat lengkap, bisa digunakan untuk jaringan, pengembangan software dan bahkan untuk pekerjaan sehari-hari. Linux sekarang merupakan alternatif sistem operasi yang jauh lebih murah jika dibandingkan dengan sistem operasi komersial (misalnya Windows 9.x/NT/2000/ME). *Linux* mempunyai perkembangan yang sangat cepat. Hal ini dapat dimungkinkan karena *linux* dikembangkan oleh beragam kelompok orang. Keragaman ini termasuk tingkat pengetahuan, pengalaman serta geografis (Megawati, 2012).

*Linux* bisa didapatkan dalam berbagai distribusi (sering disebut Distro). Distro adalah bundel dari kernel *linux*, beserta sistem dasar linux, program instalasi, tools basic, dan program-program lain yang bermanfaat sesuai dengantujuan pembuatan distro. Ada banyak sekali distro *linux*, diantaranya : Ubuntu, RedHat, Debian, Slackware, SuSE, Fedora, Kali Linux dan lain-lain.

Ubuntu merupakan salah satu distribusi *linux* yang berbasiskan [Debian](https://id.wikipedia.org/wiki/Debian) dan didistribusikan sebagai [perangkat](https://id.wikipedia.org/wiki/Perangkat_lunak_bebas) [lunak](https://id.wikipedia.org/wiki/Perangkat_lunak_bebas) [bebas](https://id.wikipedia.org/wiki/Perangkat_lunak_bebas). Nama Ubuntu berasal dari filosofi dari [Afrika](https://id.wikipedia.org/wiki/Afrika_Selatan) [Selatan](https://id.wikipedia.org/wiki/Afrika_Selatan) yang berarti kemanusiaan kepada sesama. Ubuntu dirancang untuk kepentingan penggunaan pribadi, namun versi *server* Ubuntu juga tersedia, dan telah dipakai secara luas (Megawati, 2012).

Proyek Ubuntu resmi disponsori oleh [Canonical](https://id.wikipedia.org/wiki/Canonical_Ltd.) [Ltd.](https://id.wikipedia.org/wiki/Canonical_Ltd.) yang merupakan sebuah perusahaan yang dimiliki oleh pengusaha [Afrika](https://id.wikipedia.org/wiki/Afrika_Selatan) [Selatan](https://id.wikipedia.org/wiki/Afrika_Selatan), *Mark Shuttleworth*. Tujuan dari distribusi Linux Ubuntu adalah membawa semangat yang terkandung di dalam filosofi Ubuntu ke dalam dunia perangkat lunak (Megawati, 2012).

# BAB III

# METODOLOGI PENELITIAN

## 3.1 Prosedur dan Pengumpulan Data

Dalam penyusunan skripsi ini, penulis membutuhkan data atau informasi dan referensi yang relatif lengkap. Oleh karena itu penulis melakukan *riset* atau penelitian untuk mendapatkan data dan referensi yang diperlukan.

Adapun metode pengumpulan data-data dan informasi yang diperlukan adalah Studi pustaka, yaitu teknik pengumpulan data dengan menghimpun dan menganalisis dokumen. Dokumen yang termasuk didalamnya yaitu penelitian terdahulu, buku, artikel, dan jurnal yang berkaitan dengan objek penelitian. Secara rinci yang penulis lakukan pada studi pustaka ini adalah :

1. Mempelajari dan memahami buku-buku atau dokumen-dokumen dan artikelartikel yang berhubungan dengan teknik mengamankan file menggunakan

Kriptografi.

1. Mempelajari dan memahami penelitian-penelitian sebelumnya tentang enkripsi dan dekripsi *file*/dokumen menggunakan metode kriptografi algoritma SHA-512 dan kriptografi klasik sederhana *Vigenere Cipher*.

## 3.2 Metode Pengembangan Perangkat Lunak

Dalam perancangan sistem ini, penulis menggunakan metode *Rational Unified Proses* (RUP). Pemilihan metode ini dilakukan dengan alasan aplikasi yang akan dikembangkan merupakan aplikasi dengan skala kecil/menengah, serta terfokus pada lingkup tertentu. Dalam metode ini terdapat beberapa tahapan yaitu:

1. *Inception*/ Permulaan

Pada tahap ini penulis menentukan ruang lingkup proyek akan dibuat. Adapun ruang lingkup dari proyek ini adalah sebagai berikut:

* 1. Sebagai aplikasi yang digunakan untuk mengamankan (mengenkripsi dan dekripsi) *file* berekstensi *\*.txt, \*.doc/docx, dan \*.pdf*.
  2. Algoritma yang digunakan dalam proses enkripsi dan mendekripsi *file* adalah algoritma SHA-512 sebagai metode verifikasi kunci dikombinasikan dengan algoritma sederhana *Vigenere cipher* pada *java* untuk mengenkripsi dan dekripsi kembali isi *file* berdasarkan kunci yang digunakan.
  3. Bahasa pemrograman yang digunakan adalah *java.*

1. *Elaboration*/ Elaborasi

Setelah penulis menentukan ruang lingkup proyek, selanjutnya penulis melakukan langkah-langkah berikut : a. Melakukan analisis masalah

* 1. Menggambarkan alur / proses sistem yang akan dibuat
  2. Membuat perancangan aplikasi menggunakan UML (*Unified Modelling Language*). Adapun perancangan aplikasi yang akan dibuat yaitu :
     1. Perancangan *use case diagram*
     2. Perancangan *activity diagram*
     3. Perancangan *sequence diagram*

d. Membuat perancangan tampilan (*interface*) untuk setiap tampilan form/ halaman yang ada dalm system yang akan dibuat.

1. *Contruction* / kontruksi

Pada tahap ini penulis melakukan tahap implementasi yaitu sebagai berikut:

* 1. Melakukan pengkodean (*coding*) menggunakan bahasa Pemrograman

*Java*

* 1. Melakukan proses pengujian sistem untuk mengetahui tingkat akurasi sistem

1. *Transition*/Transisi

Pada tahap ini, penulis melakukan beberapa kegiatan akhir yaitu :

* 1. melakukan beberapa ke melakukan pengujian akhir pada sistem yang telah jadi
  2. Menyerahkan perangkat lunak kepada pemakai, mengujinya ditempat pemakai, dan meperbaiki masalah-masalah yang muncul saat dan setelah pengujian

36

## 3.3 Waktu Penelitian

Adapun waktu penelitian diuraikan pada table 3.1 berikut.

**Tabel**

**3.1**

**Waktu**

**Penelitian**

No

Uraian

Waktu

Nov

Des

Jan

Feb

Maret

April

Mei

Juni

Juli

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1.

*Inception*

2.

*Elaboration*

3.

*Construction*

4.

*Transition*

## 3.4 Tempat Penelitian

Penelitian ini dilakukan di *Laboratorium System Computer And Networking*, Jurusan Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kota Kendari, Sulawesi Tenggara.

### 3.5 Analisis Sistem

Analisis sistem merupakan suatu tahapan yang bertujuan untuk mengetahui dan mengamati apa saja yang terlibat dalam suatu sistem. Pembahasan yang ada pada analisis sistem ini yaitu analisis masalah, analisis kebutuhan fungsional dan analisis kebutuhan *non fungsional*.

#### 3.5.1 Analisis Masalah

Pada peneilitian ini, berdasarkan studi literatur yang telah dilakukan sebelumnya, pencurian data sudah menjadi hal yang sangat lumrah dan sering terjadi. Sehingga dibutuhkan suatu sistem yang bisa mengamankan data itu sendiri, baik itu data yang telah tersimpan ataupun data yang akan di transferkan melalui jaringan komputer. Adapun salah satu serangan pada penyimpanan *file* adalah serangan SSH (*Secure Shell*) dan serangan pada jaringan adalah serangan *MITM (Man In The Middle*). Untuk mengamankan data dari hal tersebut, sehingga dilakukan penelitian ini untuk mengamankan *file* dengan proses enkripsi dan dekripsi file menggunakan metode pengamanan data SHA-512 dan algoritma kriptografi klasik *Vigenere cipher*. Dimana *file* akan dienkripsi dan didekripsi berdasarkan kunci yang digunakan.

#### 3.5.2 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional adalah segala bentuk data yang dibutuhkan oleh sistem, agar sistem yang akan dibangun berjalan sesuai dengan prosedur yang telah ditentukan. Setelah proses analisis telah dilakukan maka akan didapatkan tiga hasil yaitu *input*, proses, dan *output.*

37

##### 3.5.2.1 Analisis Kebutuhan Input

Kebutuhan proses dalam aplikasi ini antara lain:

1. Proses pembuatan aplikasi menggunakan bahasa pemrograman *Java* dengan menggunakan SHA-512 sebagai metode verifikasi kunci dan *chaesar cipher* sebagai metode tambahan untuk mengenkripsi isi *file* berdasarkan kunci yang digunakan.
2. Proses pengujian tingkat keamanan data dengan mengimplementasikan serangan *SSH Brute Force Attack* dan serangan MITM (*Man-In-The-Middle*) menggunakan metode ARP *Spoofing.*

##### 3.5.2.2 Analisis Kebutuhan Output

Output yang akan diperoleh dari sistem ini merupakan sebuah aplikasi berbasis *desktop* yang dapat mengamankan data, melalui proses enkripsi dan dekripsi *file* dengan menggunakan metode SHA-512 sebagai verifikasi kunci dan metode *Vigenere cipher* sederhana untuk memproses isi *file* berdasarkan kunci yang digunakan.

#### 3.5.3 Analisis Kebutuhan *Nonfungsional*

Analisis kebutuhan *nonfungsional* adalah sebuah langkah dimana seorang developer menganalisis sumber daya yang dibutuhkan untuk membangun aplikasi yang akan dibangun. Terdapat dua tahap proses menganalisis kebutuhan nonfungsional, yaitu analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

##### 3.5.3.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang akan digunakan dalam proses membangun aplikasi prediksi sebagai berikut:

**Tabel 4.1 Spesifikasi perangkat keras**

|  |  |  |
| --- | --- | --- |
| No | Nama Perangkat | Spesifikasi |
| 1. | *Processor* | *Intel (R) Coleran (R)* |
| 2. | *Monitor* | *Generic PnP Monitor* |
| 3. | *Memori* | *RAM 2 GB* |

##### 3.5.3.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang akan digunakan dalam proses membangun aplikasi enkripsi dan dekripsi *file* adalah sebagai berikut :

**Tabel 4.2 Spesifikasi perangkat lunak**

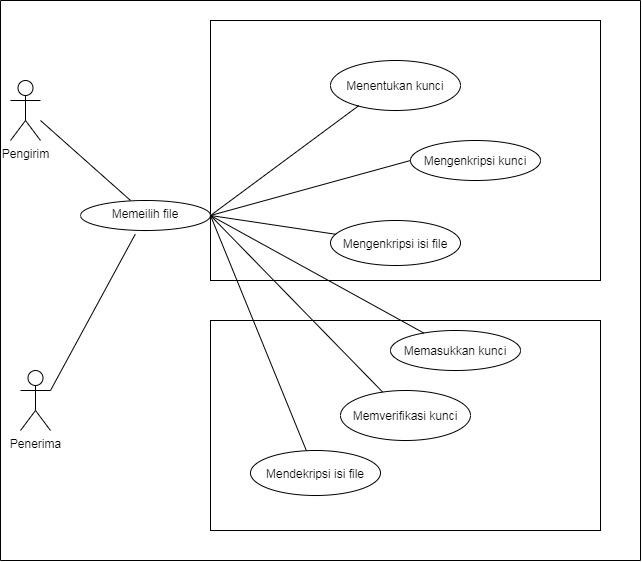
|  |  |  |  |
| --- | --- | --- | --- |
| No | Nama Perangkat |  | Spesifikasi |
| 1. | *Operating System* | *Windows 7* |  |
| 2. | *Linux Ubuntu* | *16.04* |  |
| 3. | *Mozilla Firefox* | 58.0.2 |  |
| 4. | *NetBeans* | 8.2 |  |

### 3.6 Perancangan Sistem

Aplikasi di bangun, dengan menggunakan pemodelan kebutuhan sistem UML yang digunakan antara lain *use case diagram*, *activity diagram*, dan *sequence diagram*.

#### 3.6.1 *Use Case Diagram*

Adapun gambaran *use case* aplikasi yang akan dibangun dapat dilihat pada gambar 3.4 barikut.



## Gambar 3.1 Rancangan *Use case* Diagram Sistem

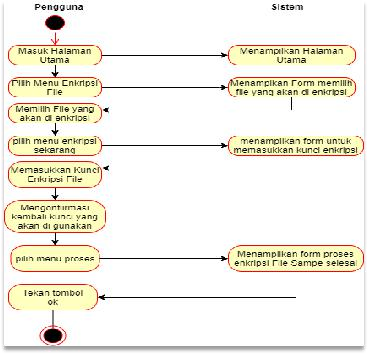
Pada gambar 4.1 terdapat dua aktor yang mempunyai peran masingmasing yaitu pengirim dan penerima. Pengirim dapat memilih *file* yang akan dienkripsi terlebih dahulu, untuk melakukan proses enkripsi maka harus menentukan kunci yang akan digunakan minimal 8 karakter. Kunci asli yang dimasukkan akan dienkripsi menggunakan algoritma SHA-512 sehingga dapat menghasilkan hash kunci sepanjang 512 bit / 64 *byte*, yang kemudian diubah ke*string hexa* dan dikembalikan kebentuk *byte* sepanjang 128 *byte*. Pengirim dapat mengenkripsi isi *file* dengan menggunakan algoritma *Vigenere Cipher* untuk menghasilkan *plainfile/plaintext,*berdasarkan kunci yang digunakan dimana panjang kunci akan diulang secara periodik sebanyak jumlah *plaintext,* hingga semua isi *file* dapat terenkripsi

Penerima dapat memilih *file* terlebih dahulu *file* yang akan didekripsi (dalam *format .enc*), untuk dapat melakukan dekripsi penerima *file* memerlukan kunci asli sama yang digunakan pengirim pada saat mengenkripsi *file* tersebut sebagai verifikasi kunci. Ketika kedua hash sesuai maka proses dekripsi dapat berlangsung dan isi *file* dapat didekripsi dengan menggunakan algoritma *Vigenere cipher* berdsarakan kunci yang digunakan. sehingga file dapat kembali seperti semula.

### 3.6.2 *Activity Diagram*

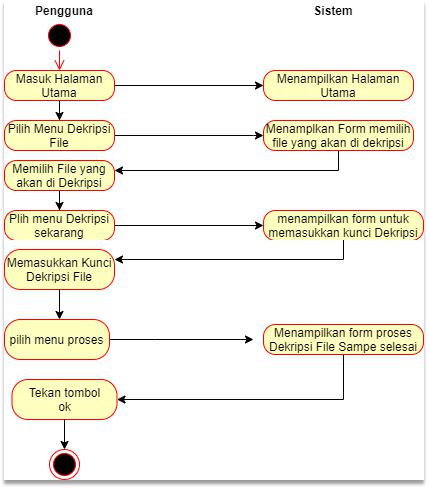
*Activity diagram* meggambarkan berbagai alur aktivitas yang ada didalam sistem yang sedang dirancang dan bagaimana masing-masing alur yang ada berawal serta berakhir. adapun *Activity diagram* dari sistem yang akan dibangun dapat dilihat pada gambar 4.2 dan 4.3.

a. Activity Enkripsi File



### Gambar 3.2 *Activity Diagram* Enkripsi *File*

Pada gambar 4.2 diatas menggambarkan bahwa pada proses awal pengguna akan masuk kehalaman utama, dan rekasi sistem akan menampilkan halaman utama. Pada halaman utama tersebut, pengguna harus memilih menu enkripsi *file*, dan reaksi sistem pada saat itu akan menampilkan halaman *form* pilih *file* yang akan di enkripsi pada *directory* penyimpanan, yang ada di komputer atau pc pengguna. Setelah *file* ditemukan, pengguna harus memilih menu enkripsi sekarang, dan sistem akan menampilkan *form* untuk memasukkan kunci dan mengonfirmasi kembali kunci. Langkah terakhir adalah mengklik tombol proses dan sistem akan menampilkan form proses enkripsi sampai selesai. b. Activity dekripsi *file*



### Gambar 3.3 *Activity Diagram* Dekripsi *File*

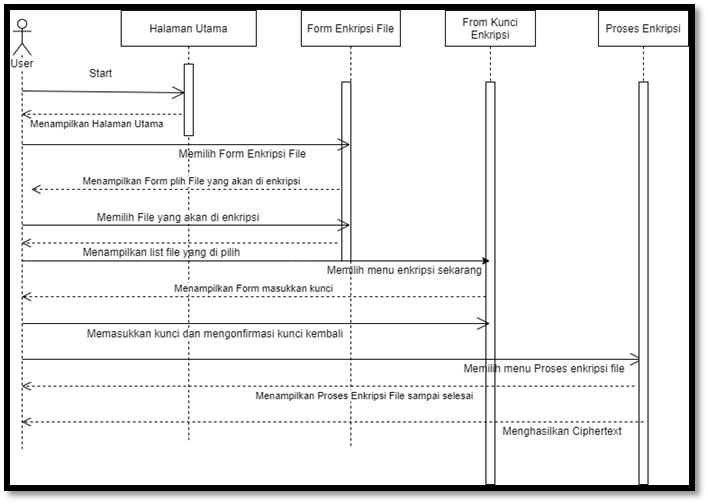
Pada gambar 4.3 diatas menggambarkan bahwa pada proses awal pengguna akan masuk kehalaman utama, dan reaksi sistem akan menampilkan halaman utama. Pada halaman utama tersebut, pengguna harus memilih menu dekripsi *file*, dan reaksi sistem pada saat itu akan menampilkan halaman *form* pilih file yang akan di dekripsi pada *directory* penyimpanan, yang ada di komputer atau pc pengguna. Setelah *file* ditemukan, pengguna harus memilih menu dekripsi sekarang, dan sistem akan menampilkan *form* untuk memasukkan kunci dan mengonfirmasi kembali kunci. Langkah terakhir adalah mengklik tombol proses dan sistem akan menampilkan *form* proses dekripsi sampai selesai.

#### 3.6.3 *Sequence Diagram*

*Sequence diagram* berfungsi untuk menggambarkan rangkaian pesan yang akan dikirim antara objek yang ada, serta interaksi yang terjadi antara objek.

Berikut *sequence diagram* yang dirancang dapat dilihat pada gambar 4.4 dan 4.5. a. *Sequence diagram* enkripsi *file*

Rancangan *sequence diagram* enkripsi file terdapat pada gambar 4.4.

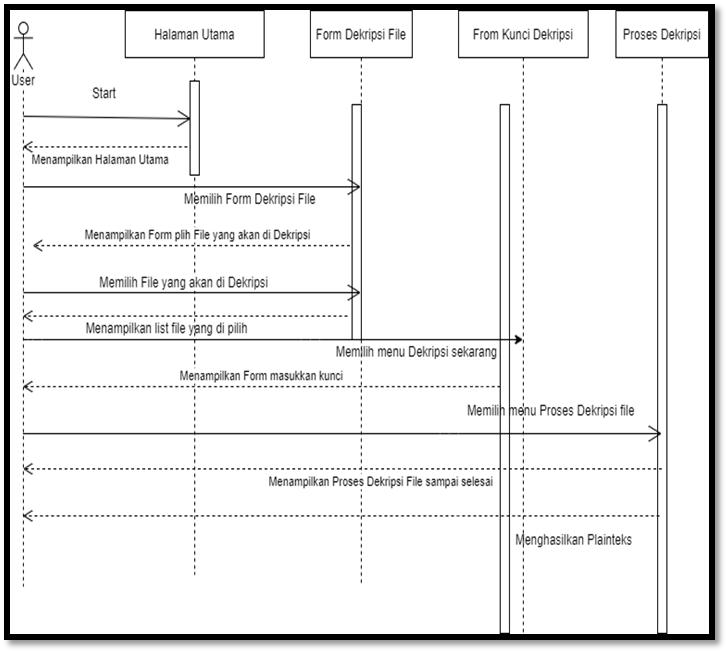


### Gambar 3.4 *Sequence Diagram* Enkripsi *File*

Pada gambar 4.4 diatas terdapat empat objek utama yaitu halaman utama, form enkripsi dan form memasukkan kunci, dan proses enkripsi. Proses awal yang dilakukan *use*r adalah memulai jalannya sistem sehinga *user* dapat masuk kehalaman utama. Setelah *user* berada di halaman uatama sistem, *user* harus memilih tab menu enkripsi *file* dan sistem akan menampilkan halaman form enkripsi *file*. Didalam *form* enkripsi, *user* harus melakukan beberapa langkah sehingga dapat melakukan proses enkripsi *file* yang diantaranya mengunggah *file* yang akan di enkripsi, memasukkan kunci, mengonfirmasi kunci yang ada, dan menekan tombol proses. Hasil dari proses enkripsi ini adalah *file* dalam bentuk *ciphertext.*

b. *Sequence diagram* dekripsi *file*

Adapun rancangan *Sequence diagram* dekripsi *file* pada sistem yang akan dibangun dapat dilihat pada gambar 4.5.



### Gambar 3.5 *Sequence Diagram* Dekripsi *File*

Pada gambar 4.5 *sequence diagram* proses dekripsi *file* tidak jauh berbeda dengan jalannya enkripsi *file*. Proses awal yang dilakukan *user* adalah memulai jalannya sistem sehinga *user* dapat masuk kehalaman utama. Setelah *user* berada di halaman uatama sistem, *user* harus memilih tab menu dekripsi *file* dan sistem akan menampilkan halaman *form* dekripsi *file*. Didalam form dekripsi, *user* harus melakukan beberapa langkah sehingga dapat melakukan proses dekripsi *file* yang diantaranya mengunggah *file* yang akan di dekripsi, memasukkan kunci yang sama dengan proses enkripsi, dan menekan tombol proses. Hasil dari proses dekripsi ini adalah *file* dalam bentuk *plainteks* atau teks asli kembali*.*

#### 3.7 Rancangan *Interface*

Rancangan *interface* menampilkan penjelasan sistem dan memberikan panduan pemakaian sistem secara menyeluruh/ *step by step* sehingga pengguna mengerti apa yang akan dilakukan terhadap suatu sistem.

##### 3.7.1 Halaman Utama Sistem

Adapun tampilan halaman utama sistem yang akan dibangun dapat dilihat pada gambar 4.6 dibawah ini.

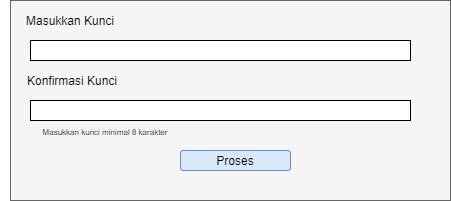


**Gambar 3.6 Tampilan Halaman Utama**

Pada gambar 4.6 halaman utama sistem terdiri dari beberapa menu yaitu menu enkripsi *file,* dekripsi *file*. Menu enkripsi *file* berfungsi untuk menampilkan *form* enkripsi, begitu pula menu dekripsi *file* berfungsi menampilkan *form* dekripsi *file*.

##### 3.7.2 Form Memasukkan Kunci Enkripsi *File*

Adapun rancangan tampilan form untuk memasukkan kunci enkripsi *file* dapat dilihat pada gambar 4.7.

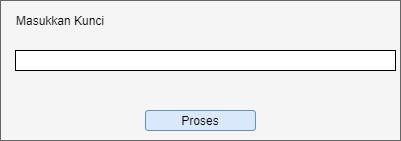


## Gambar 4.7 Tampilan Form Memasukkan Kunci Enkripsi *File*

Pada tampilan 4.7 pengguna harus memasukkan kunci enkripsi *file* terlebih dahulu untuk dapat melakukan enkripsi *file,* dan menggonfirmasinya kembali dengan kunci yang sama.

### 3.7.3 Form Untuk Memasukkan Kunci Dekripsi *File*

Adapun tampilan form untuk memasukkan kunci dekripsi kembali *file* dapat dilihat pada gambar di bawah ini:

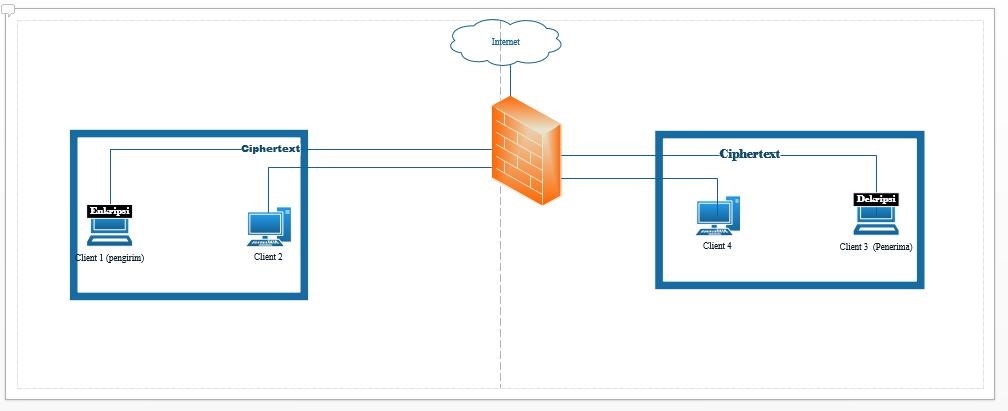


## Gambar 4.8 Tampilan Form Memasukkan Kunci Dekripsi *File*

Pada gambar 4.8, pengguna harus memasukkan kunci yang sama dengan kunci enkripsi *file* untuk dapat membaca *file* kembali.

### 3.8 Rancangan Topologi Jaringan

Adapun rancangan topologi jaringan dari proses pertukaran data antara pengirim dan penerima dalam bpenelitian ini dapat dilihat pada gambar 4.9



**Gambar 4.9 Topologi Jaringan**

Psada gambar 4.9 diatas, topologi jaringan yang digunakan adalah topologi *Star*. Dari gambar tersebut dapat diuraikan bahwa : *client* pertama (pengirim) ingin mengirimkan data ke *client* 3 (penerima)dengan syarat harus konek kejaringan yang sama. Pengirim melakukan enkripsi terlebih dahulu yaitu mengubah data asli (*plaintext*), menjadi data rahasia (*ciphertext*). *Ciphertext* itulah yang dikirim ke penerima. Penerima akan menerima data dalam bentuk *ciphertext*, dan akan melakukan proses dekripsi data tersebut agar dapat membaca kembali data asli (*plaintext*).

# DAFTAR PUSTAKA

Agustin Setyo Wardani. (2018, Oktober) Liputan 6.com. [Online].

[https://www.liputan6.com/tekno/read/3665291/45-miliar-data](https://www.liputan6.com/tekno/read/3665291/45-miliar-data%20dicuri-selama-6-bulan-pertama-2018) [dicuri-selama](https://www.liputan6.com/tekno/read/3665291/45-miliar-data%20dicuri-selama-6-bulan-pertama-2018)6-bulan-pertama-2018

Alam, A.R., 2016, Implementasi QR Code dan Algoritma Triple Data Encryption Standard pada Pengamanan Keaslian Dokumen dari

Serangan Pembajakan (Man-In-The-Middle), *Skripsi*, Fakultas Teknik, Universitas Halu Oleo, Kendari

Aristo, A., Sinlae, J., Ngaga, E., Daeng, S., & Mau, B. (2018). RANCANG

BANGUN KRIPTOSISTEM UNTUK PENGAMANAN DATA-DATA

DIGITAL, 5(1), 64–75.

Ariyanto, A., & Asmunin. (2018). Deteksi Paet Sniffing Pada Wirelles Menggunakan ARP Watch. *Jurnal Manajemen Informatika*, *8*(2), 178–181.

Bay Haqi. (2017). Membuat Aplikasi Anteran Dengan JAVA NETBEANS IDE 8.0.2 dan, DATABASE MYSQL, 1–21.

Fridayanthie, C. “Rancang Bangun Sistem Informasi Simpan Pinjam Karyawan Menggunakan Metode Object Oriented Programming (Studi Kasus: Pt. Arta Buana Sakti Tangerang),” *J. Techno Nusa Mandiri*, vol. 13, no. 2, pp. 63–71, 2016.

D. Nurnaningsih and A. A. Permana, “Rancangan Aplikasi Pengamanan Data Dengan Algoritma Advanced Encyption Standard (Aes),” J. Tek. Inform., vol. 11, no. 2, pp. 177–186, 2018, doi: 10.15408/jti.v11i2.7811.Aguk Nur Anggraini, M. (2018). Uji Fitur Intrusion Prevention Pada Firewall Untangle Dengan Pengujian Dos Dan Ssh Brute Force. Jurnal Manajemen Informatika, 9(1), 18–25.

Fauzi, A., & Suartana, I. (2018). Monitoring Jaringan Wireless Terhadap

Serangan Packet Sniffing Dengan Menggunakan Ids. Jurnal Manajemen

Informatika, 8(2).

Fauzi, A. (2019). Analisa Kombinasi Pesan Teks Ke Dalam File Audio Memanfaatkan Algoritma Data Encryption Standard Dan Metode End of File. *Jurnal Teknik Informatika Kaputama (JTIK)*, *3*(1), 1–8.

Irawan, R., Ilhamsyah, Brianorman,Y. (2015). Aplikasi Enkripsi Dan Dekripsi

Pesan Singkat Menggunakan Algoritma Knapsack Berbasis Android.

Cybrarians Journal, 03(37), 1–31. [https://doi.org/10.12816/0013114.](https://doi.org/10.12816/0013114)

Irawan,C., dkk., (2019). Studi, P., Informatika, T., Komputer, F. I., & Nuswantoro,

U. D.. IMPLEMENTASI ALGORITMA AUTOKEY CIPHER, 978–979.

James, A. A. (2020). A Novel High-Speed IoT Based Crypto Lock Using AES128 and SHA-512, 9–15.

M. Aguk Nur Anggraini, “Uji Fitur Intrusion Prevention Pada Firewall Untangle Dengan Pengujian Dos Dan Ssh Brute Force,” *J. Manaj. Inform.*, vol. 9, no. 1, pp. 18–25, 2018.

Manullang, D. I. (2018). Perancangan Aplikasi Penyandian File Teks dengan Algoritma Bifid Cipher. *Pelita Informatika Budi Darma*, *17*, 65–70.

Megawati, C. (2012). Implementasi dan Analisa Unjuk Kerja Sistem Keamanan Jaringan Wireless Berbasis Linux Platform dan DD-WRT Firmware, *Skripsi*, Universitas Indonesia, Depok.

Mohidul Islam, S. M., Hossin, A., Shah, R. K., & Bipin, P. K. (2017). Bit Adjusting Image Steganography in Blue Channel using AES and Secured

Hash Function. *International Journal of Computer Science and Mobile*

*Computing*, *611*(11), 25–30. Retrieved from

<http://ijcsmc.com/docs/papers/November2017/V6I11201715.pdf>

Nugroho, B., (2013). *Instalasi dan Konfigurasi topologi jaringan*, Yogyakarta: ANDI Yogyakarta

Refialy, L., Sediyono, E., & Setiawan, A. (2015). Pengamanan Sertifikat Tanah Digital menggunakan Digital Signature SHA-512 dan RSA. *Jurnal Teknik*

*Informatika Dan Sistem Informasi*, *1*(3), 229–234. [https://doi.org/10.28932/jutisi.v1i3.400.](https://doi.org/10.28932/jutisi.v1i3.400)

Risqiwati, D., Irawan, E. A., Teknik, F., Studi, P., Informatika, T., & Malang, U. M. (2018). Realtime Pencegahan Serangan Brute Force dan DDOS Pada Ubuntu Server, 17(4), 347–354.

Sandyansyah, I. (2018). Implementasi Kriptografi Skipjack dan Steganografi LSB pada Citra Digital.

Sakti, B., & Aziz, A. (2013). Uji Kelayakan Implementasi SSH sebagai Pengaman FTP Server dengan Penetration Testing, 2(1).

Sembiring, H., Utara, S., Manik, F. Y., Utara, S., & Utara, S. (2019). Penerapan Algoritma Secure Hash Algorithm ( SHA ) Keamanan Pada Citra, *4*(1), 33–

36.

Setiawan, W. (2011). Analisis dan Perbandingan Algoritma Whirlpool dan SHA-

512 sebagai Fungsi Hash. Makalah IF3058 Kriptografi – Sem. II Tahun 2010/2011.

Sinaga, D., Jatmoko, C., Studi, P., Infromatika, T., Komputer, F. I., & Nuswantoro, U. D. (2018). Implementasi sha512 pada aplikasi kriptografi file 1,2, 978–979.

Sofana, I. (2013). *Membangun Jaringan Komputer*, Bandung: Informatika Bandung.

Suanda,R. (2018). Pengamanan File Teks Menggunakan Algoritma Skipjack dan Kompresi Algoritma Taboo.

Suendri. (2018). Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan Sistem Informasi Remunerasi Dosen Dengan Database Oracle (Studi Kasus: UIN Sumatera Utara Medan). Jurnal Ilmu Komputer Dan Informatika, 3(1), 1–9.

Sumandri, “Studi Model Algoritma Kriptografi Klasik dan Modern,” *Semin. Mat. dan Pendidik. Mat. UNY*, pp. 265–272, 2017.

Supratman, L. P., Wahyudin, A., 2017, Penggunaan Media Sosial oleh Digital

Native International Journal of English Literature and Social Sciences, 2(5): 51-58

W. Setiawan, “Analisis dan Perbandingan Algoritma Whirlpool dan SHA- 512 sebagai Fungsi Hash,” *Makal. IF3058 Kriptografi – Sem. II Tahun 2010/2011*, 2011.

Wandani, K. D., & Sinurat, S. (2018). Implementasi Secure Hash Algoritma Untuk Pengamanan Pada File Video, *13*, 165–168.

Zulkifli, Z. (2018). Rancang Bangun Website E-Learning Dengan Pemodelan Uml (Studi Kasus Di Stkip Muhammadiyah Muara Bungo). INTECOMS: Journal of Information Technology and Computer Science, 1(2), 159–167. http://doi.org/10.31539/intecoms.v1i2.291

# LAMPIRAN

Hitungan Manual kunci SHA 512: nurhazmila

1. ubah pesan dalam bentuk biner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n =  01101110 | u =  01110101 | r = 01110010 | h = 01101000 | a = 01100001 |
| z =  01111010 | m =  01101101 | i = 01101001 | l = 01101100 | a = 01100001 |

1. Menambahkan bit 1 diakhir

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 01101110 | 01110101 | 01110010 | 01101000 | 01100001 |
| 01111010 | 01101101 | 01101001 | 01101100 | 01100001 |
| 1 |  |  |  |  |

1. Menambahkan padding hingga mencapai 1024 bit terdiri dari 0-79 putarann

i + 1 + k = 896 mod 1024 i = 10 x 5 = 50

= 50 x 8 = 400 i + 1 + k = 896 mod 1024

* 1. + 1 + k = 896 mod 1024
  2. + k = 896 mod 1024

K = 896 – 401

= 495

1. Menambahkan 495 bit ‘0’ sehingga dapat mencapai 1024 bit.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0110111  0 | 0111010  1 | 0111001  0 | 0110100  0 | 0110000  1 | 0111101  0 | 0110110  1 | 0110100  1 |
| 0110110  0 | 0110000  1 | 1000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |
| 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 | 0000000  0 |

5.Menambahkan bit akhir sebanyak bit kunci yang dimasukkan “nurhazmila”, sehingga memenuhi 1024 bit

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0110111 |  |  | 0111010 |  |  | 0111001 |  |  | 0110100 |  | |  | 0110000 |  |  | 0111101 |  |  | 0110110 |  |  | 0110100 |  |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
|  | 0110110 |  |  | 0110000 |  |  | 1000000 |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0 | 1 | 0 |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | | |
| 0000000  0 | |  | 0000000  0 | |  | 0000000  0 | |  | 000000  00 | |  | 00000000 | | |  | 0000000  0 | |  |  | 0000000 |  |  | 1001000 |  |
| 1 | 0 |

1. persipan parsing pesan dengan M0-M15.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M0 = | 01101110 | |  |  | 01110101 |  |  | 01110010 |  |  | 01101000 |  | 01100001 | 01111010 |  | 01101101 |  |  | 0110100 |  |
|  |  |  |  | 1 |
| M1 = |  | 01101100 |  |  | 01100001 |  |  | 10000000 |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
|  |  |  |
| M2 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M3 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M4 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M5 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M6 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M7 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M8 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M9 = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M10  = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M11  = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M12  = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M13  = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M14  = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 | 00000000 | |  | 0000000  0 | | |
| M15  = | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | |  | 00000000 | 00000000 |  | 00000001 |  |  | 1001000 |  |
|  | 0 |

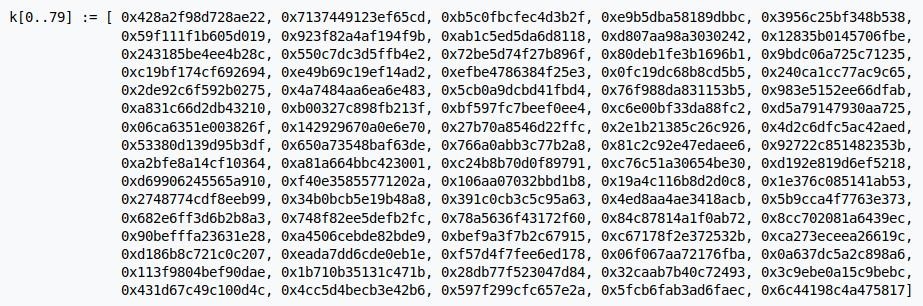
1. Konversi M0-M15 ke hexadecimal

|  |
| --- |
| M0 = 6e757268617a6d69 |
| M1 = 6c61800000000000 |
| M2 = 0000000000000000 |
| M3 = 0000000000000000 |
| M4 = 0000000000000000 |
| M5 = 0000000000000000 |
| M6 = 0000000000000000 |
| M7 = 0000000000000000 |
| M8 = 0000000000000000 |
| M9 = 0000000000000000 |
| M10 = 0000000000000000 |
| M11 = 0000000000000000 |
| M12 = 0000000000000000 |
| M13 = 0000000000000000 |
| M14 = 0000000000000000 |
| M15 = 0000000000000190 |

1. Ketetapan hash

|  |
| --- |
| a = 0x6a09e667f3bcc908 |
| b = 0xbb67ae8584caa73b |
| c = 0x3c6ef372fe94f82b |
| d = 0xa54ff53a5f1d36f1 |
| e = 0x510e527fade682d1 |
| f = 0x9b05688c2b3e6c1f |
| g = 0x1f83d9abfb41bd6b |
| h = 0x5be0cd19137e2179 |

1. Ketetapan nilai konstanta = k



1. Mempersiapkan jadwal pesan SHA-512 menggunakan jadwal pesan diberi label W0, W1, ..., W79.

|  |
| --- |
| **w0 =6e757268617a6d69** |
| **w1 =6c61800000000000** |
| **w2 =0000000000000000** |
| **w3 =0000000000000000** |
| **w4 =0000000000000000** |
| **w5 =0000000000000000** |
| **w6 =0000000000000000** |
| **w7 =0000000000000000** |
| **w8 =0000000000000000** |
| **w9 =0000000000000000** |
| **w10= 0000000000000000** |
| **w11= 0000000000000000** |
| **w12= 0000000000000000** |
| **w13= 0000000000000000** |
| **w14= 0000000000000000** |
| **w15= 0000000000000190** |

1. Untuk Mencari nilai w16 – 79 menggunakan rumus dibawah ini :

rumus : wt = 1 (512)( −2) + −7 + 0 (512)( −15) + −16, 16 ≤ ≤

79 .

w16 = 1 (w16-2) + (w16-7) + 0 (w16-15) + (w16-16)

1 (w16-2 = (w16-2) RotR 19 RotR61  SHR6

0 (w16-15)= (w16-2) RotR 1 RotR8  SHR7

dimana :

1(w16-2) = (w16-2) RotR 19  RotR61  SHR6 w16-2 = w14

= 000000000000000

=0000000000000000000000000000000000000000000000000000000000000000

ROTR 19 =0000000000000000000000000000000000000000000000000000000000000000

ROTR 61 = 0000000000000000000000000000000000000000000000000000000000000000

SHR 6 = 0000000000000000000000000000000000000000000000000000000000000000

1(w16-2) = 0000000000000000000000000000000000000000000000000000000000000000 Jadi 1(w16-2) = 000000000000000

0(w16-15) = (w16-15) RotR 1  RotR8  SHR7 w16-2 = w1

= 6c61800000000000

=0110110001100001100000000000000000000000000000000000000000000000

ROTR 1 = 0011011000110000110000000000000000000000000000000000000000000000 ROTR 8 = 0000000001101100011000011000000000000000000000000000000000000000

SHR 7 = 0000000011011000110000110000000000000000000000000000000000000000

0(w16-15) = 0011011010000100001000101000000000000000000000000000000000000

= 3684228000000000

(w16-7) = w9

= 0000000000000000 (W16-16) = w0

W0 = 6e757268617a6d69

= 01101110 01110101011100100110100001100001011110100110110101101001

W16 = 1 (w16-2) + (w16-7) + 0 (w16-15) + (w16-16)

= 000000000000000 + 0000000000000000+3684228000000000 +

6e757268617a6d69

W16 =3cae5878795d8d69

Adapun hasil penjadwalan pesan adalah sebagai berikut :

|  |  |  |  |
| --- | --- | --- | --- |
| **Wo=6e757268617a6d69** | **W20=dbabcbeed4d38** | **W40=19ce6bd97969eaf8b** | **W60=d8054b1e14ad2202** |
| **w1=6c61800000000000** | **W21=d8c361b8bcbcc** | **W41=1036b499e17e323cb** | **W61=40d8054b1e14ad00** |
| **w2=0000000000000000** | **W22=197ea3d2b4d** | **W42=bb4cb53eeb7ea000** | **W62=866d46dd1b754e00** |
| **W3=0000000000000000** | **W23=6dbf19c806c709c7** | **W43=c1be0a3a45bbc0c8** | **W63=cfab5138a9785175** |
| **W4=0000000000000000** | **W24=6c61bc51b64d0ea8** | **W44=70d10dc95a21f97b** | **W64=12ef35dc271f40d0** |
| **W5=0000000000000000** | **W25=524da120b8341d6** | **W45=33c81ac764e6ddf5** | **W65=52ef35dc271f4040** |
| **W6=0000000000000000** | **W26=514b029566f47de** | **W46=c97c2de7974aba7c** | **W66=33c81ac764e6dd80** |
| **W7=0000000000000000** | **W27=15f18de18251f6** | **W47=c81fd33a28227a80** | **W67=a333c81ac764e6dd** |
| **W8=0000000000000000** | **W28=21d0eec2f18a9e** | **W48=b908026636458e00** | **W68=74bf6e1c4c49cfa9** |
| **W9=0000000000000000** | **W29=272b227547ea** | **W49=eb2884a16172a70b** | **W69=74bf6e1c4c49cf00** |
| **W10=0000000000000000** | **W30=76a0e307342df5f5** | **W50=2f5ca1f0d86d4534** | **W70=43e23c4a5202fd40** |
| **W11=0000000000000000** | **W31=8cab8cf78254bcba** | **W51=74bf6e1c4c49cf21** | **W71=8f728a8e53a45bbc** |
| **W12=0000000000000000** | **W32=9a31e5bcb628597b** | **W52=cf8b513829785175** | **W72=74a16e1c4c49cf21** |
| **W13=0000000000000000** | **W33=c45c2ba5d5b11c3e** | **W53=866d46dd1b754ec0** | **W73=74bf6e1c4c49c021** |
| **W14=0000000000000000** | **W34=7fa0c86227a2ab46** | **W54=11eb172ed451f880** | **W74=162974cce6f10202** |
| **W15=0000000000000190** | **W35=22981896c5de26fc** | **W55=318622954a9a6d1b** | **W75=40d8054b1e14ad22** |
| **W16=3cae5878795d8d69** | **W36=350710f64fb4923d** | **W56=12ef35dc271f4780** | **W76=10edb5dc271f40c8** |
| **W17=bc61800000000006** | **W37=a938ef9bbe33d742** | **W57=12ef35dc271f52c8** | **W77=12eff5fc271f40c8** |
| **W18=36de76139b7381** | **W38=c086763018a40cd8** | **W58=10edb5dc271f40c8** | **W78=f2ef35dc271f47c8** |
| **W19=36317186000c31c** | **W39=148186a9cb7261197** | **W5=12eff5fc271f40c8** | **W79=12eff5fc271f40c8** |

12. Inisialisasi variabel kerja . Selanjutnya melakukan inisialisasi variabel kerja *a, b, c, d, e, f, g* dan *h* dimana setiap variabel diambil dari *initial hash value a=H0(0), b=H1(0), c=H2(0) , d=H3(0), e=H4(0), f=H5(0), g=H6(0), h=H7(0).* Selanjutnya dilakukan proses komputasi fungsi *hash* SHA-512 dari t=0 sampai t=79.

Untuk *t=0* lakukan perhitungan sebagai berikut :

## *a b c d e f G h*

6a09e66 bb67ae8 3c6ef372 a54ff53 0510e52 9b05688 1f83d9a 5be0cd1

7f3bcc9 584caa7 fe94f82b a5f1d36 7fade68 c2b3e6c bfb41bd 9137e21

08 3b f1 2d1 1f 6b 79

a = T1+T2

1=ℎ+ Σ( )(512)1+ ℎ( , , )+ +

ℎ = 5be0cd19137e2179

Σ( )(512)1 =( 6)⊕( 11)⊕( 25)

( 6) =( 6) = ((0510e527fade682d1) 6)

=01010001 00001110 01010010 01111111 10101101 11100110 10000010 11010001 =10001001 01000100 00111001 01001001 11111110 10110111 10011010 00001011

( 11) =( 11) = ((0510e527fade682d1) 11)

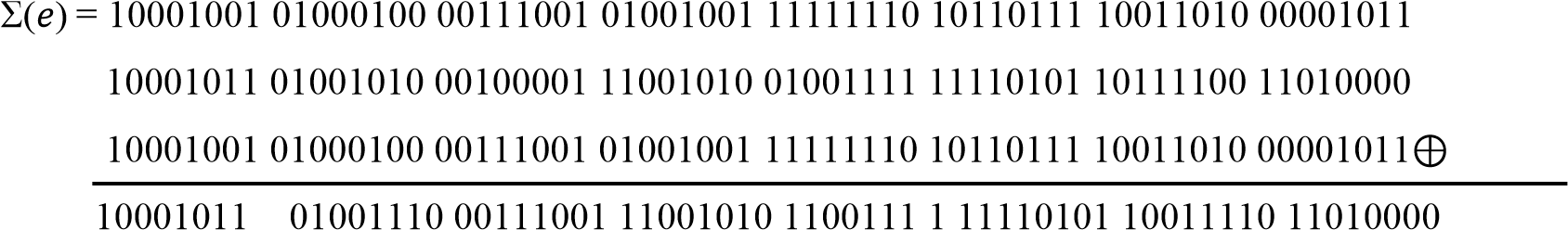
=01010001 00001110 01010010 01111111 10101101 11100110 10000010 11010001

=10001011 01001010 00100001 11001010 01001111 11110101 10111100 11010000

( 25) =( 25) = ((0510e527fade682d1) 25)

= 01010001 00001110 01010010 01111111 10101101 11100110 10000010 11010001

= 10001001 01000100 00111001 01001001 11111110 10110111 10011010 00001011



= 8b4e39cacff59ed0

ℎ( , , ) =( ^ )⊕(~ ^ )

= (0510e527fade682d1 ^9b05688c2b3e6c1f) ⊕ (~0510e527fade682d1^1f83d9abfb41bd6b)

( ^ ) = 01010001 00001110 01010010 01111111 10101101 11100110 10000010 11010001

10011011 00000101 01101000 10001100 00101011 00111110 01101100 00011111

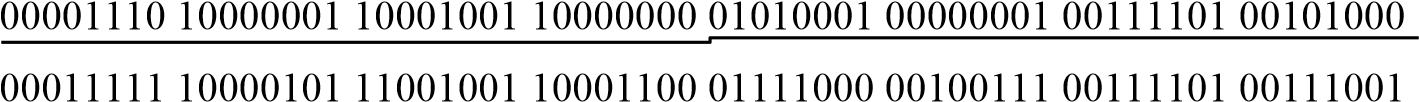
00010001 00000100 01000000 00001100 00101001 00101010 00000000 00010001

(~ ^ ) = 10101110 11110001 10101101 10000000 01010010 00011001 01111101 00101110

### 00011111 10000011 11011001 10101011 11111011 01000001 10111101 01101000

= 00001110 10000001 10001001 10000000 01010001 00000001 00111101 00101000

ℎ( , , ) = 00010001 00000100 01000000 00001100 00101001 00101010 00000000 00010001



= 1f858c78273d39

Kt (K0) = 428a2f98d728ae22

Wt (wo) = 6e757268617a6d69

T1 = ℎ + Σ( )(256)1+ ℎ ( , , )+ +

= 5be0cd19137e2179 + 8b4e39cacff59ed0+1f858c78273d39+428a2f98d728ae22 +

6e757268617a6d69

= 1964f2b0fc71e934a

T2 = Σ( )+প( , , )

Σ( ) =( 2)⊕ ( 13)⊕( 22)

( 2)=( 2) =((6a09e667f3bcc908) 2)

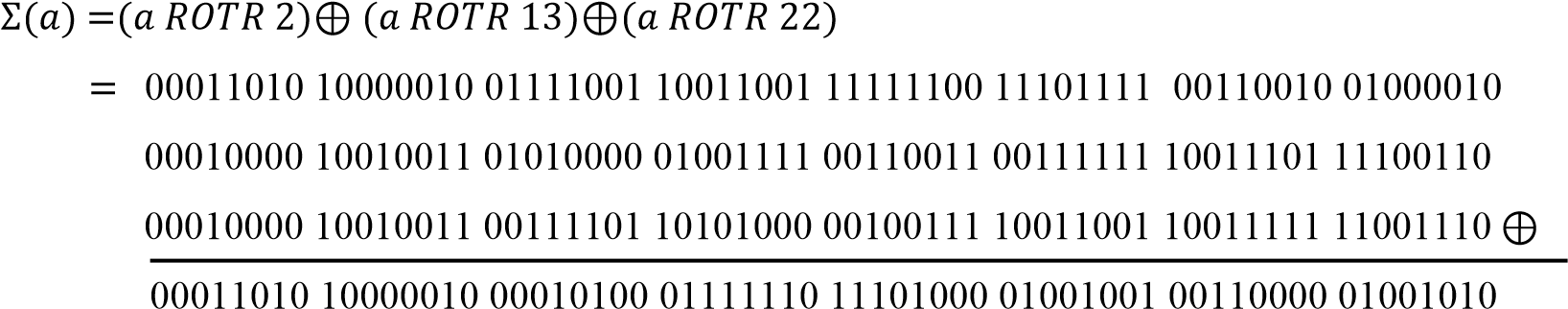
=01101010 00001001 11100110 01100111 11110011 10111100 11001001 00001000 = 00011010 10000010 01111001 10011001 11111100 11101111 00110010 01000010

( 13)= 01101010 00001001 11100110 01100111 11110011 10111100 11001001 00001000 = 00010000 10010011 01010000 01001111 00110011 00111111 10011101 11100110

( 22)=( 22) = ((6a09e667f3bcc908) 22)

= 01101010 00001001 11100110 01100111 11110011 10111100 11001001 00001000

=00010000 10010011 00111101 10101000 00100111 10011001 10011111 11001110



= 1a82147ee849304a

প( , , )=( ^ ) ⊕ ( ^ ) ⊕ ( ^ )

( , , ) = (6a09e667f3bcc908^bb67ae8584caa73b) ⊕

(6a09e667f3bcc908^3c6ef372fe94f82b) ⊕ (bb67ae8584caa73b ^ bb67ae8584caa73b)

( ^ ) = (6a09e667f3bcc908^bb67ae8584caa73b )

= 01101010 00001001 11100110 01100111 11110011 10111100 11001001 00001000

10111011 01100111 10101110 10000101 10000100 11001010 10100111 00111011 ^

00101010 00000001 10100110 00000101 10000000 10001000 10000001 00001000

( ^ ) = (6a09e667f3bcc908^3c6ef372fe94f82b)

= 01101010 00001001 11100110 01100111 11110011 10111100 11001001 00001000

00111100 01101110 11110011 01110010 11111110 10010100 11111000 00101011 ^

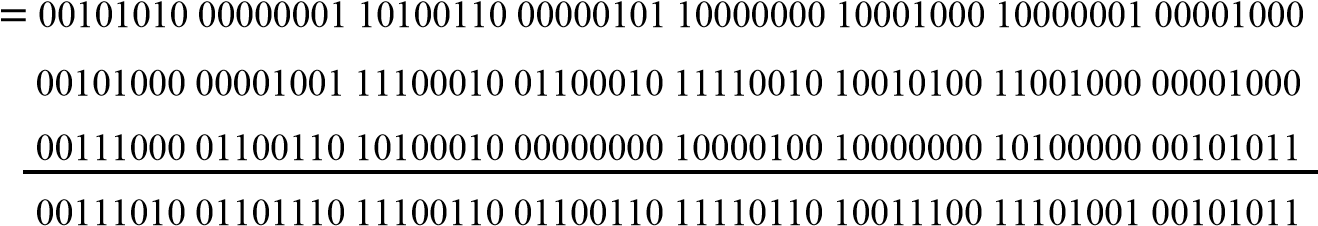
00101000 00001001 11100010 01100010 11110010 10010100 11001000 00001000

( ^ )= 10111011 01100111 10101110 10000101 10000100 11001010 10100111 00111011

00111100 01101110 11110011 01110010 11111110 10010100 11111000 00101011

00111000 01100110 10100010 00000000 10000100 10000000 10100000 00101011

প( , , ) = ( ^ ) ⊕ ( ^ ) ⊕ ( ^ )



= 3a6ee666f69ce92b

|  |  |
| --- | --- |
| T2 | = Σ( )+প( , , )  = 1a82147ee849304a + 3a6ee666f69ce92b  = e8f47da2df01d195 |
| T0(a) | = T1 + T2  = 1964f2b0fc71e934a + e8f47da2df01d195  = 27F43b0e21fedf4df |

b = c ; c = d ; e = d+T1 ; f = e ; e = g ; g = h;

Berikut hasil kerja secara keseluruhan :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **c** | **d** | **e** | **f** | **G** | **h** |
| Init | 6a09e667 f3bcc908 | 0xbb67ae8 584caa73b | 3c6ef372 fe94f82b | a54ff53a  5f1d36f1 | 510e527 fade682d1 | 9b05688 c2b3e6c1f | 1f83d9ab fb41bd6b | 5be0cd19  137e2179 |
| T0 | 27F433b0e | 6a09e667 | bb67ae85 | 3c6ef372 | 23b9f204a | 510e527 | 9b05688 | 1f83d9ab |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 21fedf4df | f3bcc908 | 84caa73b | fe94f82b | 263bd13b | fade682d1 | c2b3e6c1f | fb41bd6b |
| T1 | 6733266 7ffc00b31 | 27F433b0e  21fedf4df | 6a09e667 f3bcc908 | bb67ae85  84caa73b | 152fecd8 f70e5939 | 23b9f204a  263bd13b | 510e527 fade682d1 | 9b05688 c2b3e6c1f |
| T2 | 47b5481 dbefa4fa4 | 6733266 7ffc00b31 | 27433b0 e21fedf4df | 6a09e667 f3bcc908 | 9159015a  3070dd17 | 152fecd8 f70e5939 | 23b9f204a  263bd13b | 510e527 fade682d1 |
| T3 | db0c2e0 d64f98fa7 | 47b5481 dbefa4fa4 | 6733266 7ffc00b31 | 27433b0e  21fedf4df | 629a292a  367cd507 | 9159015a  3070dd17 | 152fecd8 f70e5939 | 23b9f204a  263bd13b |
| T4 | 8eb44a87  68581511 | db0c2e0 d64f98fa7 | 47b5481 dbefa4fa4 | 6733266 7ffc00b31 | cbbb9d5d c1059ed8 | 629a292a  367cd507 | 9159015a  3070dd17 | 152fecd8 f70e5939 |
| T5 | 00091f46  0be46c52 | 8eb44a87  68581511 | db0c2e0 d64f98fa7 | 47b5481 dbefa4fa4 | d0403c39  8fc40002 | cbbb9d5d c1059ed8 | 629a292a  367cd507 | 9159015a  3070dd17 |
| T6 | aca8e67c c3b49c2a | 00091f46  0be46c52 | 8eb44a87  68581511 | db0c2e0 d64f98fa7 | b5dd518b  7d1b2008 | d0403c39  8fc40002 | cbbb9d5d c1059ed8 | 629a292a  367cd507 |
| T7 | 778f9a3c ca642f19 | aca8e67c c3b49c2a | 00091f46  0be46c52 | 8eb44a87  68581511 | d01062f0  2e084000 | b5dd518b  7d1b2008 | d0403c39  8fc40002 | cbbb9d5d c1059ed8 |
| T8 | 1e6457e7 b074dd17 | 778f9a3c ca642f19 | aca8e67c c3b49c2a | 00091f46  0be46c52 | 429a0908  00049400 | d01062f0  2e084000 | b5dd518b  7d1b2008 | d0403c39  8fc40002 |
| T9 | abab4b0c a75a17c7 | 1e6457e7 b074dd17 | 778f9a3c ca642f19 | aca8e67c c3b49c2a | 42c3a57 cfa78513d | 429a0908  00049400 | d01062f0  2e084000 | b5dd518b  7d1b2008 |
| T10 | 5cdf6c58f c052572 | abab4b0c a75a17c7 | 1e6457e7 b074dd17 | 778f9a3c ca642f19 | 4779767c c2ec5321 | aca8e67c c3b49c2a | 42c3a57 cfa78513d | 429a0908  00049400 |
| T11 | 4abe0af6 a67db2fe | 5cdf6c58f c052572 | abab4b0c a75a17c7 | 1e6457e7 b074dd17 | 500f7b61  186f6c2e | 4779767c c2ec5321 | aca8e67c c3b49c2a | 42c3a57 cfa78513d |
| T12 | 14e98634  2ddced0f | 4abe0af6 a67db2fe | 5cdf6c58f c052572 | abab4b0c a75a17c7 | 8347f573  6531b3ec | 500f7b61  186f6c2e | 4779767c c2ec5321 | aca8e67c c3b49c2a |
| T13 | cdf3bff28  83fc9d9 | 14e98634  2ddced0f | 4abe0af6 a67db2fe | 5cdf6c58f c052572 | 0ae07c86 b1181c75 | 8347f573  6531b3ec | 500f7b61  186f6c2e | 500f7b61  186f6c2e |
| T14 | f14998dd  5f70767e | cdf3bff28  83fc9d9 | 14e98634  2ddced0f | 4abe0af6 a67db2fe | 9a3fb4d3  8ab6cf06 | 0ae07c86 b1181c75 | 8347f573  6531b3ec | 500f7b61  186f6c2e |
| T15 | c4562769 a37d02c0 | f14998dd  5f70767e | cdf3bff28  83fc9d9 | 14e98634  2ddced0f | 0062e70a  1ef705c1 | 9a3fb4d3  8ab6cf06 | 0ae07c86 b1181c75 | 8347f573  6531b3ec |
| T16 | 96f60209 b6dc35ba | c4562769 a37d02c0 | f14998dd  5f70767e | cdf3bff28  83fc9d9 | 5e412143  88186c14 | 0062e70a  1ef705c1 | 9a3fb4d3  8ab6cf06 | 0ae07c86 b1181c75 |
| T17 | 0ac2b11d | 96f60209 | c4562769 | f14998dd | 3e264456 | 5e412143 | 0062e70a | 9a3fb4d3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a8f571c6 | b6dc35ba | a37d02c0 | 5f70767e | 7b709a78 | 88186c14 | 1ef705c1 | 8ab6cf06 |
| T18 | a77b7c03  5dd4c161 | a77b7c03  5dd4c161 | 96f60209 b6dc35ba | c4562769 a37d02c0 | 2deecc6b  39d64d78 | 3e264456  7b709a78 | 5e412143  88186c14 | 0062e70a  1ef705c1 |
| T19 | 8e30ff09a d488753 | a77b7c03  5dd4c161 | a77b7c03  5dd4c161 | a77b7c03  5dd4c161 | da34d667  3d452dcf | 2deecc6b  39d64d78 | 3e264456  7b709a78 | 5e412143  88186c14 |
| T20 | 4bb9e66d  1145bfdc | 8e30ff09a d488753 | a77b7c03  5dd4c161 | a77b7c03  5dd4c161 | d375471b de1ba3f2 | da34d667  3d452dcf | 2deecc6b  39d64d78 | 3e264456  7b709a78 |
| T21 | af4e23eb  622d0df4 | 4bb9e66d  1145bfdc | 8e30ff09a d488753 | a77b7c03  5dd4c161 | 64b5ae54  24598428 | d375471b de1ba3f2 | da34d667  3d452dcf | 2deecc6b  39d64d78 |
| T22 | 1d05d9ba  779a1a78 | af4e23eb  622d0df4 | 4bb9e66d  1145bfdc | 8e30ff09a d488753 | b889ed34 abd7aa37 | 64b5ae54  24598428 | d375471b de1ba3f2 | da34d667  3d452dcf |
| T23 | dc8049af a6acd502 | 1d05d9ba  779a1a78 | af4e23eb  622d0df4 | 4bb9e66d  1145bfdc | 8407818e  9b28cc12 | b889ed34 abd7aa37 | 64b5ae54  24598428 | d375471b de1ba3f2 |
| T24 | caed4b5f a322b984 | dc8049af a6acd502 | 1d05d9ba  779a1a78 | af4e23eb  622d0df4 | 05f3fba45  4e5de3d | 8407818e  9b28cc12 | b889ed34 abd7aa37 | 64b5ae54  24598428 |
| T24 | 9356ac3e c3e51459 | caed4b5f a322b984 | dc8049af a6acd502 | 1d05d9ba  779a1a78 | 481b8a6e e5e07031 | 05f3fba45  4e5de3d | 8407818e  9b28cc12 | b889ed34 abd7aa37 |
| T25 | 7196e8fa  538ba4bf | 9356ac3e c3e51459 | caed4b5f a322b984 | dc8049af a6acd502 | 04673551  3cdd14d3 | 481b8a6e e5e07031 | 05f3fba45  4e5de3d | 8407818e  9b28cc12 |
| T26 | d9c900e0 1968c33e | 7196e8fa  538ba4bf | 9356ac3e c3e51459 | caed4b5f a322b984 | 7d40e6be  47d85702 | 04673551  3cdd14d3 | 481b8a6e e5e07031 | 05f3fba45  4e5de3d |
| T27 | a41eb7e5 a27588e3 | d9c900e0 1968c33e | 7196e8fa  538ba4bf | 9356ac3e c3e51459 | caed4b5f a322b984 | 7d40e6be  47d85702 | 04673551  3cdd14d3 | 481b8a6e e5e07031 |
| T28 | d6d4f860  8b8ab199 | a41eb7e5 a27588e3 | d9c900e0 1968c33e | 7196e8fa  538ba4bf | 24b9c216 f915da60 | caed4b5f a322b984 | 7d40e6be  47d85702 | 04673551  3cdd14d3 |
| T29 | 9fe22e39  448d50ed | d6d4f860  8b8ab199 | a41eb7e5 a27588e3 | d9c900e0 1968c33e | 88761eb6  7845978e | 24b9c216 f915da60 | caed4b5f a322b984 | 7d40e6be  47d85702 |
| T30 | 6a8143b1  716ee33d | 9fe22e39  448d50ed | d6d4f860  8b8ab199 | a41eb7e5 a27588e3 | 6bc425af  294bbf79 | 88761eb6  7845978e | 24b9c216 f915da60 | caed4b5f a322b984 |
| T31 | b6df019c a515cafb | 6a8143b1  716ee33d | 9fe22e39  448d50ed | d6d4f860  8b8ab199 | 754b3a46  1a665640 | 6bc425af  294bbf79 | 88761eb6  7845978e | 24b9c216 f915da60 |
| T32 | bd7176b d099bb2f2 | b6df019c a515cafb | 6a8143b1  716ee33d | 9fe22e39  448d50ed | 8575839b  0f08472b | 754b3a46  1a665640 | 6bc425af  294bbf79 | 88761eb6  7845978e |
| T33 | 082ee70d | bd7176b | b6df019c | 6a8143b1 | 807f4578 | 8575839b | 754b3a46 | 754b3a46 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3f352aac | d099bb2f2 | a515cafb | 716ee33d | 4852303f | 0f08472b | 1a665640 | 1a665640 |
| T34 | bb3fcd75  85b59e32 | 082ee70d  3f352aac | bd7176b d099bb2f2 | b6df019c a515cafb | 2201c7cb d34e31fe | 807f4578  4852303f | 8575839b  0f08472b | 754b3a46  1a665640 |
| T35 | d43e5686  506fa05d | bb3fcd75  85b59e32 | 082ee70d  3f352aac | bd7176bd  099bb2f2 | 0e109efc  47927341 | 2201c7cb d34e31fe | 807f4578  4852303f | 8575839b  0f08472b |
| T36 | eec6fca5a a657661 | d43e5686  506fa05d | bb3fcd75  85b59e32 | 082ee70d  3f352aac | 7be0b7e7  0bdabe53 | 0e109efc  47927341 | 2201c7cb d34e31fe | 807f4578  4852303f |
| T37 | 0aaf99c5  9e7fadbd | eec6fca5a a657661 | d43e5686  506fa05d | bb3fcd75  85b59e32 | d5471e3d c7171224 | 7be0b7e7  0bdabe53 | 0e109efc  47927341 | 2201c7cb d34e31fe |
| T38 | e4dd663f  44d313bc | 0aaf99c5  9e7fadbd | eec6fca5a a657661 | d43e5686  506fa05d | f517b690 d940a294 | d5471e3d c7171224 | 7be0b7e7  0bdabe53 | 0e109efc  47927341 |
| T39 | fb90a8a7 6dea1bfe | e4dd663f  44d313bc | 0aaf99c5  9e7fadbd | eec6fca5a a657661 | 65234d5b  5049e665 | f517b690 d940a294 | d5471e3d c7171224 | 7be0b7e7  0bdabe53 |
| T40 | c1af3588  33cb03c0 | fb90a8a7 6dea1bfe | e4dd663f  44d313bc | 0aaf99c5  9e7fadbd | d4883c0c  21dda190 | 65234d5b  5049e665 | f517b690 d940a294 | d5471e3d c7171224 |
| T41 | 1e99cae9 d4cf0409 | c1af3588  33cb03c0 | fb90a8a7 6dea1bfe | e4dd663f  44d313bc | a2a475de ac0e8b42 | d4883c0c  21dda190 | 65234d5b  5049e665 | f517b690 d940a294 |
| T42 | 714e2ad f4e23ff24 | 1e99cae9 d4cf0409 | c1af3588  33cb03c0 | fb90a8a7 6dea1bfe | 48253e21 b26d8cf9 | a2a475de ac0e8b42 | d4883c0c  21dda190 | 65234d5b  5049e665 |
| T43 | ae70c7d1  1ea84a83 | 714e2ad f4e23ff24 | 714e2ad f4e23ff24 | 714e2ad f4e23ff24 | 371f12f3  33f7e5b9 | 48253e21 b26d8cf9 | a2a475de ac0e8b42 | d4883c0c  21dda190 |
| T44 | bf8d9453 b9876b0a | ae70c7d1  1ea84a83 | 714e2ad f4e23ff24 | 714e2ad f4e23ff24 | 59d0238c e137abd7 | 371f12f3  33f7e5b9 | 48253e21 b26d8cf9 | a2a475de ac0e8b42 |
| T45 | 5f3c64b7  546e2cec | bf8d9453 b9876b0a | bf8d9453 b9876b0a | 714e2ad f4e23ff24 | b81e85d4  27045550 | 59d0238c e137abd7 | 371f12f3  33f7e5b9 | 48253e21 b26d8cf9 |
| T46 | c45dd4a2 d2fea059 | 5f3c64b7  546e2cec | bf8d9453 b9876b0a | bf8d9453 b9876b0a | e0847194  6c17b0b6 | b81e85d4  27045550 | 59d0238c e137abd7 | 371f12f3  33f7e5b9 |
| T47 | f579471  1faa60f63 | c45dd4a2 d2fea059 | 5f3c64b7  546e2cec | bf8d9453 b9876b0a | b4838c1 c28fee7bc | e0847194  6c17b0b6 | b81e85d4  27045550 | 59d0238c e137abd7 |
| T48 | 754b3a46  1a665640 | f579471  1faa60f63 | c45dd4a2 d2fea059 | 5f3c64b7  546e2cec | b6df019c a515cafb | b4838c1 c28fee7bc | e0847194  6c17b0b6 | b81e85d4  27045550 |
| T49 | a5778494  5550cf92 | 754b3a46  1a665640 | f579471  1faa60f63 | c45dd4a2 d2fea059 | eac55163  36bc8882 | b6df019c a515cafb | b4838c1 c28fee7bc | e0847194  6c17b0b6 |
| T50 | 24a11242 | a5778494 | 754b3a46 | f579471 | 06edacae | eac55163 | b6df019c | b4838c1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 62a331c7 | 5550cf92 | 1a665640 | 1faa60f63 | 6e7b54ad | 36bc8882 | a515cafb | c28fee7bc |
| T51 | f5198a41 eb80e022 | 24a11242  62a331c7 | 24a11242  62a331c7 | 754b3a46  1a665640 | 3dc764f8  9e54043a | 06edacae  6e7b54ad | eac55163  36bc8882 | eac55163  36bc8882 |
| T52 | 56fb5883 f1c87a05 | f5198a41 eb80e022 | 24a11242  62a331c7 | 24a11242  62a331c7 | 617bbf70 bb165211 | 3dc764f8  9e54043a | 06edacae  6e7b54ad | eac55163  36bc8882 |
| T53 | eb85d192  01c89694 | 56fb5883 f1c87a05 | f5198a41 eb80e022 | 24a11242  62a331c7 | 9ced2498  3eec8723 | 617bbf70 bb165211 | 3dc764f8  9e54043a | 06edacae  6e7b54ad |
| T54 | ccf99a80 f92bf004 | eb85d192  01c89694 | 56fb5883 f1c87a05 | f5198a41 eb80e022 | 6c27893a  31b0e07e | 9ced2498  3eec8723 | 617bbf70 bb165211 | 3dc764f8  9e54043a |
| T55 | 587f308d  58681928 | ccf99a80 f92bf004 | eb85d192  01c89694 | 56fb5883 f1c87a05 | ae993474  363efe68 | 6c27893a  31b0e07e | 6c27893a  31b0e07e | 617bbf70 bb165211 |
| T56 | 7799c75 acc748c0f | 587f308d  58681928 | ccf99a80 f92bf004 | eb85d192  01c89694 | 19316beb c88e01f2 | ae993474  363efe68 | 6c27893a  31b0e07e | 6c27893a  31b0e07e |
| T57 | a7bbd65b f64f58c8 | 7799c75 acc748c0f | 587f308d  58681928 | ccf99a80 f92bf004 | e52a24fa e4e8fc9b | 19316beb c88e01f2 | ae993474  363efe68 | 6c27893a  31b0e07e |
| T58 | ccf99a80f  92bf002 | a7bbd65b f64f58c8 | 7799c75 acc748c0f | 587f308d  58681928 | 042c2dc9 a5bf558a | e52a24fa e4e8fc9b | 19316beb c88e01f2 | ae993474  363efe68 |
| T59 | 0928b75c  925f91e2 | ccf99a80f  92bf002 | a7bbd65b f64f58c8 | 7799c75 acc748c0f | 79f4be3c  5a372911 | 042c2dc9 a5bf558a | e52a24fa e4e8fc9b | 19316beb c88e01f2 |
| T60 | 17758950  2dd39aa2 | 0928b75c  925f91e2 | ccf99a80f  92bf002 | a7bbd65b f64f58c8 | 69c8f40eb  38b6022 | 79f4be3c  5a372911 | 042c2dc9 a5bf558a | e52a24fa e4e8fc9b |
| T61 | 75476f54  56e82f9c | 17758950  2dd39aa2 | 0928b75c  925f91e2 | ccf99a80f  92bf002 | 30247024  47f76224 | 69c8f40eb  38b6022 | 79f4be3c  5a372911 | 042c2dc9 a5bf558a |
| T62 | 7a499ae0  5da50bf2 | 75476f54  56e82f9c | 17758950  2dd39aa2 | 0928b75c  925f91e2 | 89dc825e  7235c74b | 30247024  47f76224 | 69c8f40eb  38b6022 | 79f4be3c  5a372911 |
| T63 | 40c28c34 b1bbe906 | 7a499ae0  5da50bf2 | 75476f54  56e82f9c | 17758950  2dd39aa2 | 356d08d9  82e2ead4 | 89dc825e  7235c74b | 30247024  47f76224 | 69c8f40eb  38b6022 |
| T64 | 97901f33  3e662fdc | 40c28c34 b1bbe906 | 7a499ae0  5da50bf2 | 75476f54  56e82f9c | 4472b2e3  31ddfab4 | 356d08d9  82e2ead4 | 89dc825e  7235c74b | 30247024  47f76224 |
| T65 | 6db5469f a19c0e27 | 97901f33  3e662fdc | 40c28c34 b1bbe906 | 7a499ae0  5da50bf2 | e7748fbf  744a5240 | 4472b2e3  31ddfab4 | 356d08d9  82e2ead4 | 89dc825e  7235c74b |
| T66 | 0d7ab032  08f1d7a5 | 6db5469f a19c0e27 | 97901f33  3e662fdc | 40c28c34 b1bbe906 | 16beec0f ec168e79 | e7748fbf  744a5240 | 4472b2e3  31ddfab4 | 356d08d9  82e2ead4 |
| T67 | b4cb0df3 | 0d7ab032 | 6db5469f | 97901f33 | 8fe3d280 | 16beec0f | e7748fbf | 4472b2e3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 32d108ab | 08f1d7a5 | a19c0e27 | 3e662fdc | 97f18618 | ec168e79 | 744a5240 | 31ddfab4 |
| T68 | 11200c2d  47c082f8 | b4cb0df3  32d108ab | 0d7ab032  08f1d7a5 | 6db5469f a19c0e27 | 7416ca18 d9e265e0 | 8fe3d280  97f18618 | 16beec0f ec168e79 | e7748fbf  744a5240 |
| T69 | 4abe0af6 a67db2fe | 11200c2d  47c082f8 | b4cb0df3  32d108ab | 0d7ab032  08f1d7a5 | 500f7b6  1186f6c2e | 7416ca18 d9e265e0 | 8fe3d280  97f18618 | 16beec0f ec168e79 |
| T70 | 5cdf6c58 fc052572 | 4abe0af6 a67db2fe | 11200c2d  47c082f8 | b4cb0df3  32d108ab | 4779767c c2ec5321 | 500f7b6  1186f6c2e | 7416ca18 d9e265e0 | 8fe3d280  97f18618 |
| T71 | e1053fc8  5f9e56be | 5cdf6c58 fc052572 | 4abe0af6 a67db2fe | 4abe0af6 a67db2fe | 70012019  48fb3d71 | 4779767c c2ec5321 | 500f7b6  1186f6c2e | 7416ca18 d9e265e0 |
| T72 | 5ec3802b  9ecfef33 | e1053fc8  5f9e56be | 5cdf6c58 fc052572 | 4abe0af6 a67db2fe | 88146da7  6ff6f23a | 70012019  48fb3d71 | 4779767c c2ec5321 | 500f7b6  1186f6c2e |
| T73 | 8901cffe 7a74db98 | 5ec3802b  9ecfef33 | e1053fc8  5f9e56be | 5cdf6c58 fc052572 | 0ac2b11d a8f571c6 | 88146da7  6ff6f23a | 70012019  48fb3d71 | 4779767c c2ec5321 |
| T74 | 5f2eead6  9efb4233 | 8901cffe 7a74db98 | 5ec3802b  9ecfef33 | e1053fc8  5f9e56be | 629a292a  367cd507 | 0ac2b11d a8f571c6 | 88146da7  6ff6f23a | 70012019  48fb3d71 |
| T75 | 4ad358c6  082d4717 | 5f2eead6  9efb4233 | 8901cffe 7a74db98 | 5ec3802b  9ecfef33 | 152fecd8 f70e5939 | 629a292a  367cd507 | 0ac2b11d a8f571c6 | 88146da7  6ff6f23a |
| T76 | 1323f9c9  63181372 | 4ad358c6  082d4717 | 5f2eead6  9efb4233 | 8901cffe 7a74db98 | 1ca640c2  1fef775 | 152fecd8 f70e5939 | 629a292a  367cd507 | 0ac2b11d a8f571c6 |
| T77 | 8c384dfe f0e7c148 | 1323f9c9  63181372 | 4ad358c6  082d4717 | 5f2eead6  9efb4233 | b077d2ac  67d9a425 | 1ca640c2  1fef775 | 152fecd8 f70e5939 | 629a292a  367cd507 |
| T78 | a5cfd4c7 e26fe069 | 8c384dfe f0e7c148 | 1323f9c9  63181372 | 4ad358c6  082d4717 | 2bd78a7e  7a6d37b7 | b077d2ac  67d9a425 | 1ca640c2  1fef775 | 152fecd8 f70e5939 |
| T79 | 48485e78  5985172a | a5cfd4c7 e26fe069 | 8c384dfe f0e7c148 | 1323f9c9  63181372 | 4ad358c6  082d4717 | 2bd78a7e  7a6d37b7 | b077d2ac  67d9a425 | 1ca640c2  1fef775 |

13. penjumlahan variabel kerja dengan inisialisasi Hash

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variabel** | **Hash value** |  | **Variabel Kerja** | **Hasil** |
| **H0 (a)** | **6a09e667f3bcc908** | **+** | **48485e785985172a** | **71c5156e3efe31de** |
| **H1 (b)** | **bb67ae8584caa73b** | **+** | **a5cfd4c7e26fe069** | **1597d9bda28a08c2** |
| **dH2 (c)** | **3c6ef372fe94f82b** | **+** | **8c384dfef0e7c148** | **2c8a74171ef7cb973** |
| **H3 (d)** | **a54ff53a5f1d36f1** | **+** | **1323f9c963181372** | **922bfb70fc052a74** |
| **H4 (e)** | **510e527fade682d1** | **+** | **4ad358c6082d4717** | **3fa48d4e7183bbba** |
| **H5 (f)** | **9b05688c2b3e6c1f** | **+** | **2bd78a7ea6d37b7** | **c6dcf30ab22ffc56** |
| **H6 (g)** | **1f83d9abfb41bd6b** | **+** | **b077d2ac67d9a425** | **cffbae582780978d** |
| **H7 (h)** | **5be0cd19137e2179** | **+** | **1ca640c21fef775** | **78870ddb336918ee** |

Sehingga didapatkan:

71c5156e3efe31de1597d9bda28a08c2c8a74171ef7cb973922bfb70fc052a743fa48d

4e7183bbbac6dcf30ab22ffc56cffbae582780978d78870ddb336918ee