**ALGORITMA KRIPTOGRAFI**

**DAN CONTOHNYA**



**Disusun Oleh :**

Nanda Annisa Awalia (201581039)

**FAKULTAS ILMU KOMPUTER**

**JURUSAN TEKNIK INFORMATIKA**

**UNIVERSITAS ESA UNGGUL**

**JAKARTA - INDONESIA**

**2017/2018**

# **KATA PENGANTAR**

Puji syukur kami panjatkan kehadirat Allah SWT karena atas bimbingan dan petunjuk serta kemudahan yang diberikan-Nya, penulis dapat menyelesaikan penyusunan makalah dengan judul *“Algoritma Kriptografi dan Contohnya”* ini dengan baik dan tepat pada waktunya.

Penulis menyadari bahwa dalam penyusunan makalah ini tidak akan tuntas tanpa adanya bimbingan serta bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ucapkan terima kasih yang sebesar-besarnya, khususnya kepada:

1. Bapak Tri Ismardiko Widyawan selaku dosen mata kuliah Kriptografi.
2. Dan untuk teman-teman yang lain yang tergabung dalam kelas “Kriptografi Sesi 1”.

Penulis mengakui bahwa dalam makalah ini masih banyak terdapat kekurangan dan kesalahan baik dari segi rangkaian kalimat maupun penyajian materinya. Oleh karena itu, untuk perbaikan makalah ini, penulis mengharapkan saran dan kritikan yang membangun dari semua pihak untuk dijadikan pedoman dalam penulisan kearah yang lebih baik lagi. Akhir kata, semoga makalah ini dapat berguna dan bermanfaat bagi kita semua.

Jakarta, 17 Oktober 2017

Penulis

**DAFTAR ISI**

[**KATA PENGANTAR** i](#_Toc504858492)

[**DAFTAR ISI** ii](#_Toc504858493)

[**BAB I**](#_Toc504858494) [**PENDAHULUAN** 1](#_Toc504858495)

[**1.1** **Algoritma Kriptografi** 1](#_Toc504858496)

[**1.2** **Tujuan Kriptografi** 1](#_Toc504858497)

[**1.3** **Jenis-Jenis Kriptografi** 2](#_Toc504858498)

[**BAB II**](#_Toc504858499) [**PEMBAHASAN** 3](#_Toc504858500)

[**2.1** **DES (*Data Encryption Standard*)** 3](#_Toc504858501)

[**2.1.1** **Definisi Algoritma DES** 3](#_Toc504858502)

[**2.1.2** **Sejarah DES** 3](#_Toc504858503)

[**2.1.3** **Cara Kerja DES** 3](#_Toc504858504)

[**2.1.4** **Implementasi DES** 8](#_Toc504858505)

[**2.1.5** **Kelebihan dan Kekurangan DES** 9](#_Toc504858506)

[**2.2** **AES (*Advanced Encryption Standard)*** 9](#_Toc504858507)

[**2.2.1** **Definisi Algoritma AES** 9](#_Toc504858508)

[**2.2.2** **Sejarah AES** 10](#_Toc504858509)

[**2.2.3** **Metode Algoritma AES** 11](#_Toc504858510)

[**2.2.3.1** ***Add Round Key*** 12](#_Toc504858511)

[**2.2.3.2** ***Sub Bytes*** 12](#_Toc504858512)

[**2.2.3.3** ***Shift Rows*** 13](#_Toc504858513)

[**2.2.3.4** ***Mix Column*** 14](#_Toc504858514)

[**2.2.4** **Diagram Alir AES** 15](#_Toc504858515)

[**2.2.5** **Implementasi AES** 17](#_Toc504858516)

[**2.2.6** **Kelebihan dan Kekurangan AES** 17](#_Toc504858517)

[**2.3** **RSA (*Rivest-Shamir-Adleman*)** 17](#_Toc504858518)

[**2.3.1** **Definisi Algoritma RSA** 17](#_Toc504858519)

[**2.3.2** **Konsep RSA** 18](#_Toc504858520)

[**2.3.3** **Cara Kerja Algoritma RSA** 19](#_Toc504858521)

[**2.3.4** **Kelebihan dan Kekurangan RSA** 21](#_Toc504858522)

[**2.4** **MD5 (*Message Digest 5*)** 21](#_Toc504858523)

[**2.4.1** **Definisi Algortima MD5** 21](#_Toc504858524)

[**2.4.2** **Sejarah MD5** 21](#_Toc504858525)

[**2.4.3** **Cara Kerja MD5** 21](#_Toc504858526)

[**2.4.4** **Implementasi MD5** 27](#_Toc504858527)

[**2.4.5** **Kelebihan dan Kekurangan MD5** 27](#_Toc504858528)

[**2.5** **SHA-1 (*Secure Hash Algorithm*)** 27](#_Toc504858529)

[**2.5.1** **Definisi SHA-1** 27](#_Toc504858530)

[**2.5.2** **Cara Kerja SHA-1** 28](#_Toc504858531)

[**2.5.3** **Implementasi SHA-1** 28](#_Toc504858532)

[**2.5.4** **Kelebihan dan Kekurangan SHA-1** 29](#_Toc504858533)

[**2.6** **Triple DES (*Triple Data Encryption Standard*)** 29](#_Toc504858534)

[**2.6.1** **Definisi Algoritma Triple DES** 29](#_Toc504858535)

[**2.6.2** **Sejarah Triple DES** 30](#_Toc504858536)

[**2.6.3** **Cara Kerja Triple DES** 30](#_Toc504858537)

[**2.6.3.1** ***Triple DES with 2-Key*** 30](#_Toc504858538)

[**2.6.3.2** ***Triple DES with 3-Key*** 31](#_Toc504858539)

[**2.6.4** **Kelebihan dan Kekurang Triple DES** 32](#_Toc504858540)

[**2.7** **DSA (*Digital Signature Algorithm*)** 32](#_Toc504858541)

[**2.7.1** **Definisi Algoritma DSA** 32](#_Toc504858542)

[**2.7.2** **Sejarah DSA** 33](#_Toc504858543)

[**2.7.3** **Fungsi Hash** 34](#_Toc504858544)

[**2.7.4** **Cara Kerja DSA** 34](#_Toc504858545)

[**2.7.5** **Penggunaan Tanda Tangan Digital** 35](#_Toc504858546)

[**2.7.6** **Implementasi Skema Tanda Tangan Digital** 35](#_Toc504858547)

[**2.7.7** **Kelebihan dan Kekurangan Digital Signature** 36](#_Toc504858548)

[**2.8** **Screenshoot Coding dan Program** 37](#_Toc504858549)

[**2.8.1** **DES (*Data Encryption Standard*)** 37](#_Toc504858550)

[**2.8.2** **AES (*Advanced Encryption Standard*)** 38](#_Toc504858551)

[**2.8.3** **Triple DES (*Triple Data Encryption Standard*)** 39](#_Toc504858552)

[**2.8.4** **MD5 (*Message Digest 5*)** 41](#_Toc504858553)

[**2.8.5** **SHA-1 (*Secure Hash Algorithm*)** 44](#_Toc504858554)

[**2.8.6** **DSA (Digital Signature Algorithm)** 49](#_Toc504858555)

[**BAB III**](#_Toc504858556) [**KESIMPULAN** 50](#_Toc504858557)

[**3.1** **Kesimpulan** 50](#_Toc504858558)

[**3.2** **Saran** 50](#_Toc504858559)

# **BAB I**

# **PENDAHULUAN**

1. **Algoritma Kriptografi**

Algoritma merupakan urutan langkah-langkah logis untuk menyelesaikan masalah yang disusun secara matematis dan benar. Sedangkan kriptografi *(cryptography)* berasal dari kata *“crypto”* yang berarti *“secret”* (rahasia) dan *“graphy”* yang berarti *“writing”* (tulisan). Kriptografi merupakan suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan, dari pengirim ke penerima tanpa mengalami gangguan dari pihak ketiga. Sehingga algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut.

**Algoritma kriptografi terdiri dari tiga fungsi dasar, yaitu :**

1. **Enkripsi**, merupakan hal yang sangat penting dalam kriptografi. Enkripsi merupakan pengaman data yang dikirimkan agar terjaga kerahasiannya. Enkripsi dapat diartikan dengan cipher atau kode.
2. **Deskripsi**, merupakan kebalikan enkripsi. Pesan yang telah dienkripsi-dikembalikan ke bentuk asalnya (tesk-asli), disebut dengan deskripsi pesan.
3. **Kunci**, yang dimaksud adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, yaitu kunci rahasia (*private key*) dan kunci umum (*public key*).

Secara umum fungsi tersebut dapat digambarkan dengan :



**Gambar 1 – Proses Enkripsi dan Dekripsi**

* 1. **Tujuan Kriptografi**

Secara umum tujuan ilmu kriptografi diciptakan adalah untuk merahasiakan suatu pesan agar tidak bisa dibaca oleh pihak yang tidak absah. Berpedoman dari tujuan umum tersebut, terdapat empat tujuan mendasar ilmu kriptografi digunakan dalam bidang keamanan informasi yaitu :

1. ***Confidelity* (kerahasian)** yaitu layanan agar isi pesan yang dikirimkan tetap rahasia dan tidak diketahui oleh pihak lain (kecuali pihak pengirim, pihak penerima atau pihak-pihak memiliki ijin).
2. ***Data integrity* (keutuhan data)** yaitu layanan yang mampu mengenali/mendeteksi adanya manipulasi (penghapusan, pengubahan atau penambahan) data yang tidak sah (oleh pihak lain).
3. ***Authentication* (otentik)** yaitu layanan yang berhubungan dengan identifikasi. Baik otentikasi pihak-pihak yang terlibat dalam pengiriman data maupun otentikasi keaslian data/informasi.
4. ***Non-repudiation* (anti-penyangkalan)** yaitu layanan yang dapat mencegah suatu pihak untuk menyangkal aksi yang dilakukan sebelumnya (menyangkal bahwa pesan tersebut berasal dari dirinya).
   1. **Jenis-Jenis Kriptografi**

Berdasarkan kunci yang dipakainya, algoritma kriptografi dibagi menjadi tiga jenis :

1. **Algoritma Simetri**

Algoritma yang memakai kunci simetri diantaranya adalah :

* *Blok Chiper* : *Data Encryption Standard* (DES), *International Data Encryption* *Algorithm* (IDEA), *Advanced Encryption Standard* (AES).
* *Stream Chiper* : On Time Pad (OTP), A5, RC4.

1. **Algoritma Asimetri**

Algoritma yang memakai kunci publik diantaranya adalah : *Rivest-Shamir-Adleman* (RSA), *Diffle-Hellman* (DH), *Elliptic Curve Cryptography* (ECC), dan *Digital Signature Algorithm* (DSA).

1. **Fungsi Hash**

Contoh algoritma yang menggunakan fungsi hash adalah MD5 dan SHA1.

# **BAB II**

# **PEMBAHASAN**

1. **DES (*Data Encryption Standard*)** 
   * 1. **Definisi Algoritma DES**

DES atau singkatan dari *Data Encryption Standard* merupakan algoritma penyandian yang diadopsi dan dibakukan oleh NBS (*National Bureau Standard*) yang kini menjadi NIST (*National Institute of Standards and Technology*) pada tahun 1977 sebagai FIPS 46 (*Federal Information Processing Standard*). Algoritma DES merupakan algoritma enkripsi blok simetris. DES dikatakan enkripsi blok karena pemrosesan data baik enkripsi maupun dekripsi, diimplementasikan per blok (dalam hal ini 8 byte). DES dikatakan enkripsi simetris karena algoritma yang digunakan untuk enkripsi relatif atau bahkan sama persis dengan algoritma yang digunakan dalam proses dekripsi. Dalam algoritma penyandian DES, kunci yang digunakan dalam proses enkripsi dan dekripsi haruslah sama, supaya data dapat dikembalikan ke bentuk aslinya. Bisa jadi, karena “kesamaan” kunci inilah DES juga dinamakan algoritma enkripsi simetris. Inti dari proses enkripsi adalah penyembunyian data dengan mengaburkan data “asli” dan mengurangi keteraturan informasi, sehingga data tersebut tidak dapat “dibaca” kecuali oleh pihak yang berhak.

* + 1. **Sejarah DES**

DES bermula dari hasil riset Tuchman Meyer yang diajukan sebagai kandidat Sandi Standard Nasional yang diusulkan oleh NBS. Konon katanya, algoritma yang dikembangkan oleh Tuchman Meyer ini merupakan algoritma terbaik dari semua kandidat Sandi Standard Nasional. Pada mulanya, algoritma yang kini disebut DES, memiliki panjang kunci sandi 128 bit. Namun selama proses pengadopsian, NBS melibatkan NSA (*National Security Agency*), dan algoritma sandi ini mengalami pengurangan ukuran kunci sandi dari 128 bit menjadi 56 bit saja. Sebagian orang mungkin mengira bahwa pengurangan panjang kunci sandi ini merupakan usulan NSA untuk melemahkan algoritma Tuchman Meyer karena motif politik tertentu. Entah itu untuk mempermudah penyadapan atau untuk melemahkan pengamanan informasi lawan politik. Mungkin NSA menginginkan algoritma Tuchman Meyer ini “cukup aman” untuk digunakan warga sipil, tetapi mudah dipecahkan oleh organisasi besar semisal NSA dengan peralatan canggihnya. Bila dibandingkan dengan performa komputer personal pada saat itu, algoritma sandi dengan panjang kunci 56 bit dapat dikatakan cukup aman bila digunakan oleh orang-orang “biasa”, tapi dapat dengan mudah dipecahkan dengan peralatan canggih dan tentunya kepemilikan alat canggih ini hanya dapat dijangkau oleh organisasi elit seperti NSA. Dengan dukungan dana yang melimpah, pembuatan alat *brute‐force* DES bukanlah hal yang mustahil pada saat itu. Kini algoritma DES sudah usang dan keamanannya pun sudah tidak dapat dipertanggungjawabkan lagi. Kini komputer personal pun sudah cukup untuk membobol algoritma DES, apalagi dengan adanya teknologi *parallel computing* dan internet yang berkembang pesat. DES telah secara resmi digantikan fungsinya oleh AES (*Advanced Encryption Standard*) dengan panjang kunci sandi 128, 192 dan 256 bit.

* + 1. **Cara Kerja DES**

Dalam DES, algoritma dekripsi tepatnya merupakan proses kebalikan (*inverse*) algoritma enkripsi. Dalam prakteknya proses pembalikan (proses dekripsi) ini diimplementasikan dengan membalikkan urutan sub kunci yang digunakan dalam proses enkripsi, selebihnya algoritma enkripsi dan dekripsi adalah sama. Algoritma enkripsi DES bekerja dengan mengolah blok data 8 byte (64 bit) dengan blok kunci 8 byte (64 bit). Proses penyandian dalam DES diawali dengan fungsi pengacakan bit yang dinamai IP (*Initial Permutation*) kemudian fungsi inti DES yang diulang sebanyak 16 kali dan terakhir ditutup dengan fungsi pengacakan bit lain yang dikenal dengan nama IP-1 (*Inverse Initial Permutation*). Pada sisi lain algoritma penjadwalan sub kunci akan menghasilkan 16 sub kunci secara berurutan dari parameter kunci yang diberikan untuk digunakan pada setiap putaran fungsi inti DES. Sub kunci pertama untuk putaran pertama, sub kunci kedua untuk putaran pertama, sub kunci kedua untuk putaran kedua dan seterusnya hingga putaran ke 16.

Perlu diingat, kendatipun slot kunci yang disediakan digunakan berukuran 8 byte (64 bit), ternyata pada faktanya ukuran kunci yang digunakan hanya sebanyak 56 bit saja, karena bit paling signifikan (MSB) dari setiap bit diabaikan. Jadi sebenarnya ukuran kunci DES adalah 56 bit. Algoritma penjadwalan sub kunci dibentuk dari pengacakan bit dan pemutaran kiri ruas kanan dan kiri kunci. Pertama kali, bit-bit kunci diacak dengan *Permutation Choice* 1 dan dibagi dua menjadi ruas kiri dan ruas kanan. Kedua ruas tersebut kemudian diputar kiri dan diacak kembali dengan *Permutation Choice* 2 untuk menghasilkan sub kunci. Jumlah pemutaran ke kiri ditentukan secara spesifik untuk setiap sub kunci.

**Rinciannya adalah sebagai berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

**Inti dari semua proses permutasi dalam DES adalah pengacakan bit**. Sebagai contoh, jika masukan permutasi sebanyak n bit, maka akan ada sebanyak 2n kemungkinan masukan permutasi dan ada 2n kemungkinan hasil permutasi. Setiap satu kemungkinan masukan akan berpasangan dengan satu kemungkinan keluaran. Sebelum proses penjadwalan kunci dimulai, kunci terlebih dahulu dipetakan menjadi matriks 8x8 dan diberi indeks. Dalam setiap byte, indeks paling kecil melambangkan LSB dan indeks paling besar melambangkan MSB. Sebagai contoh, indeks ke 1 melambangkan LSB byte pertama, index ke 8 melambangkan MSB byte pertama, indeks ke 9 melambangkan LSB byte kedua, indeks ke 16 melambangkan MSB byte kedua dan seterusnya hingga indeks ke 64 yang melambangkan MSB byte ke 8. Mari kita perhatikan contoh dibawah ini.

**Kunci = 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

**Setelah pemetaan, hasilnya adalah sebagai berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

**Matriks Indeks Hasil Pemetaan**

Pengacakan bit *Permutation Choice*-1 akan mengolah 8 byte blok kunci menjadi 56 bit sub kunci yang siap diproses lebih lanjut. Untuk lebih mudahnya, proses pengacakan bit dilambangkan dengan pengacakan indeks bit yang bersangkutan. **Berikut ini adalah detail *Permutation Choice*-1.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Setelah kunci diacak dengan *Permutation Choice-*1, hasil pengacakan bit tersebut kemudian dibagi 2, yakni ruas kiri dan ruas kanan, masing masing berukuran 28 bit (ditandai dengan garis tebal pada hasil *Permutation Choice-*1). Selanjutnya kedua ruas tersebut kemudian mengalami pemutaran kiri sebanyak jumlah yang tertera pada tabel penjadwalan jumlah pemutaran yang telah kita bahas sebelumnya. **Berikut ini adalah ilustrasi pemutaran ke kiri sebanyak 1 kali (untuk ruas kiri atau ruas kanan kunci yang panjangnya 28 bit).** Untuk pemutaran ke kiri dengan jumlah yang lebih besar, cukup mengulangi proses ini sebanyak yang diinginkan.

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Setelah ruas kiri dan ruas kanan diputar kiri dengan jumlah tertentu, selanjutnya hasil pemutaran tersebut digabungkan kembali menjadi 56 bit dan diacak dengan *Permutation Choice*-2 untuk menghasilkan sub kunci. **Rincian *Permutation Choice*-2 adalah sebagai berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Karena jumlah keluaran *Permutation Choice* 48 bit sementara masukannya 56 bit, dengan demikian ada 8 bit yang “dihilangkan”. Bit-bit yang tidak muncul dalam keluaran *Permutation Choice*-2 diwarnai abu-abu. Selanjutnya, mari kita perjelas algoritma enkripsinya. Setelah melihat diagram blok secara keseluruhan proses enkripsi, ada tiga hal yang perlu digarisbawahi dan dibahas lebih lanjut yaitu, pertama IP (*Initial Permutation*), kedua detail fungsi F dan IP 1 (*Inverse Initial Permutation*).  
Selama proses enkripsi, pertama data dipetakan dan diberi indeks dengan prosedur sama persis seperti pemberian indeks pada penjadwalan kunci yang telah didiskusikan sebelumnya. **Selanjutnya hasil pemetaan diacak dengan menggunakan *Initial Permutation* dengan rincian sebagai berikut.**

**Masukan IP**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Setelah data melewati proses *Initial Permutation*, data yang akan disandikan kemudian dibagi menjadi dua ruas, yaitu ruas kiri dan ruas kanan yang masing-masing lebarnya 4 byte (32 bit). Pada setiap putaran, ruas kanan dan sub kunci yang bersangkutan diproses dalam fungsi F dan hasilnya di XOR dengan ruas kiri ruas kanan dan kiri dipertukarkan. Proses ini diulang sebanyak 16 kali.Pada putaran terakhir ruas kiri dan ruas kanan dipertukarkan kembali untuk menghilangkan efek pertukaran pada putaran terakhir. Hasil ini kemudian diacak kembali dengan menggunakan IP 1 (*Inverse Initial Permutation*). **XOR merupakan fungsi Boolean yang didefinisikan dengan table benaran berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

*Initial Permutation* dan *Inverse Initial Permutation* memiliki karakter saling menetralkan. Dalam notasi matematika, IP−1(IP(A))=A. ***Inverse Initial Permutation* didefinisikan sebagai berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Fungsi F merupakan fungsi inti kompleks yang terdiri dari beberapa proses. Fungsi F menerima dua parameter, yaitu sub kunci dan ruas kanan data yang akan dienkripsi. **Berikut ini adalah diagram blok detail fungsi F**. Pada fungsi F, ruas kanan (32 bit) diacak sekaligus diperluas dengan permutasi E menjadi 48 bit. Hasil pengacakan tersebut kemudian di XOR dengan sub kunci yang telah ditetapkan dengan putaran yang bersangkutan. Hasil XOR kemudian dipecah menjadi 8 unit yang masing‐masing lebarnya 6 bit. Setiap unit tersebut kemudian disubstitusikan dalam S-*Box* S1 hingga S8. 6 bit paling kiri disubstitusikan ke dalam S1 dan 6 bit paling kanan disubstitusikan ke dalam S8. Hasil setiap substitusi kemudian digabungkan menjadi data selebar 48 bit yang kemudian diacak dan diperpendek dengan permutasi P menjadi 32 bit. Hasil permutasi P kemudian dinyatakan sebagai keluaran fungsi F yang nantinya akan di XOR kan dengan ruas kiri data yang akan dienkripsi.

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Sekarang mari kita bahas detail fungsi F satu per satu. Pertama, permutasi E memetakan 32 bit masukan menjadi 48 bit keluaran. Karena lebar keluaran lebih besar dari lebar masukan, maka ada beberapa bit masukan yang digandakan untuk mengisi kekosongan. **Permutasi E didefinisikan sebagai berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Kedua, DES memiliki 8 buah S-*Box* (S1 hingga S8) yang memiliki masukan selebar 6 bit dan keluaran selebar 4 bit. Karena lebar keluaran S-*Box* lebih kecil daripada lebar masukannya, maka adakemungkinan beberapa kombinasi masukan yang berbeda akan menghasilkan keluaran yang sama. **Seandainya masukan setiap S-*Box* adalah 1 2 3 4 5 6 x x x x x x maka S1 hingga hingga S8 didefinisikan sebagai berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Ketiga, hasil substitusi S-*Box* kemudian digabungkan menjadi 32 bit dan diacak dengan permutasi P dan hasil permutasi P merupakan keluaran fungsi F yang nantinya di XOR dengan ruas kiri. **Permutasi P didefinisikan sebagai berikut.**

[](https://link.safelinkconverter.com/?id=aHR0cDovL3B1c2F0LW1ha2FsYWhrdS5ibG9nc3BvdC5jb20v&c=0&user=61204)

Dalam DES, algoritma yang digunakan dalam proses enkripsi sama persis dengan algoritma yang digunakan dalam proses dekripsi, hanya saja penggunaan sub kuncinya saja yang berbeda. Dalam proses dekripsi, urutan sub kunci yang digunakan merupakan kebalikan urutan sub kunci yang digunakan dalam proses enkripsi.

* + 1. **Implementasi DES**

Operasi yang digunakan dalam algoritma DES merupakan operasi‐operasi sederhana semisal *move*, *bit copy*, XOR, *lookup*, *shift* dan *rotate*. Semua operasi tersebut tersedia dalam mikroprosesor/mikrokontroler 8 bit. **Dengan demikian dapat kita simpulkan bahwa DES dapat diterapkan dalam platform 8 bit.**

Semua operasi permutasi dalam DES, baik IP, IP 1, PC 1, PC 2, E dan P, pada intinya hanyalah operasi penyalinan bit. Jika instruksi penyalinan bit tidak tersedia, maka permutasi juga dapat diimplementasikan dengan operasi *shift*, dengan memanfaatkan *carry* yang timbul dari setiap instruksi *shift*. Selain itu, operasi substitusi dengan S-*Box* juga dapat dengan mudah diimplementasikan menggunakan *table lookup* dengan ukuran yang masih dapat dijangkau. Dalam platform 32 atau 64 bit, DES dapat diimplementasikan lebih efektif lagi, tapi sayangnya operasi bit per bit seperti permutasi mungkin sedikit menyita performa prosesor dan memperlambat laju enkripsi per detik.

* + 1. **Kelebihan dan Kekurangan DES**

**Kelebihan DES:**

1. Sistem sandi lebih kompleks
2. Sulit diketahui pihak luar

**Kekurangan DES:**

1. Proses lebih lama
   1. **AES (*Advanced Encryption Standard)*** 
      1. **Definisi Algoritma AES**

AES (*Advanced Encryption Standard*) merupakan algoritma *cryptographic* yang dapat digunakan untuk mengamankan data. Algoritma AES adalah blok *chipertext* simetrik yang dapat mengenkripsi (*enchiper*) dan deskripsi (*decipher*) informasi. Enkripsi merubah data yang tidak dapat dibaca lagi disebut *ciphertext*, sebaliknya deskripsi adalah merubah *ciphertext* data menjadi bentuk semula yang kita kenal sebagai *plaintext*. **Algoritma AES menggunakan kunci kriptografi 128, 192, dan 256 bit.**



**Gambar - Panjang Kunci Algoritma Kriptografi AES**

AES adalah lanjutan dari algoritma enkripsi standar DES (*Data Encryption Standard*) yang masa berlakunya dianggap telah usai karena faktor keamanan. Kecepatan komputer yang sangat pesat dianggap sangat membahayakan DES, sehingga pada tanggal 2 Maret tahun 2001 ditetapkanlah algoritma baru Rinjadael sebagai AES. **Kriteria pemilihan AES didasarkan pada 3 kriteria utama yaitu: keamanan, harga, dan karakteristik algoritma beserta implementasinya.** Keamanan merupakan faktor penting dalam evaluasi (minimal semanan Triple DES), yang meliputi ketahanan terhadap semua analisis sandi yang telah diketahui dan diharapkan dapat menghadapi analisis sandi yang belum diketahui. Di samping itu, AES juga harus dapat digunakan secara bebas tanpa harus membayar royalti, dan juga murah untuk diimplementasikan pada *smart* *card* yang memiliki ukuran memori kecil. AES juga harus efisien dan cepat (minimal secepat Triple DES) dijalankan dalam berbagai mesin 8 bit hingga 64 bit, dan berbagai perangkat lunak. DES menggunakan stuktur Feistel yang memiliki kelebihan bahwa struktur enkripsi dan dekripsinya sama, meskipun menggunakan fungsi F yang tidak invertibel. Kelemahan Feistel yang utama adalah bahwa pada setiap ronde, hanya setengah data yang diolah. Sedangkan AES menggunakan struktur SPN (*Substitution* *Permutation* *Network*) yang memiliki derajat paralelisme yang lebih besar, sehingga diharapkan lebih cepat dari pada Feistel.

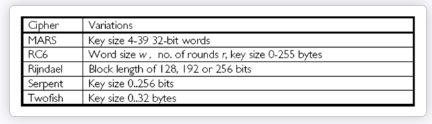
Kelemahan SPN pada umumnya (termasuk pada Rijndael) adalah berbedanya struktur enkripsi dan dekripsi sehingga diperlukan dua algoritma yang berbeda untuk enkripsi dan dekripsi. Dan tentu pula tingkat keamanan enkripsi dan dekripsinya menjadi berbeda. AES memiliki blok masukan dan keluaran serta kunci 128 bit. Untuk tingkat keamanan yang lebih tinggi, AES dapat menggunakan kunci 192 dan 256 bit. Setiap masukan 128 bit plaintext dimasukkan ke dalam state yang berbentuk bujursangkar berukuran 4×4 byte. State ini di-XOR dengan key dan selanjutnya diolah 10 kali dengan subtitusi-transformasi *linear*-*Addkey*. Dan di akhir diperoleh *ciphertext*.

**Berikut ini adalah operasi Rijndael (AES) yang menggunakan 128 bit kunci:**

1. Ekspansi kunci utama (dari 128 bit menjadi 1408 bit)
2. Pencampuran *subkey*
3. Ulang dari i=1 sampai i=10 Transformasi : *ByteSub* (subtitusi per byte), *ShiftRow* (Pergeseren byte perbaris), *MixColumn* (Operasi perkalian GF(2) per kolom)
4. Pencampuran *subkey* (dengan XOR)
5. Transformasi : *ByteSub* dan *ShiftRow*
6. Pencampuran *subkey*
   * 1. **Sejarah AES**

Pada tahun 1997, *National Institute of Standard and Technology* (NIST) *of United States* mengeluarkan *Advanced Encryption Standard* (AES) untuk menggantikan *Data Encryption Standard* (DES). AES dibangun dengan maksud untuk mengamankan pemerintahan di berbagai bidang. Algoritma AES di desain menggunakan blok *chiper* minimal dari blok 128 bit *input* dan mendukung ukuran 3 kunci (3-*key*-*sizes*), yaitu kunci 128 bit, 192 bit, dan 256 bit. Pada Agustus 1998, NIST mengumumkan bahwa ada 15 proposal AES yang telah diterima dan dievaluasi, setelah mengalami proses seleksi terhadap algoritma yang masuk**, NIST mengumumkan pada tahun 1999 bahwa hanya ada 5 algoritma yang diterima, algoritma tersebut adalah:**

1. MARS
2. RC6
3. Rijndael
4. Serpent
5. Twofish



**Gambar - Algoritma yang diterima oleh NIST**

Algoritma-algoritma tersebut manjalani berbagai macam pengetesan. Pada bulan Oktober 2000, NIST mengumumkan bahwa **Rijndael** sebagai algoritma yang terpilih untuk standar AES yang baru. Baru pada Februari 2001 NIST mengirimkan *draft* kepada *Federal Information Processing Standards* (FIPS) untuk standar AES. Kemudian pada 26 November 2001, NIST mengumumkan produk akhir dari *Advanced Encryption Standard* (AES).

* + 1. **Metode Algoritma AES**

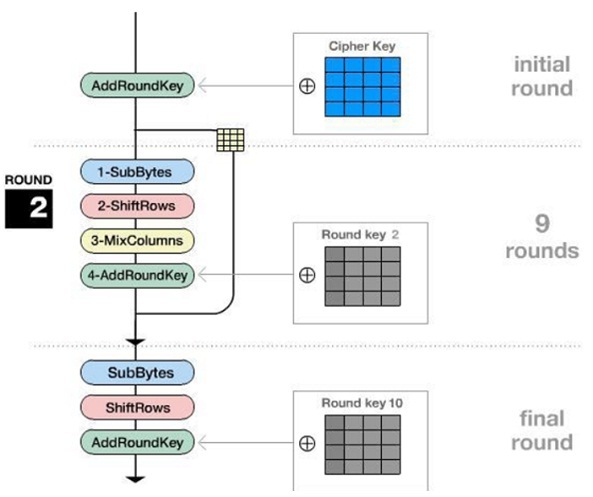
Algoritma kriptografi bernama Rijndael yang didesain oleh oleh Vincent Rijmen dan John Daemen asal Belgia keluar sebagai pemenang kontes algoritma kriptografi pengganti DES yang diadakan oleh NIST (*National Institutes of Standards and Technology*) milik pemerintah Amerika Serikat pada 26 November 2001. **Algoritma Rijndael inilah yang kemudian dikenal dengan *Advanced Encryption Standard* (AES).** Setelah mengalami beberapa proses standardisasi oleh NIST, Rijndael kemudian diadopsi menjadi standard algoritma kriptografi secara resmi pada 22 Mei 2002. Pada 2006, AES merupakan salah satu algoritma terpopuler yang digunakan dalam kriptografi kunci simetrik.

AES ini merupakan algoritma *block cipher* dengan menggunakan sistem permutasi dan substitusi (P-*Box* dan S-*Box*) bukan dengan jaringan Feistel sebagaimana *block cipher* pada umumnya. **Jenis AES terbagi menjadi 3, yaitu:**

1. AES-128
2. AES-192
3. AES-256

Pengelompokkan jenis AES ini adalah berdasarkan panjang kunci yang digunakan. Angka-angka di belakang kata AES menggambarkan panjang kunci yang digunakan pada tiap-tiap AES. Selain itu, hal yang membedakan dari masing-masing AES ini adalah banyaknya *round* yang dipakai. **AES-128 menggunakan 10 round, AES-192 sebanyak 12 *round*, dan AES-256 sebanyak 14 *round*.**

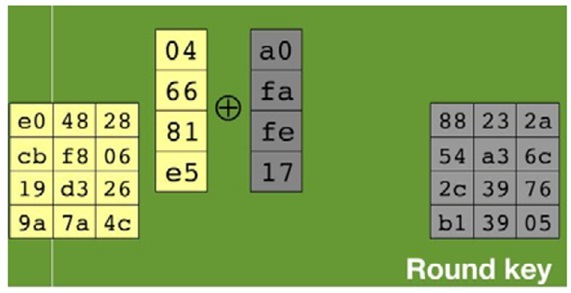
AES memiliki ukuran *block* yang tetap sepanjang 128 bit dan ukuran kunci sepanjang 128, 192, atau 256 bit. Tidak seperti Rijndael yang *block* dan kuncinya dapat berukuran kelipatan 32 bit dengan ukuran minimum 128 bit dan maksimum 256 bit. Berdasarkan ukuran *block* yang tetap, AES bekerja pada matriks berukuran 4x4 di mana tiap-tiap sel matriks terdiri atas 1 byte (8 bit). Sedangkan Rijndael sendiri dapat mempunyai ukuran matriks yang lebih dari itu dengan menambahkan kolom sebanyak yang diperlukan. Blok *chiper* tersebut dalam pembahasan ini akan diasumsikan sebagai sebuah kotak. Setiap *plainteks* akan dikonversikan terlebih dahulu ke dalam blok-blok tersebut dalam bentuk *heksadesimal*. Barulah kemudian blok itu akan diproses dengan metode yang akan dijelaskan. Secara umum metode yang digunakan dalam pemrosesan enkripsi dalam algoritma ini dapat dilihat melalui Gambar dibawah ini.



**Gambar 1. Diagram AES**

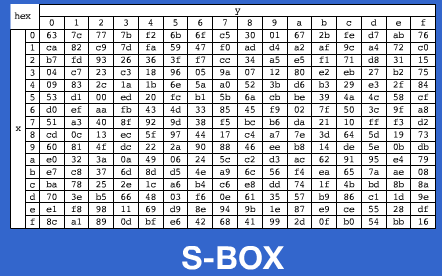
* + - 1. ***Add Round Key***

*Add Round Key* merupakan transformasi yang melakukan operasi XOR terhadap sebuah *round key* dengan *array state* dan hasilnya disimpan di *array state*

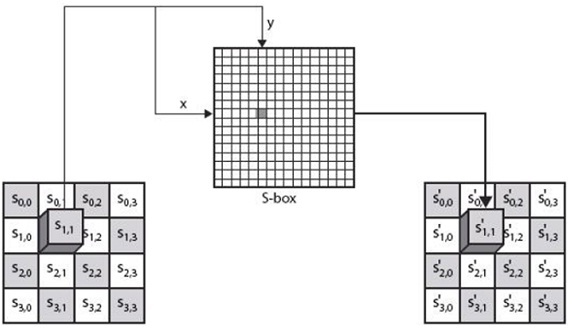
**Gambar 2. Ilustrasi *Add Round Key***

* + - 1. ***Sub Bytes***

Prinsip dari *Sub Bytes* adalah menukar isi matriks/tabel yang ada dengan matriks/tabel lain yang disebut dengan **Rijndael S-*Box***. Dibawah ini adalah contoh *Sub Bytes* dan Rijndael S-*Box*.



**Gambar 3. Rjindael S-*Box***



**Gambar 4. Ilustrasi *Sub Bytes***

**Keterangan:** Gambar 4 adalah contoh dari Rijndael S-*Box*, di sana terdapat nomor kolom dan nomor baris. Seperti yang telah disebutkan sebelumnya, tiap isi kotak dari blok chiper berisi informasi dalam bentuk *heksadesimal* yang terdiri dari dua digit, bisa angka-angka, angka-huruf, ataupun huruf-angka yang semuanya tercantum dalam Rijndael S-*Box*. Langkahnya adalah mengambil salah satu isi kotak matriks, mencocokkannya dengan digit kiri sebagai baris dan digit kanan sebagai kolom. Kemudian dengan mengetahui kolom dan baris, kita dapat mengambil sebuah isi tabel dari Rijndael S-*Box*. Langkah terakhir adalah mengubah keseluruhan blok *chiper* menjadi blok yang baru yang isinya adalah hasil penukaran semua isi blok dengan isi langkah yang disebutkan sebelumnya.

* + - 1. ***Shift Rows***

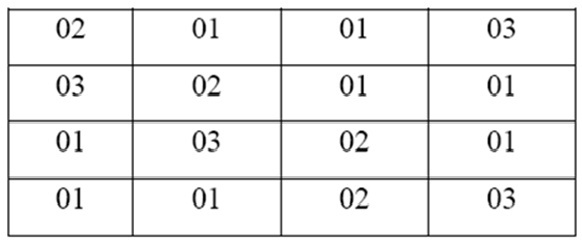
*Shift Rows* seperti namanya adalah sebuah proses yang melakukan *shift* atau pergeseran pada setiap elemen blok/tabel yang dilakukan per barisnya. Yaitu baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran 1 *byte*, baris ketiga dilakukan pergeseran 2 *byte*, dan baris keempat dilakukan pergeseran 3 *byte*. Pergeseran tersebut terlihat dalam sebuah blok adalah sebuah pergeseran tiap elemen ke kiri tergantung berapa byte tergesernya, tiap pergeseran 1 *byte* berarti bergeser ke kiri sebanyak satu kali. Ilustrasi dari tahap ini diperlihatkan oleh **Gambar 5. Ilustrasi Shift Rows** di bawah ini.

**** **** **** ****

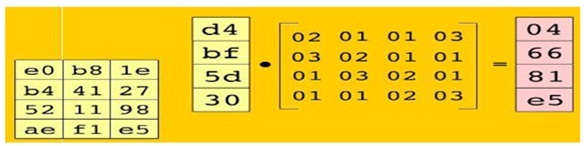
Seperti yang terlihat pada Gambar 5, tahap *Shift Row* sama sekali tidaklah rumit, karena ini adalah proses standar yang hanya berupa pergeseran. Langkah terakhir adalah *Mix Column*.

* + - 1. ***Mix Column***

Yang terjadi saat *Mix Column* adalah mengalikan tiap elemen dari blok *chiper* dengan matriks yang ditunjukkan oleh Gambar 6. Tabel sudah ditentukan dan siap pakai. Pengalian dilakukan seperti perkalian matriks biasa yaitu menggunakan *dot product* lalu perkalian keduanya dimasukkan ke dalam sebuah blok chiper baru. Ilustrasi dalam Gambar 7 akan menjelaskan mengenai bagaimana perkalian ini seharusnya dilakukan. Dengan begitu seluruh rangkaian proses yang terjadi pada AES telah dijelaskan dan selanjutnya adalah menerangkan mengenai penggunaan tiap-tiap proses tersebut.



**Gambar 6. Tabel untuk *Mix Column***

****

**Gambar 7. Ilustrasi *Mix Column***

* + 1. **Diagram Alir AES**

Kembali melihat diagram yang ditunjukkan oleh Gambar 1. Seperti yang terlihat semua proses yang telah dijelaskan sebelumnya terdapat pada diagram tersebut. Yang artinya adalah mulai dari ronde dua, dilakukan pengulangan terus menerus dengan rangkaian proses *Sub Bytes, Shift Rows, Mix Columns*, dan *Add Round Key*, setelah itu hasil dari ronde tersebut akan digunakan pada ronde berikutnya dengan metode yang sama. Namun pada ronde kesepuluh, proses *Mix Columns* tidak dilakukan, dengan kata lain urutan proses yang dilakukan adalah *Sub Bytes, Shift Rows*, dan *Add Round Key*, hasil dari *Add Round Key* inilah yang dijadikan sebagai *chiperteks* dari AES. Lebih jelasnya bisa dilihat dengan Gambar 8 dan 9 yang akan menerangkan mengenai kasus tersebut.

******

**Gambar 8. Ilustrasi Ronde 2 hingga Ronde 6**

****

**Gambar 9. Ilustrasi Ronde 7 hingga Ronde 10**

Dengan mengetahui semua proses yang ada pada AES, maka kita dapat menggunakannya dalam contoh kasus yang muncul di kehidupan sehari-hari.

* + 1. **Implementasi AES**

AES atau algoritma Rijndael sebagai salah satu algoritma yang penting tentu memiliki berbagai kegunaan yang sudah diaplikasikan atau diimplementasikan di kehidupan sehari-hari yang tentu saja membutuhkan suatu perlindungan atau penyembunyian informasi di dalam prosesnya. **Salah satu contoh penggunaan AES adalah pada kompresi 7-Zip.** Salah satu proses di dalam 7-Zip adalah mengenkripsi isi dari data dengan menggunakan metode AES-256. Yang kuncinya dihasilkan melalui fungsi *Hash*. Perpaduan ini membuat suatu informasi yang terlindungi dan tidak mudah rusak terutama oleh virus yang merupakan salah satu musuh besar dalam dunia komputer dan informasi karena sifatnya adalah merusak sebuah data.

**Hal yang serupa digunakan pada WinZip sebagai salah satu perangkat lunak yang digunakan untuk melakukan kompresi.** Tapi prinsip kompresi pun tidak sama dengan prinsip enkripsi. Karena kompresi adalah mengecilkan ukuran suatu data, biasanya digunakan kode Huffman dalam melakukan hal tersebut. Contoh penggunaan lain adalah pada perangkat lunak *DiskCryptor* yang kegunaannya adalah mengenkripsi keseluruhan isi disk/partisi pada sebuah komputer. Metode enkripsi yang ditawarkan adalah menggunakan AES-256, Twofish, atau Serpent.

* + 1. **Kelebihan dan Kekurangan AES**

**Kelebihan AES:**

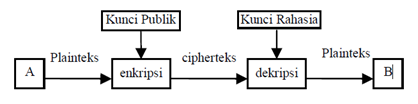
1. Algoritma ini dirancang sehingga proses enkripsi/deskripsi membutuhkan waktu yang singkat.
2. Ukuran kunci relatif lebih pendek.
3. Algoritmanya bisa menghasilkan cipher (sebuah algoritma untuk menampilkan enkripsi dan kebal akan deskripsi, serangkaian langkah yang terdefinisi yang diikuti sebagai prosedur) yang lebih kuat.
4. Autentifikasi pengiriman pesan langsung diketahui dan *ciphertext* yang diterima, karena kuncinya diketahui oleh pengirim dan penerima pesan.

**Kekurangan AES:**

1. Kunci harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci ini.
2. Kunci harus sering diubah, mungkin pada setiap sesi komunikasi.
   1. **RSA (*Rivest-Shamir-Adleman*)**
      1. **Definisi Algoritma RSA**

Sandi RSA merupakan algoritma kriptografi kunci publik (asimetris). Ditemukan pertama kali pada tahun 1977 oleh R. Rivest, A. Shamir, dan L. Adleman. Nama RSA sendiri diambil dari ketiga penemunya tersebut.

**Sebagai algoritma kunci publik, RSA mempunyai dua kunci, yaitu kunci publik dan kunci rahasia.** Kunci publik boleh diketahui oleh siapa saja, dan digunakan untuk proses enkripsi. Sedangkan kunci rahasia hanya pihak-pihak tertentu saja yang boleh mengetahuinya, dan digunakan untuk proses dekripsi. Keamanan sandi RSA terletak pada sulitnya memfaktorkan bilangan yang besar. Sampai saat ini RSA masih dipercaya dan digunakan secara luas di internet.

****

**Gambar – Skema Algoritma Kunci Publik**

**Sandi RSA terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses deskripsi.** Sebelumnya diberikan terlebih dahulu beberapa konsep perhitungan matematis yang digunakan RSA.

* + 1. **Konsep RSA**
       1. ***Public Key – Cryptography***

Konsep fundamental dari *Public Key – Cryptography* ditemukan oleh Whitfield Diffie dan Martin Hellman, dan secara terpisah oleh Ralph Merkle. Sedangkan konsep dasar *Public Key – Cryptography* terletak pada pemahaman bahwa *keys* selalu berpasangan: ***encryption key*** dan ***decryption key*.** Juga perlu diingat bahwa sebuah *key* tidak dapat digenerate dari *key* lainnya. Pemahaman *encryption* dan *decryption key* sering disebut sebagai *public* dan *private key*. Seseorang harus memberikan *public* *key*-nya agar pihak lain dapat meng-*encrypt* sebuah pesan. *Decryption* hanya terjadi jika seseorang mempunyai *private key*.

* + - 1. ***Scenario***

Bagian ini menjelaskan skenario bagaimana *public-key cryptosystem* bekerja. Kami akan menggunakan partisipan klasik Alice dan Bob sebagai orang-orang yang melakukan pertukaran informasi.

1. Alice dan Bob setuju untuk menggunakan *public-key cryptosystem*.
2. Bob mengirimkan *public key*-nya kepada Alice.
3. Alice meng-*encrypt* pesan yang dibuatkan dengan menggunakan *public key* milik Bob dan mengirimkan pesan yang sudah di-*encrypt* kepada Bob.
4. Bob meng-*encrypt* pesan dari Alice menggunakan *private key* miliknya.
   * + 1. ***Mathematical Notation***

Untuk memahami algoritma RSA, seseorang harus memahami beberapa notasi matematika dasar, teori dan formula. Hal tersebut dibutuhkan untuk mendukung semua kalkulasi yang dilakukan dalam algoritma RSA.

1. **Modulo** (didenotasikan dengan 'x mod m' atau 'x % m' dalam beberapa bahasa komputer)

x % m = x mod m = pembagian x dengan m dan mengambil sisanya.

Contoh : 25 mod 5 = 0 karena 5 habis membagi 25

                25 mod 4 = 1 karena 25 / ( 4 \* 6 ) menyisakan 1

                x  mod m = x jika dan hanya jika x < m

1. **GCD(A,B)**

GCD adalah *Greatest Common Divisor*.

GCD(A,B)   = D

GCD(78,32) = 2, karena tidak ada bilangan yang lebih besar dari dua yang membagi 78 dan 32.

GCD(A,B) dapat ditemukan dengan menggunakan algoritma *extended euclid*. Jika GCD(A,B) = 1 maka A and B adalah *coprime* satu sama lainnya (dengan kata lain, A dan B adalah *relatively prime*).

1. ***Key Generation***

Misalkan Alice ingin Bob mengirimnya sebuah pesan melalui jalur yang aman. Alice akan memberikan *public key*-nya kepada Bob dan menyimpan *private key* untuk dirinya.

1. Pilih 2 bilangan prima besar seperti p,q dimana p tidak sama dengan q
2. Hitung M = p x q
3. Hitung phi(M) = phi(p) \* phi(q)
4. Pilih sebuah integer 'e' dimana 1 < e < phi(M) dan 'e' serta phi(M) adalah *coprime*
5. Hitung 'd' integer sehingga (d \* e) mod M = 1
6. (M,e) adalah public key dimana M adalah modulo dan e adalah eksponen *encryption*
7. (M,d) adalah private key dimana M adalah modulo dan d adalah eksponen *decryption*
8. ***Decrypting Message***

Misalkan Alice menerima sebuah pesan ter-*encrypt,* ia akan men-*decrypt*-nya menggunakan tahapan-tahapan berikut ini:

1. Alice mempunyai *private key* dari langkah-langkah diatas (M,d)
2. N = C^d (mod M)
3. N adalah bilangan. Gunakan konversi table alphabet untuk mengubah N menjadi karakter yang direpresentasikan
   * 1. **Cara Kerja Algoritma RSA**

|--------------------------------------------------------------------------------------------------|

**Jika diketahui :**

p = 3

q = 7

**Penyelesaian :**

n = p.q

= 3 x 7

= **21**

M = (p-1) (q-1)

= (3 -1)(7 – 1)

= **12**

e \* d mod 12 = 1 (Cari bilangan prima yang jika mod M hasilnya 1)

**e = 5**

**d = 17**

*Public Key* = (e,n) = (5,21)

*Private Key* = (d,n) = (17,21)

S I S J A R

83 73 83 74 65 82

|--------------------------------------------------------------------------------------------------|

**Proses Enkripsi dan Deskripsi**

S = 83

|  |  |
| --- | --- |
| Enkripsi | Deskripsi |
| C = M ^ e mod n | M = C ^  d mod n |
| 8 ^ 5 mod 21 = 8 | 8 ^ 17 mod 21 = 8 |
| 3 ^ 5 mod 21 = 12 | 12 ^ 17 mod 21 = 3 |

I = 73

|  |  |
| --- | --- |
| Enkripsi | Deskripsi |
| C = M ^ e mod n | M = C ^  d mod n |
| 7 ^ 5 mod 21 = 7 | 7 ^ 17 mod 21 = 7 |
| 3 ^ 5 mod 21 = 12 | 12 ^ 17 mod 21 = 3 |

S = 83

|  |  |
| --- | --- |
| Enkripsi | Deskripsi |
| C = M ^ e mod n | M = C ^  d mod n |
| 8 ^ 5 mod 21 = 8 | 8 ^ 17 mod 21 = 8 |
| 3 ^ 5 mod 21 = 12 | 12 ^ 17 mod 21 = 3 |

J = 74

|  |  |
| --- | --- |
| Enkripsi | Deskripsi |
| C = M ^ e mod n | M = C ^  d mod n |
| 7 ^ 5 mod 21 = 7 | 7 ^ 17 mod 21 = 7 |
| 4 ^ 5 mod 21 = 16 | 16 ^ 17 mod 21 = 4 |

A = 64

|  |  |
| --- | --- |
| Enkripsi | Deskripsi |
| C = M ^ e mod n | M = C ^  d mod n |
| 6 ^ 5 mod 21 = 6 | 6 ^ 17 mod 21 = 6 |
| 4 ^ 5 mod 21 = 16 | 16 ^ 17 mod 21 = 4 |

R = 82

|  |  |
| --- | --- |
| Enkripsi | Deskripsi |
| C = M ^ e mod n | M = C ^  d mod n |
| 8 ^ 5 mod 21 = 8 | 8 ^ 17 mod 21 = 8 |
| 1. ^ 5 mod 21 = 11 | 1. ^ 17 mod 21 = 2 |

* + 1. **Kelebihan dan Kekurangan RSA**

**Kelebihan RSA:**

1. Hanya *private key* yang harus benar-benar rahasia/aman.
2. Sangat jarang untuk perlu merubah *public key* dan *private key*.

**Kelemahan RSA:**

1. Ukuran kunci lebih besar daripada *symmetric key*.
2. Tidak adanya jaminan bahwa *public key* benar-benar aman.
   1. **MD5 (*Message Digest 5*)**
      1. **Definisi Algortima MD5**

MD5 adalah salah satu dari serangkaian algortima (merupakan salah satu fungsi Hash) *message digest* yang didesain oleh Profesor Ronald Rivest dari MIT (Rivest, 1994). Saat kerja analitik menunjukkan bahwa pendahulu MD5, yaitu MD4 mulai tidak aman, MD5 kemudian didesain pada tahun 1991 sebagai pengganti dari MD4 (kelemahan MD4 ditemukan oleh Hans Dobbertin).

* + 1. **Sejarah MD5**

Sejarah singkat MD5 di mulai pada tahun 1991 yang didesain oleh Prof. Ronald Rivest dari universitas di Amerika Serikat yaitu MIT, Prof. Ronald Rivest mendesain MD5 karena telah ditemukan kelemahan pada MD4 yang ditemukan Hans Dobbertin. Pada Tahun 1996 Hans Dobbertin menemukan sebuah kerusakan/celah pada fungsi kompresi MD5, namun hal ini bukanlah serangan terhadap hash MD5 sepenuhnya, sehingga dia mengumumkan untuk para pengguna kriptografi menganjurkan supaya mengganti dengan WHIRLPOOL, SHA-1, atau RIPEMD-160.

Namun lambat laun MD5 sudah tidak bisa diandalkan lagi karena hash hasil *encrypt* MD5 mulai menampakkan kerusakannya dan sudah diketahui rahasia algoritma pada MD5, hal tersebut ditemukan kerusakannya pada tanggal 17 Agustus 2004 oleh Xiaoyun Wang, Dengguo Feng, Xuejia Lay dan Hongbo Yu, kalau dilihat dari namanya mereka berasal dari negri tirai bambu China, sekedar info saja bahwa serangan yang mereka lakukan untuk bisa men-*decrypt* hash MD5 ke *plain text* hanya membutuhkan waktu satu jam saja, dengan menggunakan IBM P690 *cluster*.

* + 1. **Cara Kerja MD5**

Cara kerja kriptografi algoritma MD5 adalah menerima *input* berupa pesan dengan ukuran sembarang dan menghasilkan *message diggest* yang memiliki panjang 128 bit. Berikut ilustrasi gambar dari pembuatan *message diggest* pada kriptografi algoritma MD5.



**Gambar – Pembuatan *Message Digest* dengan Algoritma MD5**

**Menilik dari gambar diatas, secara garis besar pembuatan message digest ditempuh melalui empat langkah, yaitu :**

1. **Penambahan bit bit pengganjal**

Proses pertama yang dilakukan adalah menambahkan pesan dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti setelah menambahkan bit-bit pengganjal, kini panjang pesan adalah 64 bit kurang dari kelipatan 512. Hal yang perlu diingat adalah angka 512 muncul karena algoritma MD5 memproses pesan dalam blok-blok yang berukuran 512.

Apabila terdapat pesan dengan panjang 448 bit, maka pesan tersebut akan tetap ditambahkan dengan bit-bit pengganjal. Pesan akan ditambahkan dengan 512 bit menjadi 96 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Lalu satu hal lagi yang perlu diperhatikan adalah bahwasanya bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

1. **Penambahan nilai panjang pesan semula**

Kemudian proses berikutnya adalah pesan ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Apabila panjang pesan lebih besar dari 264 maka yang diambil adalah panjangnya dalam modulo 264. Dengan kata lain, jika pada awalnya panjang pesan sama dengan K bit, maka 64 bit yang ditambahkan menyatakan K modulo 264. Sehingga setelah proses kedua ini selesai dilakukan maka panjang pesan sekarang adalah 512 bit.

1. **Inisialisasi penyangga MD5**

Pada algoritma MD5 dibutuhkan empat buah penyangga atau *buffer*, secara berurut keempat nama penyangga diberi nama A, B, C dan D. Masing-masing penyangga memiliki panjang 32 bit. Sehingga panjang total:

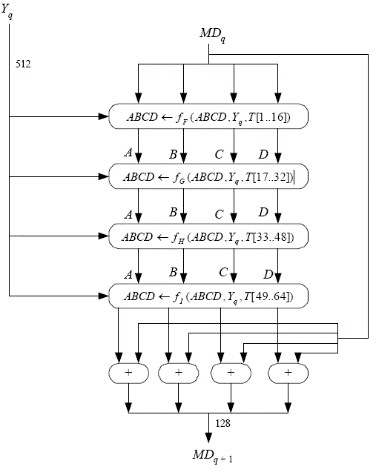


Keempat penyangga ini menampung hasil antara dan hasil akhir. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi Hexadesimal) sebagai berikut:

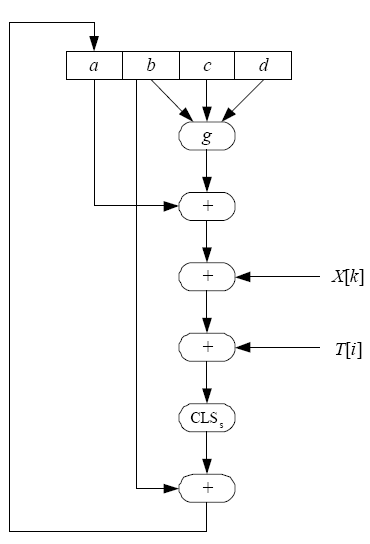


1. **Pengolahan pesan dalam blok berukuran 512 bit**

Proses berikutnya adalah pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y0 sampai YL-1). Setelah itu setiap blok 512 bit diproses bersama dengan penyangga MD yang menghasilkan keluaran 128 bit, dan ini disebut HMD5. **Berikut ini gambaran dari proses HMD5**:



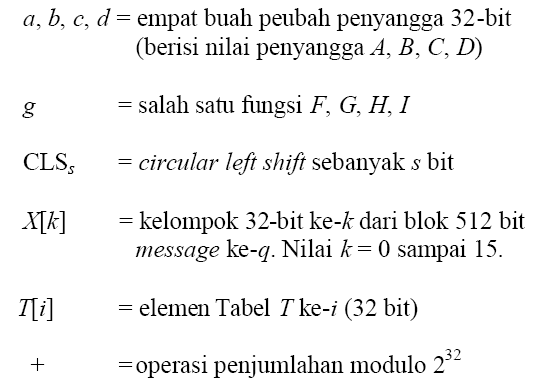
Dari gambar diatas dapat kita lihat bahwa proses HMD5 terdiri dari 4 buah putaran, dan masing-masing putaran melakukan opersi dasar MD5 sebanyak 16 kali. Dimana disetiap operasi dasar memakai sebuah elemen T. Sehingga setiap putaran memakai 16 elemen tabel T.

Pada gambar 3.2, Yq menyatakan blok 512 bit ke-q dari pesan yang telah ditambahkan dengan bit-bit pengganjal pada proses pertama dan tambahan 64 bit nilai panjang pesan semula pada proses kedua. MDq adalah nilai *message digest* 128 bit dari proses HMD5 ke-q. Pada awal proses , MDq berisi nilai inisialisasi penyangga MD. Kemudian fungsi fF, fG, fH, dan fI pada gambar, masing-masing berisi 16 kali operasi dasar terhadap input, setiap operasi dasar menggunakan elemen tabel T.

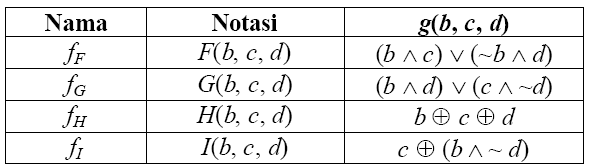
**Gambar – Operasi Dasar MD5**

Operasi dasar MD5 yang diperlihatkan gambar diatas dapat dituliskan dengan persamaan berikut ini:

https://lh3.googleusercontent.com/yc1r6MUqLhvl51nGPUE-Rv5PUhO24HhD5F0jrYruf-7VTGyBPSu-ym-Qy8ogAS3LXakZf-AMHZ_rPlVJVf-lYQ4YnwGb6xeMpHmfJi_grcO8yi_Hknjyse1ArY7CFa4EDHaRF5Pxoeeup1mplg

Dimana:

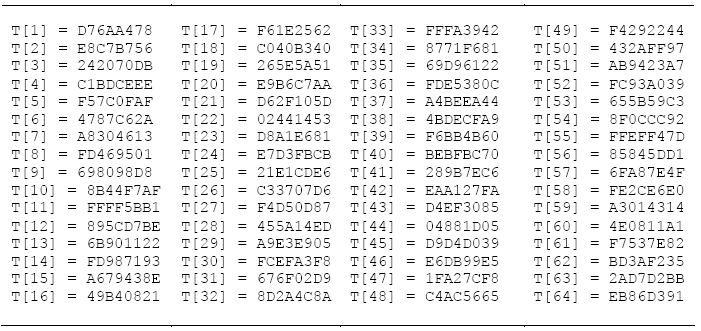
Fungsi fF, fG, fH, dan fI adalah fungsi untuk memanipulasi masukan a, b, c, dan d dengan ukuran 32-bit. Masing-masing fungsi dapat dilihat pada Tabel 1 dibawah ini:



**Tabel 1. Fungsi-fungsi dasar MD5**

**Keterangan:** Secara berurut peartor logika AND, OR, NOT dan XOR dilambangkan dengan ∧, ∨, ~, ⊕

Kemudian nilai T[i] dapat dilihat pada tabel dibawah ini. Tabel ini disusun oleh fungsi 232 x abs(sin(i)), i dalam radian.



**Tabel 2. Nilai T[1]**

Sebagaimana telah dijelaskan sebelumnya bahwa fungsi fF, fG, fH, dan fI melakukan 16 kali operasi dasar. Misalkan notasi berikut ini:

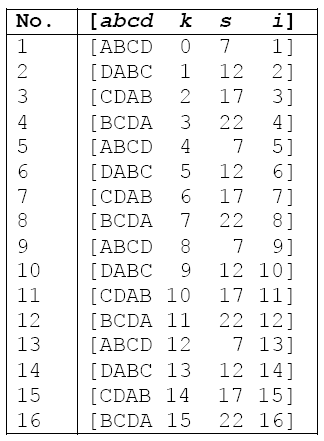
https://lh4.googleusercontent.com/KNYqQeuE7ee-uetp4ZzmTtRqVgftNTzZy211ck5GzAYW_1_4tMhQMI9wG3KR-igME2CAhzRDgnGkJACrAtM_w67L7xP8X40BzEdxfqcaGi4BiXtjXuQet5-ACWk8j80Q-7_FiJwRQmN_K8bmMg

Menyatakan operasi:

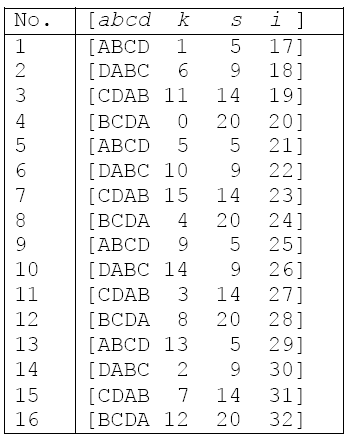
https://lh3.googleusercontent.com/3iFGid5GP_vunvEXdR1lGEOdFFKzIBDOUBO9X9Oi11mPpYIsFjjCwRqpfpBEpgLfYa-UIw6DqEe7YNzP7zQX1DER8SqeY6-Bk0yyuSynQ9_JmvaVvcDMFD01tvB1KfbefsmB4GqhcHsZdGHi8A

Untuk operasi diatas, <<<s melambangkan opersi *circular left shift* 32 bit, maka operasi dasar pada masing-masing putaran dapat ditabulasikan sebagai berikut:

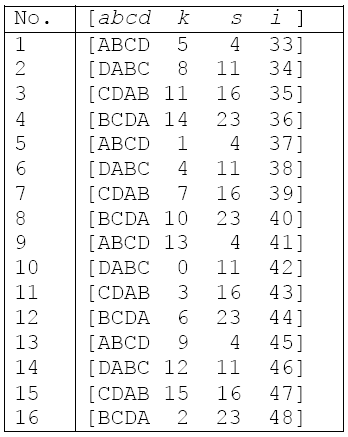
1. **Putaran 1** - 16 kali operasi dasar dengan g(b, c, d) – F(b, c, d), dapat dilihat pada tabel berikut ini.



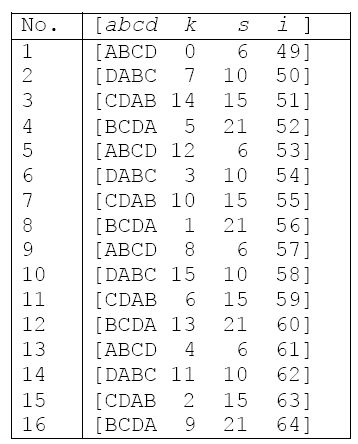
1. **Putaran 2** - 16 kali operasi dasar dengan g(b, c, d) – G(b, c, d), dapat dilihat pada tabel berikut ini.



1. **Putaran 3** - 16 kali operasi dasar dengan g(b, c, d) – H(b, c, d), dapat dilihat pada tabel berikut ini.



1. **Putaran 4 -** 16 kali operasi dasar dengan g(b, c, d) – I(b, c, d), dapat dilihat pada tabel berikut ini.

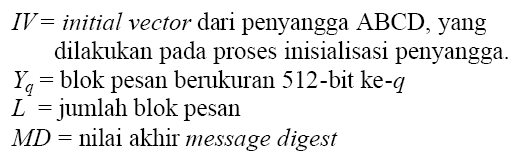


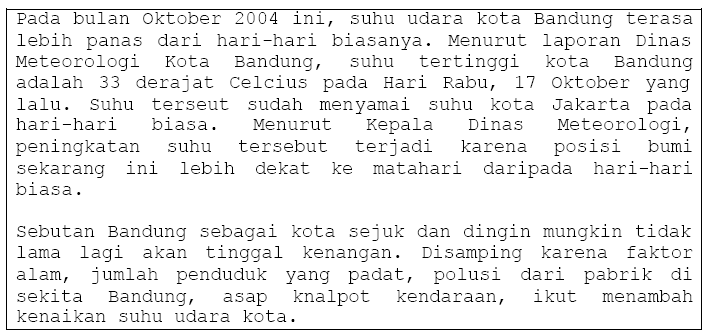
Setelah putaran keempat, a, b, c dan d di tambahkan ke A, B, C dan D yang selanjutnya algoritma akan memproses untuk blok data berikutnya (Yq+1). *Output* akhir dari algoritma MD5 adalah hasil penyambungan bit-bit di A, B, C dan D.

https://lh4.googleusercontent.com/TP3lYN8Ih19TPMiWo1LpuM_Y9NLRvsotnRzJKeryDgTUdhSQtGt6ZiKhK5KGCU5iaI3DkkRbsCpuOD3gUBtEiRHfqGajQctj8pySqPmmzdt6JZLF70Z_6_EvNeNnNfwwF9TcnsUPAUaDIN2fwQDari uraian diatas, secara umum fungsi hash MD5 dapat ditulis dalam persamaan matematis sebagai berikut.

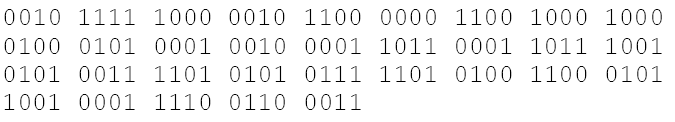
https://lh6.googleusercontent.com/fE3htJqzjmPU1ReBuwuV1-YmDqd378MA37gabrJWfSvtpbRsEDsdnjtF4q_LHGJxMQWUXfvrTVh-OyZD5-PSQHit2etBAhr1Ka9jyq0K7y0PY7I2xjErwA4bk91IwstIuf1LsKbpeL344VTvGg

https://lh6.googleusercontent.com/K7bVR59pKvc-YIIUU0oKrxmw0aF6KLlJiRVhDGuP_PA3NFYYgDo7EwZGj0gSEYhzuWx1m0-tRPOyubtNmuta7_1opt7KAqsrloHYBcWIdOdjBYUCg2ZwlaF_YkTz5mrmMCG4iBM5yYwsXJo3HQ

Dimana:

Agar lebih mudah dipahami, berikut ini contoh penerapan kriptografi MD5 pada suatu pesan yang ingin dirahasiakan. Misalkan terdapat sebuah arsip dengan nama **bandung.txt** sebagai berikut:

**Message digest yang dihasilkan dari arsip diatas adalah untuk 128 bit :**



**Lalu untuk notasi hexadesimal :**

https://lh6.googleusercontent.com/3fEIyvxnVCkaeGo5n1x7sWPueH3dJkoI6ZvcBwu6TE1dRB2Wjdy-ow_WSVIwOUSQIycmeHlYf5TiwlAn37CDHP4n2Lumw9ufwNL-QUXPRDAEFQEksRFHkEBPBHqtbxanXVzeRyS_4r6LbGdSuA

* + 1. **Implementasi MD5**

Implementasi algoritma kriptografi MD5 dapat digunakan untuk menjaga keintegritasan data. Dengan aplikasi fungsi hash yang menjadi asas algoritma MD5, perubahan kecil pada data sekalipun dapat terdeteksi. Langkah yang harus ditempuh adalah bangkitkan *message digest* dari isi arsip menggunakan algoritma MD5. Kemudian gabung *message digest* ke dalam arsip. Verifikasi isi arsip dapat dilakukan secara berkala dengan membandingkan MD isi arsip sekarang dan MD dari arsip asli (MD yang telah disimpan sebelumnya). Jika hasilnya berbeda maka telah terjadi perubahan pada arsip. Aplikasi ini didasarkan pada kenyataan bahwa perubahan 1 bit pesan akan mengubah, secara rata-rata, setengah dari bit-bit *message digest*. Dengan kata lain, algoritma MD5 dengan fungsi hashnya sangat peka terhadap perubahan sekecil apapun pada data masukan.

* + 1. **Kelebihan dan Kekurangan MD5**

**Kelebihan MD5:**

1. Untuk memeriksa integritas file dalam berbagai situasi.
2. Sebagai fungsi enkripsi sertifikat SSL.
3. Menghasilkan tanda tangan digital sepanjang 32 karakter, tanpa tergantung panjang *input*.
4. Hasil *output* tidak akan sama untuk input yang berbeda.
5. Sulit untuk dipecahkan walaupun dengan serangan *brute force*, tingkat keamanan MD5 adalah salah satu yang terbaik, tidak bisa diubah kembali.

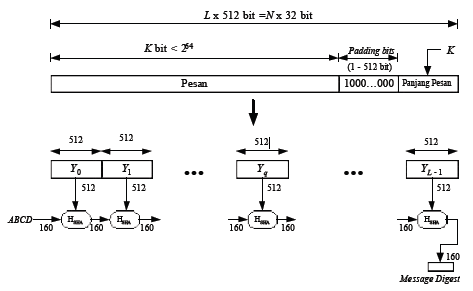
**Kelemahan MD5:**

1. Serangan *collision*.
2. Kriptanalisis lebih lanjut terhadap MD5.
3. *Preimage Attack*.
   1. **SHA-1 (*Secure Hash Algorithm*)**
      1. **Definisi SHA-1**

SHA adalah fungsi *hash* satu-arah yang dibuat oleh NIST dan digunakan bersama DSS (*Digital Signature Standard*). Oleh NSA, SHA dinyatakan sebagai standard fungsi *hash* satu-arah. SHA didasarkan pada MD4 yang dibuat oleh Ronald L. Rivest dari MIT. SHA disebut aman (*secure*) karena ia dirancang sedemikian sehingga secara komputasi tidak mungkin menemukan pesan yang berkoresponden dengan *message digest* yang diberikan.

* + 1. **Cara Kerja SHA-1**

Algoritma SHA menerima masukan berupa pesan dengan ukuran maksimum 264 bit (2.147.483.648 *gigabyte*) dan menghasilkan *message digest* yang panjangnya 160 bit, lebih panjang dari *message digest* yang dihasilkan oleh MD5. Gambaran pembuatan *message digest* dengan algoritma SHA diperlihatkan pada gambar dibawah:



**Gambar – Pembuatan Message Digest dengan Algoritma SHA**

Algoritma SHA mengambil pesan yang panjangnya kurang dari 264 bit dan menghasilkan *message digest* 160-bit. Sehingga algoritma ini lebih lambat daripada MD5, namun *message digest* yang lebih besar membuatnya semakin aman dari *bruteforce collison* dan serangan inversi.

**Langkah-langkah pada SHA-1 adalah sebagai berikut:**

1. Melakukan *padding* terhadap pesan sehingga panjangnya adalah 448 modulus 512.
2. 64 bit sisanya adalah representasi biner dari panjang pesan.
3. Melakukan inisialisasi 5 word *buffer* (160 bit) A, B, C, D, dan E dengan nilai A = 67452301, B = efcdab89, C = 98badcfe, D = 10325476, dan E = c3d2e1f0.
4. Memproses pesan dalam blok-blok 16 *word* (512 bit) dengan ketentuan:
5. Ekspansi 16 *words* menjadi 80 *words* dengan teknik *mixing* dan *shifting*.
6. Menggunakan 4 *round* dari 20 operasi bit pada blok pesan dan *buffer*.
7. Menambahkan *output* dengan *input* untuk memperoleh nilai *buffer* yang baru.
8. Output nilai hash adalah nilai terakhir dari *buffer*.

Jadi intinya SHA-1 adalah suatu algoritma hash yang memetakan inputan string dengan panjang sembarang menjadi *output* dengan panjang tetap yaitu 160 bit.

* + 1. **Implementasi SHA-1**

SHA-1 digunakan pada aplikasi atau *system* keamanan dan protokol seperti:

* TLS dan SSL (*Transport Layer Security/Secure Socket Layer*) merupakan kelanjutan dari protokol kriptografi yang menyediakan komunikasi yang aman di internet. Secara umum *site* yang menggunakan SSL/TLS ini bisa dilihat dari URL yang digunakan yakni menggunakan https (https://) atau bisa juga kita lihat *icon* gembok/kunci yang berada pada *browser* yang digunakan.
* PGP (*Pretty Good Privacy*) adalah suatu program komputer yang dikembangkan oleh Phil Zimmermann pada pertengahan tahun 1980 yang memungkinkan seseorang untuk saling bertukar pesan melalui email dan juga file dengan memberikan perlindungan kerahasiaan berupa enkripsi dan otentikasi berupa *digital signature* (tanda tangan digital).
* SSH (*Secure Shell*) adalah protokol jaringan yang memungkinkan pertukaran data secara aman antara dua komputer. SSH dapat digunakan untuk mengendalikan komputer dari jarak jauh dalam mengirim file, membuat tunnel yang terenkripsi dan lain-lain.
* MIME (*Multipurpose Internet Mail Extensions*) merupakan tambahan fungsi mail dari internet yang memungkinkan kita menyisipkan data yang bentuknya bukan tulisan dalam email, misalnya berbentuk gambar atau suara.
* IPsec (*Internet Protocol Security*) adalah sebuah *framework* standar yang dikembangkan oleh IETF (*Internet Engineering Task Force*). Ipsec juga menyediakan keamanan untuk transmisi informasi yang bersifat sensitive melalui jaringan yang tanpa proteksi dan bebas seperti internet.
  + 1. **Kelebihan dan Kekurangan SHA-1**

**Kelebihan SHA-1**:

1. ***Validate a password* (memvalidasi password)**

Nilai hash dari password akan disimpan, kemudian ketika password diotentifikasi, maka password yang dimasukkan oleh *user* akan dihitung hashnya dan jika hashnya sesuai maka password dinyatakan valid. Namun untuk mendapatkan password yang asli tidak dapat diperoleh dari hash yang telah disimpan.

1. ***Challenge handshake authentication***

Untuk menghindari kesalahan pengiriman password dalam kondisi “*clear*”, *client* dapat mengirim nilai hash sebuah password melalui internet untuk divalidasi oleh server tanpa beresiko disadapnya password yang asli.

1. ***Anti-tamper***

Untuk memastikan data tidak berubah selama ditransmisikan. Penerima akan menghitung nilai hash dan mencocokan dengan hash yang dikirimkan, apabila nilainya sama berarti data yang dikirimkan tidak berubah.

1. ***Digital Signature***

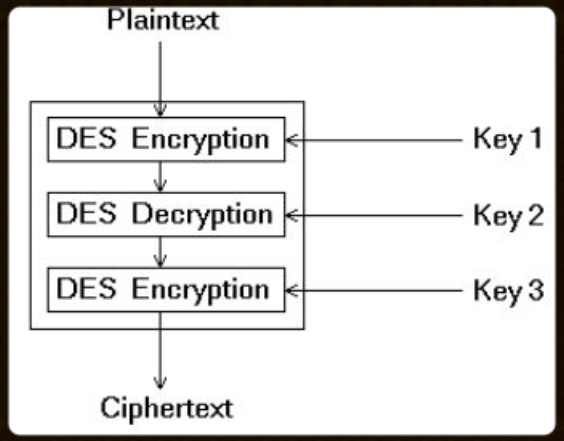
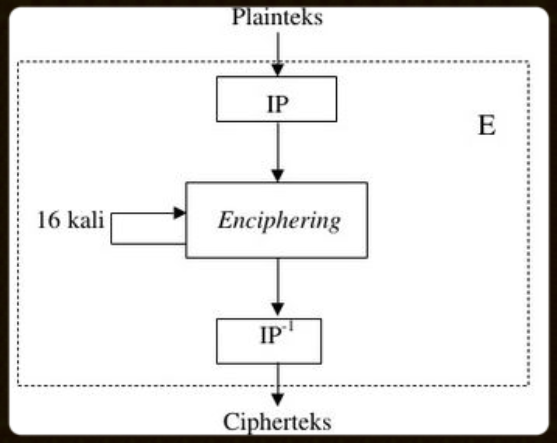
Dilakukan dengan cara mengenkrip nilai hash sebuah dokumen dengan menggunakan *private key*, sehingga menghasilkan tanda tangan digital untuk dokumen tersebut. Orang lain dapat mengecek otentifikasi dokumen tersebut dengan cara mendekrip tanda tangan tersebut menggunakan *public key* untuk mendapatkan nilai hash yang asli dan membandingkannya dengan nilai hash dari teks.

**Kekurangan SHA-1:**

*Avalanche effect* dapat terjadi hanya dengan mengubah sedikit bagian dari pesan. Akan tetapi, algortima SHA-1 masih dapat ditemui *collision*, sehingga dianggap masih kurang aman. SHA-256 dianggap lebih aman dibanding SHA-1 karena tingkat kompleksitas yang lebih tinggi dan *message digest* yang dihasilkan lebih panjang.

* 1. **Triple DES (*Triple Data Encryption Standard*)**
     1. **Definisi Algoritma Triple DES**

Triple DES atau *Triple Data Encryption Standard* merupakan salah satu algoritma simetris chiper blok. Algoritma Triple DES ini adalah suatu pengembangan dari algoritma DES (*Data Encryption Standard*).



**Gambar – DES (sebelah kiri) dan Triple *DES* (sebelah kanan)**

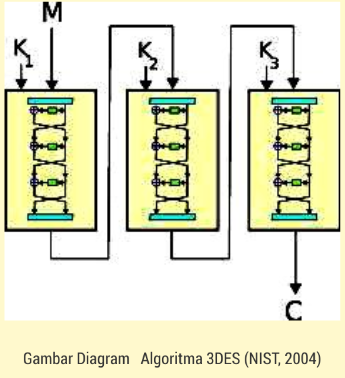
* + 1. **Sejarah Triple DES**

DES dianggap sudah tidak aman lagi, karena dengan perangkat keras khusunya kuncinya dapat ditemukan dalam waktu beberapa hari. Tingkat kerahasiaan algoritma Triple DES terletak pada panjangnya kunci yang digunakan, maka penggunaan algoritma dengan algoritma DES.

DES beroperasi pada ukuran blok 64-bit. DES mengenkripsikan 64-bit *plainteks* menjadi 64-bit *cipherteks* dengan menggunakan 56-bit kunci internal yang dibangkitkan dari kunci eksternal yang panjangnya 64-bit.

* + 1. **Cara Kerja Triple DES**

Pada dasarnya algoritma yang digunakan sama, hanya pada Triple DES dikembangkan dengan melakukan enkripsi dengan implementasi melakukan enkripsi dengan implementasi algoritma DES sebanyak tiga kali. Triple DES memiliki tiga buah kunci yang berukuran 168-bit (tiga kali kunci 56-bit dari DES). Pada algoritma Triple DES dibagi menjadi tiga tahap, setiap tahapnya merupakan implementasi dari algoritma DES.



**Gambar – Diagram Algoritma Triple DES**

* + - 1. ***Triple DES with 2-Key***

**Langkah-langkah yang harus dilakukan, yaitu:**

* Gunakan tiga tahap DES untuk enkripsi dari deskripsi.
* Tahap 1, tahap 3 menggunakan tombol 𝐾1 dan menggunakan tombol 𝐾22*nd stage*.
* Untuk membuat *triple* DES yang kompatibel dengan DES tunggal, tahap tengah menggunakan dekripsi di sisi enkripsi dan enkripsi di sisi dekripsi.
* Ini jauh lebih kuat dari *double DES*.

**Gambar – *Triple* DES *with* 2-*Key***

* *The function follows an encrypt-decrypt-encrypt (EDE) sequence:*

*C = E(𝐾1, D(𝐾2, E(𝐾1, P)))*

*P = D(𝐾1, E(𝐾2, D(𝐾1, C)))*

* Dengan penggunaan *triple* DES dengan enkripsi 2 kunci, ia meningkatkan biaya serangan *meet-in-the-middle* ke 2112.
* Ini memiliki kekurangan membutuhkan panjang kunci 56x3 = 168 bits yang mana mungkin agak berat.
  + - 1. ***Triple DES with 3-Key***
* Meskipun serangan yang baru saja dijelaskan tampak tidak praktis, siapapun yang menggunakan dua tombol Triple DES mungkin merasa khawatir.
* Dengan demikian, banyak penelitian sekarang merasa bahwa 3-*key* alternatif.
* Gunakan tiga tahap DES untuk enkripsi dan deskripsi dengan tiga tombol yang berbeda.
* Triple DES 3-kunci memiliki panjang kunci efektif 168 bit dan didefinisikan sebagai:

C = E (𝐾3, D (𝐾2, E (𝐾1, P)))  
P = D (𝐾1, E (𝐾2, D (𝐾3, C)))

**Gambar – *Triple DES* 3-*Key***

* + 1. **Kelebihan dan Kekurang Triple DES**

**Kelebihan Triple DES:**

1. Lebih aman dibandingkan algoritma kriptografi DES.
2. Tahan terhadap serangan *meet-in-the-middle* dan *brute force attack* karena panjang kuncinya.

**Kekurangan Triple DES:**

Algoritma Triple DES walaupun aman namun pemanfaatannya saat ini semakin berkurang. Hal ini disebabkan oleh waktu performansi algoritma ini dan algoritma DES yang kurang baik ketika diterapkan dalam *software*. Sehingga saat ini pemanfaatan terbesar algortima ini terletak pada implementasi *hardware*.

* 1. **DSA (*Digital Signature Algorithm*)**
     1. **Definisi Algoritma DSA**

Tanda tangan digital adalah mekanisme otentikasi yang mengijinkan pemilik pesan membubuhkan sebuah sandi pada pesannya yang bertindak sebagai tanda tangan. Tanda tangan dibentuk dengan mengambil nilai hash dari pesan dan mengenkripsi nilai hash pesan tersebut dengan kunci privat pemilik pesan (Stallings, 2005). Prinsip yang digunakan dalam tanda tangan digital ini adalah dokumen yang dikirimkan harus ditandatangani oleh pengirim dan tanda tangan bisa diperiksa oleh penerima untuk memastikan keaslian dokumen yang dikirimkan. Fungsinya adalah untuk melakukan validasi terhadap data yang dikirim. Tanda tangan digital menggunakan algoritma yang disebut dengan istilah *hashing algorithm*. Fungsi tersebut akan menghasilkan sebuah kombinasi karakter yang yang unik yang disebut *Message Digest*. Dengan cara ini pengirim bertanggung jawab terhadap isi dokumen dan dapat di cek keaslian dokumen oleh penerima. Keunikannya adalah jika di tengah perjalanan data mengalami modifikasi, penghapusan maupun di sadap diam-diam oleh *hacker* walaupun hanya 1 karakter saja, maka *message digest* yang berada pada si penerima akan berbeda dengan yang dikirimkan pada awalnya. Keunikan lainnya adalah *message digest* tersebut tidak bisa dikembalikan lagi ke dalam bentuk awal seperti sebelum disentuh dengan fungsi algoritma, sehingga disebutlah sebagai *one-way* hash (Sianturi, 2008). Fungsi utama dari tanda tangan digital pada pada aspek keamanan kriptografi adalah non-repudiation atau anti penyangkalan dimana apabila dokumen valid maka pengirim tidak bisa menyangkal bahwa keberadaan dokumen benar dikirim oleh pengirim yang bersangkutan. Suatu tanda tangan digital dapat digunakan di segala macam pesan, apakah itu terenkripsi maupun tidak, sehingga penerima dapat memastikan identitas pengirim itu dan pesan tiba secara utuh.

**Keunggulan Digital Signature**

* Memberikan keaslian, integritas dan ketidaktaatan pada dokumen elektronik.
* Untuk menggunakan internet sebagai media yang aman dan nyaman untuk *e-Governance* dan *e-Commerce*.

**Sifat-sifat yang dimiliki tanda tangan digital**

1. **Otentikasi**

Otentik berarti tidak bisa bahkan sulit untuk ditiru oleh orang lain. Pesan dan tanda tangan pesan tersebut juga dapat menjadi barang bukti sehingga penanda tangan takbisa menyangkal bahwa dulu ia tidak pernah menandatanganinya. Meskipun pesan seringkali dapat mencakup informasi tentang entitas mengirim pesan, bahwa informasi mungkin tidak akurat. Tanda tangan digital dapat digunakan untuk otentikasi sumber pesan. Ketika kepemilikan kunci rahasia tanda tangan digital terikat kepada pengguna tertentu, tanda tangan yang sah menunjukkan bahwa pesan yang dikirim oleh pengguna tersebut. Pentingnya kepercayaan yang tinggi dalam otentisitas pengirim ini terutama jelas dalam konteks keuangan. Misalnya, kantor cabang bank mengirimkan instruksi ke kantor pusat meminta perubahan saldo *account*. Apabila kantor pusat tidak yakin bahwa pesan tersebut benar-benar dikirim dari sumber resmi, bertindak atas permintaan semacam itu bisa menjadi kesalahan besar.

1. **Integritas**

Dalam skenario banyak, pengirim dan penerima pesan mungkin memiliki kebutuhan untuk keyakinan bahwa pesan belum diubah selama transmisi. Meskipun menyembunyikan enkripsi isi pesan, dimungkinkan untuk mengubah sebuah pesan terenkripsi tanpa memahaminya (algoritma enkripsi beberapa, yang dikenal sebagai *nonmalleable* yang mencegah hal ini tetapi yang lain tidak). Namun, jika pesan secara digital ditandatangani, setiap perubahan dalam pesan setelah tanda tangan akan membatalkan tanda tangannya. Selain itu, tidak ada cara yang efisien untuk memodifikasi pesan dan tanda tangan untuk menghasilkan pesan baru dengan tanda tangan yang sah, karena ini masih dianggap layak oleh sebagian besar komputasi fungsi hash kriptografi (lihat resistensi tabrakan).

1. ***Non-repudiation***

*Non-repudiation*, atau lebih khusus non-repudiation asal, merupakan aspek penting dari tanda tangan digital. Dengan properti ini suatu entitas yang telah menandatangani beberapa informasi tidak dapat di lain waktu menyangkal memiliki menandatanganinya. Demikian pula, akses ke kunci publik hanya tidak memungkinkan pihak penipuan untuk palsu tanda tangan valid.

* + 1. **Sejarah DSA**

Pada tahun 1976, Whitfield Diffie dan Martin Hellman pertama kali menggambarkan gagasan tentang skema tanda tangan digital, walaupun mereka hanya menduga bahwa skema semacam itu ada berdasarkan fungsi peralihan satu arah jalan jebakan. Segera setelah itu, Ronald Rivest, Adi Shamir, dan Len Adleman menemukan algoritma RSA, yang dapat digunakan untuk menghasilkan tanda tangan digital primitif (walaupun hanya sebagai bukti konsep tanda tangan RSA "polos" tidak aman). Paket perangkat lunak pertama yang dipasarkan untuk menawarkan tanda tangan digital adalah Lotus Notes 1.0, dirilis pada tahun 1989, yang menggunakan algoritma RSA. Skema tanda tangan digital lainnya segera dikembangkan setelah RSA, tanda tangan Lamport yang paling awal, tanda tangan Merkle (juga dikenal sebagai "pohon Merkle" atau hanya "pohon Hash"), dan tanda tangan Rabin.

Sedangkan pada tahun 1984, Shafi Goldwasser, Silvio Micali, dan Ronald Rivest menjadi orang pertama yang secara ketat menentukan persyaratan keamanan skema tanda tangan digital. Mereka menggambarkan sebuah hirarki model serangan untuk skema tanda tangan, dan juga mempresentasikan skema tanda tangan GMR, yang pertama yang dapat terbukti mencegah pemalsuan eksistensial terhadap serangan pesan terpilih yang merupakan definisi keamanan yang berlaku saat ini untuk skema tanda tangan. Skema pertama yang tidak dibangun pada fungsi trapdoor melainkan pada keluarga fungsi dengan properti permutasi satu arah yang jauh lebih lemah disajikan oleh Moni Naor dan Moti Yung.

* + 1. **Fungsi Hash**

Fungsi hash merupakan suatu fungsi yang menerima masukan berupa string yang panjangnya sembarang dan mengonversi masukan tersebut menjadi string yang mempunyai panjang tetap (*fixed*) dan umumnya menjadi lebih kecil dari panjang semula. Keluaran dari fungsi hash disebut juga nilai hash atau pesan-ringkas (*message digest*). Fungsi hash sering juga disebut fungsi satu arah (*one way function*), *message digest*, *fingerprint*, fungsi kompresi, dan *Message Authentication Code* (MAC). Fungsi ini biasanya diperlukan bila kita menginginkan pengambilan sidik jari suatu pesan. Dinamakan fungsi kompresi karena biasanya masukan fungsi satu arah ini selalu lebih besar dari keluarannya, sehingga seolah-olah mengalami kompresi. Namun kompresi hasil fungsi ini tidak dapat dikembalikan ke asalnya sehingga disebut sebagai fungsi satu arah. Dinamakan sebagai *message digest* karena seolah-olah merupakan inti sari pesan mestinya merupakan ringkasan pesan yang masih dipahami maknanya, sedangkan disini justru sebaliknya, malahan dengan diketahuinya sidik jari ini, justru orang diharapkan tidak tahu pesan aslinya (Kurniawan, 2004).

Fungsi hash satu arah merupakan fungsi satu arah (*one way function*) yang dapat menghasilkan ciri (*signature*) dari data (berkas, *stream*). Perubahan satu bit saja akan mengubah keluaran hash secara drastis. Fungsi hash biasanya digunakan untuk menjamin integritas dan tanda tangan digital.

* + 1. **Cara Kerja DSA**

Dimulai dari pencetus pesan menggunakan tombol tanda tangan (kunci pribadi) untuk menandatangani pesan dan mengirim pesan dan tanda tangan digital ke penerima. Penerima menggunakan kunci verifikasi (kunci publik) untuk memverifikasi asal pesan dan pesan tersebut tidak dirusak saat dalam perjalanan.

Tanda tangan digital menggunakan tipe *Asymmetric Cryptography*. Skema biasanya terdiri dari tiga Algoritma yakni:

1. *A key generation algorithm*, yang memilih kunci pribadi secara seragam secara acak dari sekumpulan kunci pribadi yang mungkin. Algoritma mengeluarkan kunci privat dan kunci publik yang sesuai.
2. *A signing algorithm,* yang mana jika diberi pesan dan kunci privat, menghasilkan tanda tangan.
3. *A signature verifying alhorithm*, yang mana ketika diberi pesan, kunci publik dan tanda tangan, akan menerima atau menolak klaim pesan tersebut atas keasliannya.

Cara kerja *Digital Signature* dengan memanfaatkan dua buah kunci, yaitu kunci publik dan kunci privat. Kunci publik digunakan untuk mengenkripsi data, sedangkan kunci privat digunakan untuk mendekripsi data. Pertama, dokumen di-hash dan menghasilkan *Message Digest*. Kemudian, *Message Digest* dienkripsi oleh kunci publik menjadi *Digital Signature*. Untuk membuka *Digital Signature* tersebut diperlukan kunci privat. Bila data telah diubah oleh pihak luar maka *Digital Signature* juga ikut berubah sehingga kunci privat yang ada tidak akan bisa membukanya. Ini merupakan salah satu syarat keamanan jaringan yaitu *Authenticity*. Artinya adalah keaslian data dapat terjamin dari perubahan-perubahan yang dilakukan pihak luar. Dengan cara yang sama pengirim data tidak dapat menyangkal data yang telah dikirimkannya. Bila *Digital Signature* cocok dengan kunci privat yang dipegang oleh penerima data, maka dapat dipastikan bahwa pengirim adalah pemegang kunci privat yang sama. Ini berarti *Digital Signature* memenuhi salah satu syarat keamanan jaringan, yaitu *Nonrepudiation* atau non-penyangkalan.

* + 1. **Penggunaan Tanda Tangan Digital**

Salah satu cara yang digunakan untuk memastikan surat tersebut adalah dengan mengecek tanda tangan yang ada di dalam surat tersebut dan stempel yang menunjukkan keaslian pengirim surat. Tanda tangan digital atau yang lebih dikenal dengan digital signature mempunyai fungsi yang sama dengan tanda tangan analog yang ditulis di atas kertas. Tanda tangan digital harus unik sehingga dapat membedakanpengirim yang satu degan yang lainnya. Tanda tangan digital juga harus sulit untuk ditiru dan dipalsukan sehingga integritas dan keabsahan pesan dapat terjaga. Dengan demikian diharapkan pencatutan identitas ketika pesan atau email tersebut dikirim dapat dihindari. Tidak hanya pencatutan Untuk keperluan yang penting ini, tersedia alat bantu yang dapat diperoleh secara cumacuma, yakni *Pretty Good Privacy* (PGP) dan *Gnu Privacy Guard* atau GPG. Tentu saja masih terdapat penyedia layanan tanda tangan digital lainnya, namun PGP dan GPG lebih dikenal luas. GPG adalah produk *Open Source* yang dapat diperoleh secara gratis tanpa harus membayar lisensi. Penggunaaan PGP di luar Amerika Serikat harus menggunakan versi internasional. Sedangkan GPG sendiri karena dikembangkan di luar wilayah hukum Amerika Serikat, maka bebas digunakan oleh siapapun. Restriksi ini berkaitan dengan aturan ekspor produk enkripsi yang berkait dengan pemakaian kunci sandi untuk pemakaian tanda tangan digital ini [DIR04]. Penggunaan tanda tangan digital ini tidak terlalu sulit. Kedua belah pihak yang akan berkomunikasi harus menyiapkan sepasang kunci, yaitu kunci privat (*private key*) dan kunci publik (*public key*). Kunci privat hanya dipegang oleh pemiliknya sendiri. Sedangkan kunci publik dapat diberikan kepada siapapun yang memerlukannya.

**Otentifikasi Tanda Tangan**

Untuk melakukan verifikasi tanda tangan, pesan beserta tanda tangannya harus dimasukkan ke dalam aplikasi. Pesan tanpa tanda tangan akan diubah menjadi message digest kembali dengan menggunakan fungsi HAVAL dan diubah ke dalam *BigInteger* kemudian tanda tangan yang diberikan ke pesan tersebut didekripsi dengan algoritma RSA dan dibandingkan dengan hasil *message digest* dalam bentuk *BigInteger* tersebut. Apabila hasil dekripsi tersebut sama dengan message digest berarti pesan tersebut telah lulus otentikasi.

* + 1. **Implementasi Skema Tanda Tangan Digital**

Implementasi akan dilakukan untuk menguji skema tanda tangan digital yang telah dibuat dengan menggunakan kombinasi algoritma kriptografi kunci publik RSA dan fungsi hash HAVAL. Implementasi dilakukan dengan menggunakan bahasa pemrograman C# dan diuji pada lingkungan Windows.

**Spesifikasi Implementasi**

Aplikasi diberi nama “RSA-HAVAL *Digital Signature*” ini memiliki spesifikasi sebagai berikut:

* Dapat melakukan pembangkitan kunci publik dan privat secara acak dan menyimpannya dalam file eksternal
* Dapat menandatangani dokumen teks
* Dapat melakukan otentikasi terhadap dokumen yang telah ditandatangani

**Berikut ini daftar nama kelas yang telah diimplementasikan untuk membangun aplikasi tersebut:**

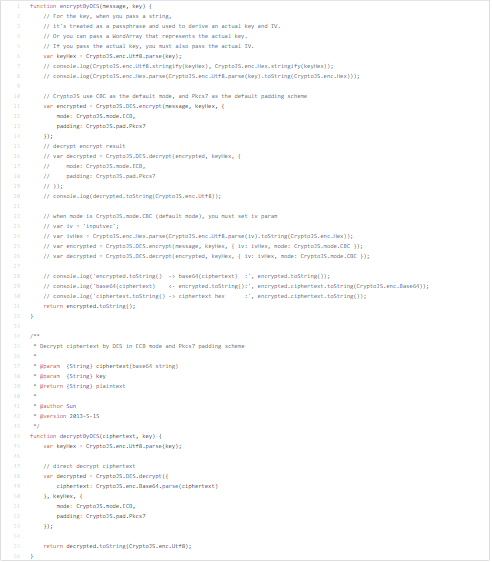
* *BigInteger*: kelas *BigInteger* dan operasinya
* GenAlgo: kelas untuk membangkitkan kunci publik dan kunci privat
* RSA: kelas yang mengimplementasikan algoritma RSA
* Haval: kelas yang mengimplementasikan algoritma fungsi hash HAVAL
* *AuthSignature*: merupakan kelas *interface* untuk melakukan otentikasi dokumen
* *DigitalSignature*: merupakan kelas *interface* utama dari aplikasi
* *KeyGen*: merupakan kelas *interface* untuk pembangkitan kunci acak
* *GiveSignature*: merupakan kelas *interface* untuk memberikan tanda tangan *KeyGen* akan secara otomatis membangkitkan kunci publik dan privat beserta nilai n yang kemudian kedua kunci tersebut disimpan ke dalam file eksternal terpisah dengan ekstensi *public key* (kunci publik) dan .*private key* (kunci privat). Algoritma RSA sendiri membutuhkan parameter yang cukup besar sehingga tidak memungkinkan untuk menggunakan integer. Oleh karena itu, aplikasi ini menggunakan kelas *BigInteger* yang telah ada dikembangkan untuk C#.
  + 1. **Kelebihan dan Kekurangan Digital Signature**

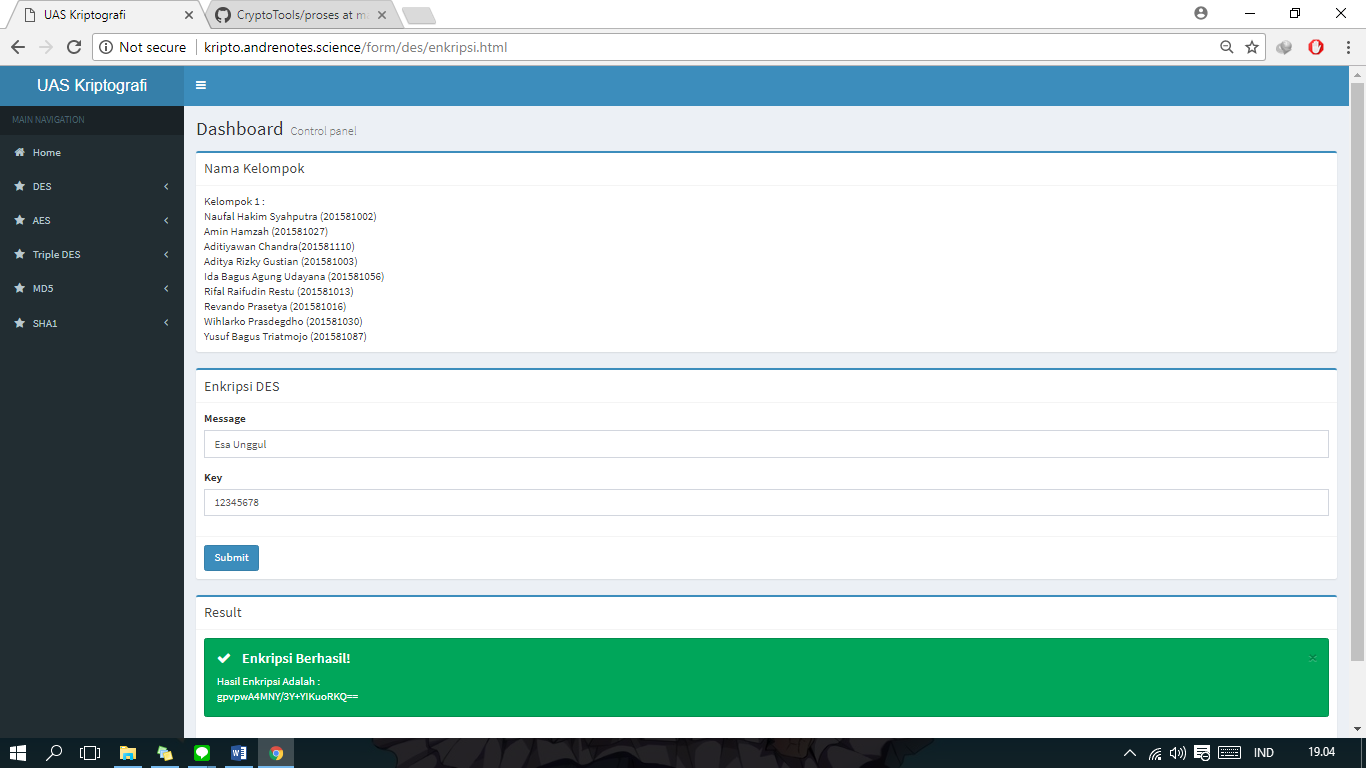
**Kelebihan Digital Signature:**

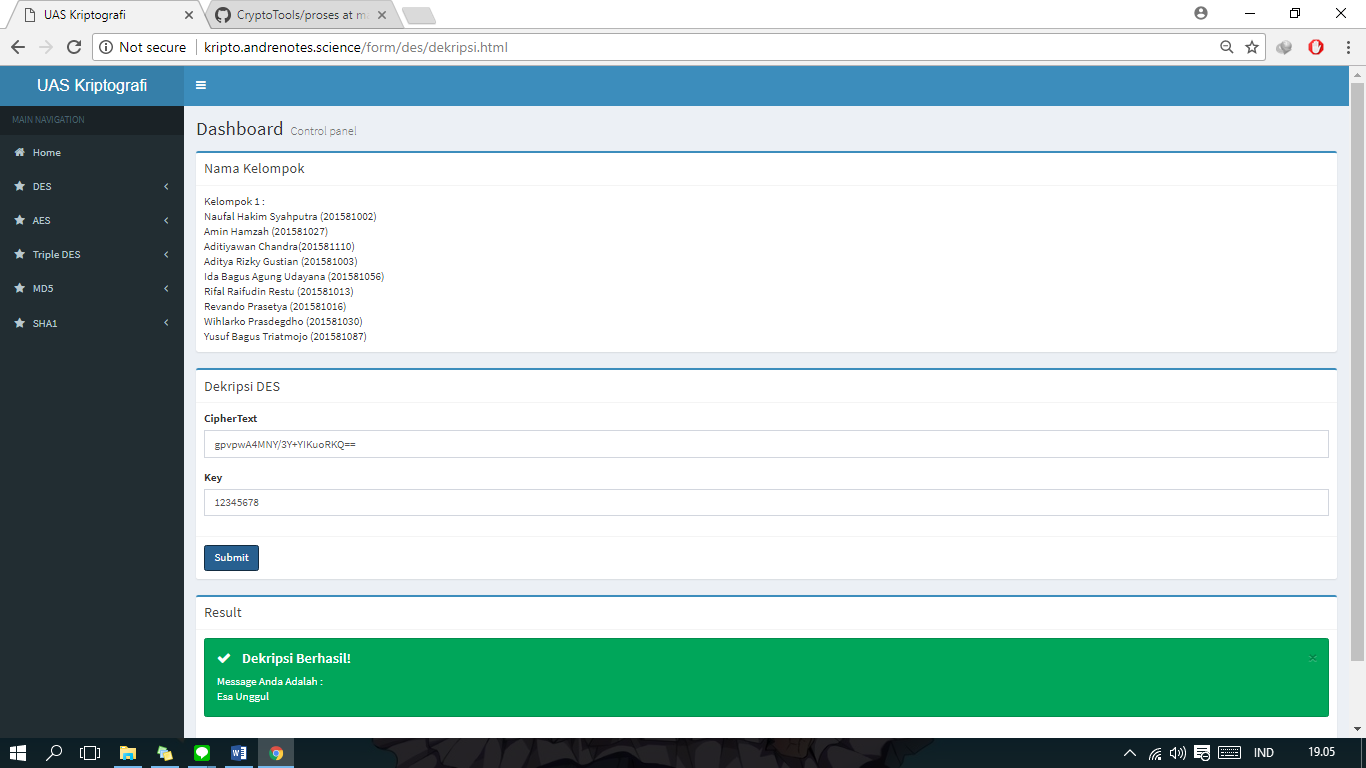
1. Algoritma ini dirancang sehingga proses enkripsi/deskripsi membutuhkan waktu yang singkat.
2. Ukuran kunci relatif lebih pendek.
3. Algoritmanya bisa menghasilkan *chiper* (sebuah algoritma untuk menampilkan enkripsi dan kebalikannya deskripsi, serangkaian langkah yang terdefinisi yang diikuti sebagai prosedur) yang lebih kuat.
4. Autentifikasi pengiriman pesan langsung diketahui oleh *chipertext* yang diterima, karena kunci hanya diketahui oleh pengirim dan penerima saja.

**Kelemahan Digital Signature:**

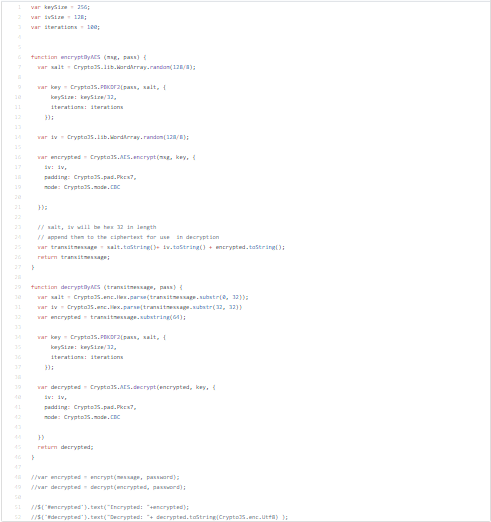
1. Kunci harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci ini.
2. Kunci harus sering diubah, mungkin pada setiap sesi komunikasi.
3. **Screenshoot Coding dan Program**
4. **DES (*Data Encryption Standard*)**

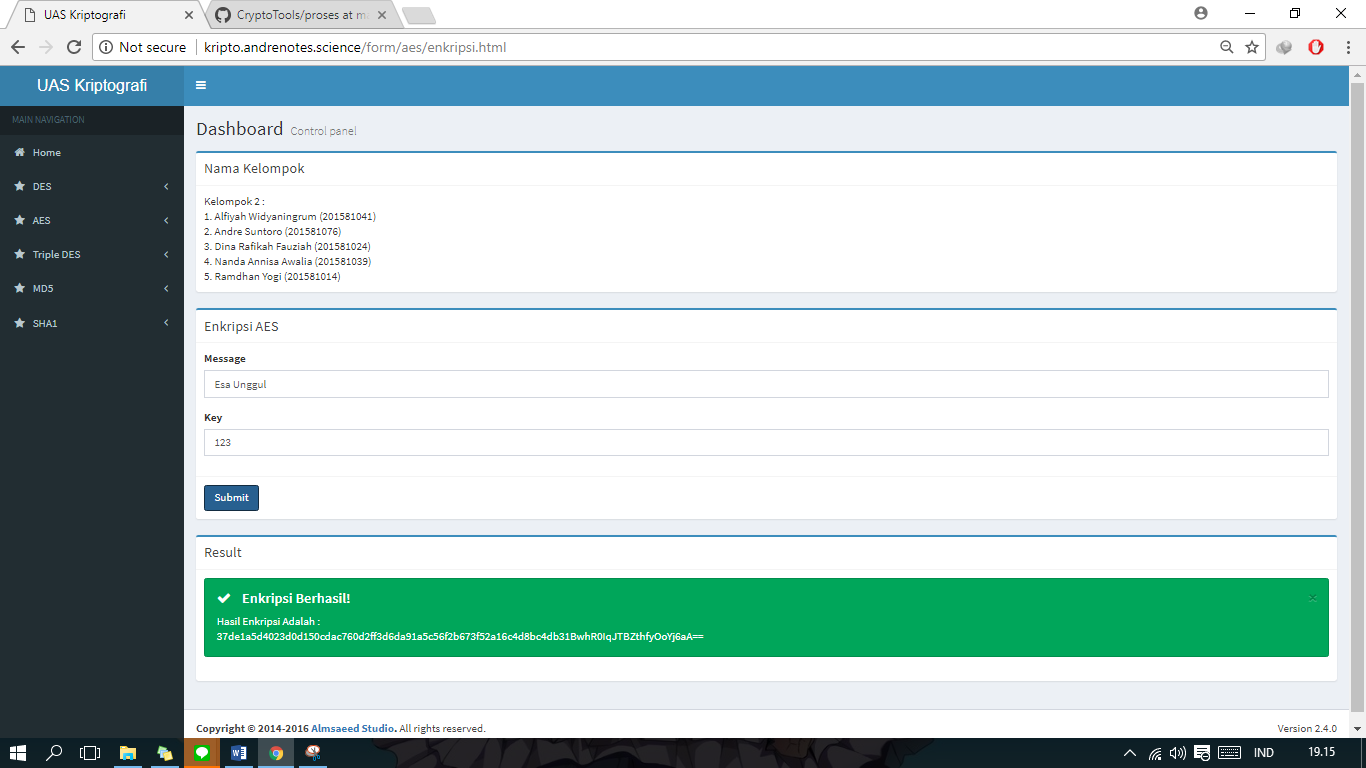
****

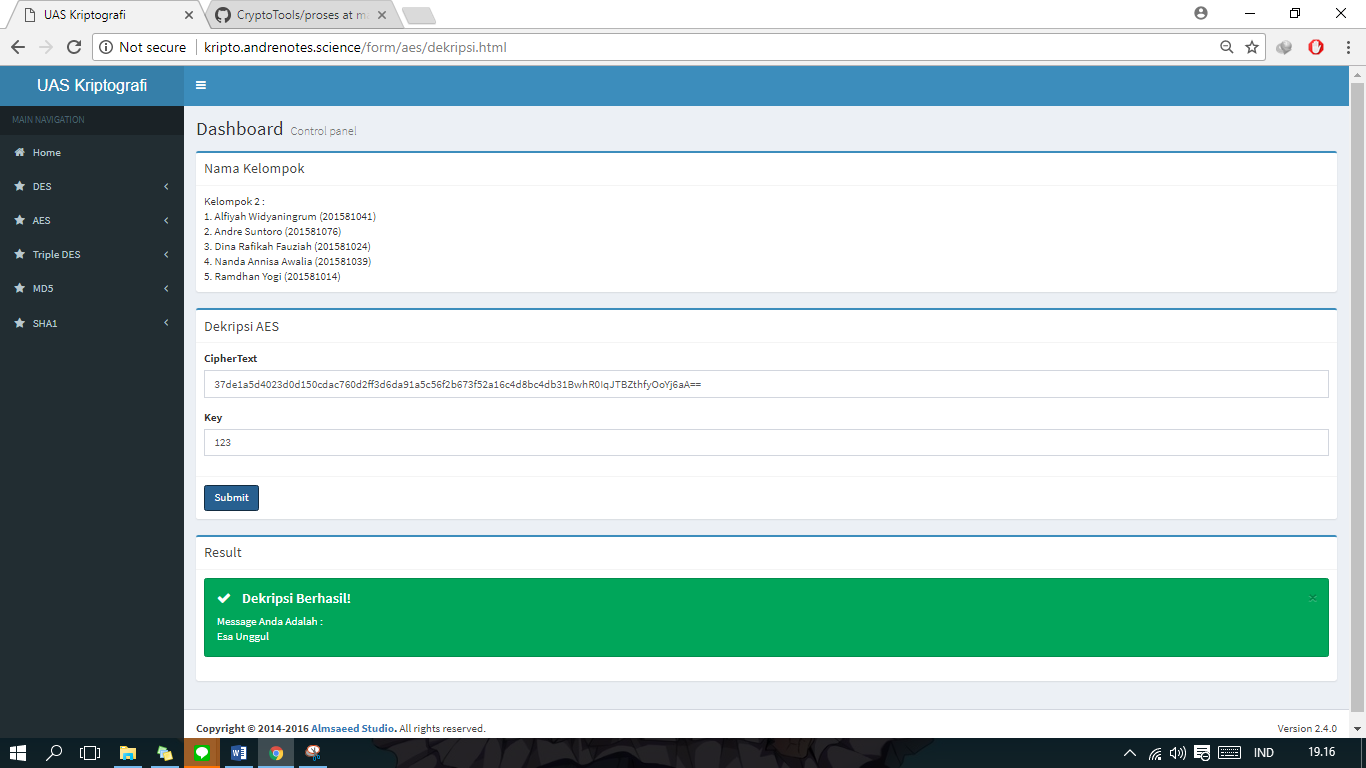




1. **AES (*Advanced Encryption Standard*)**

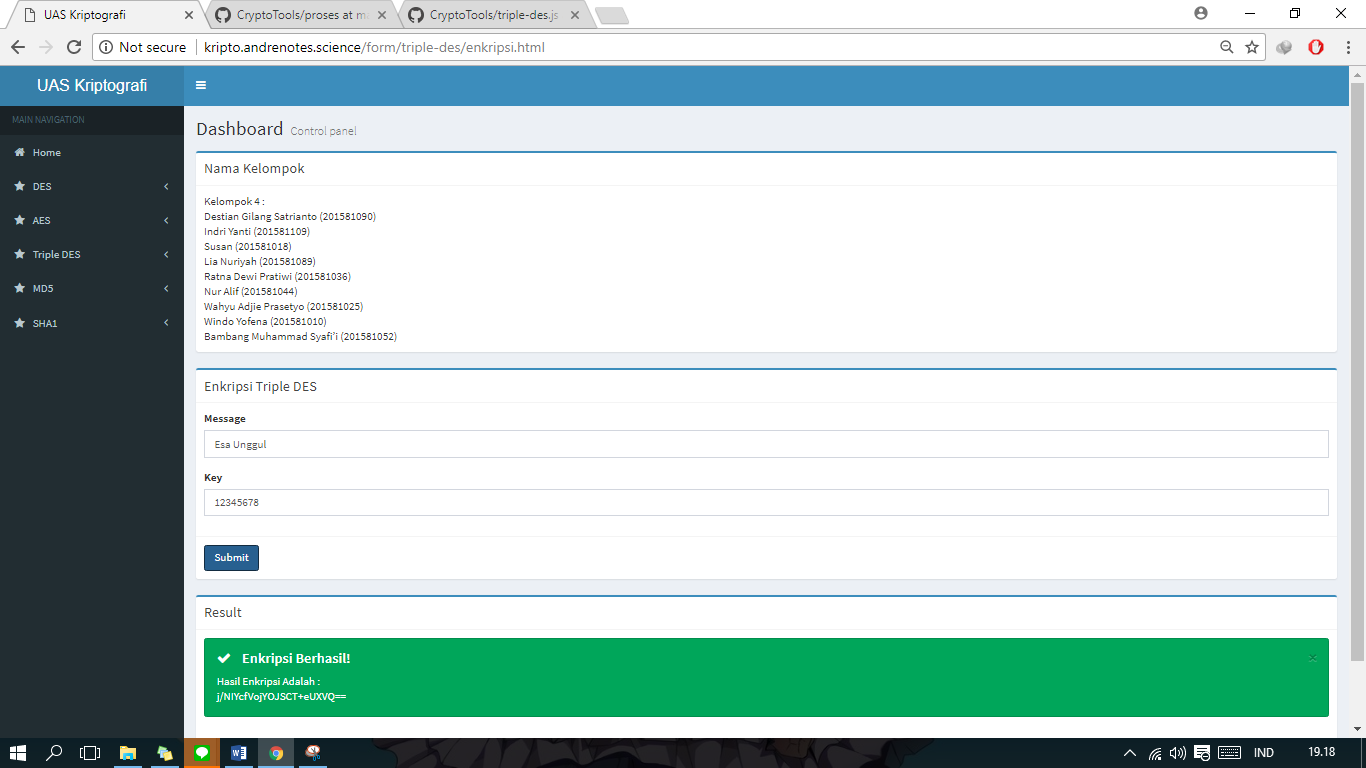
****

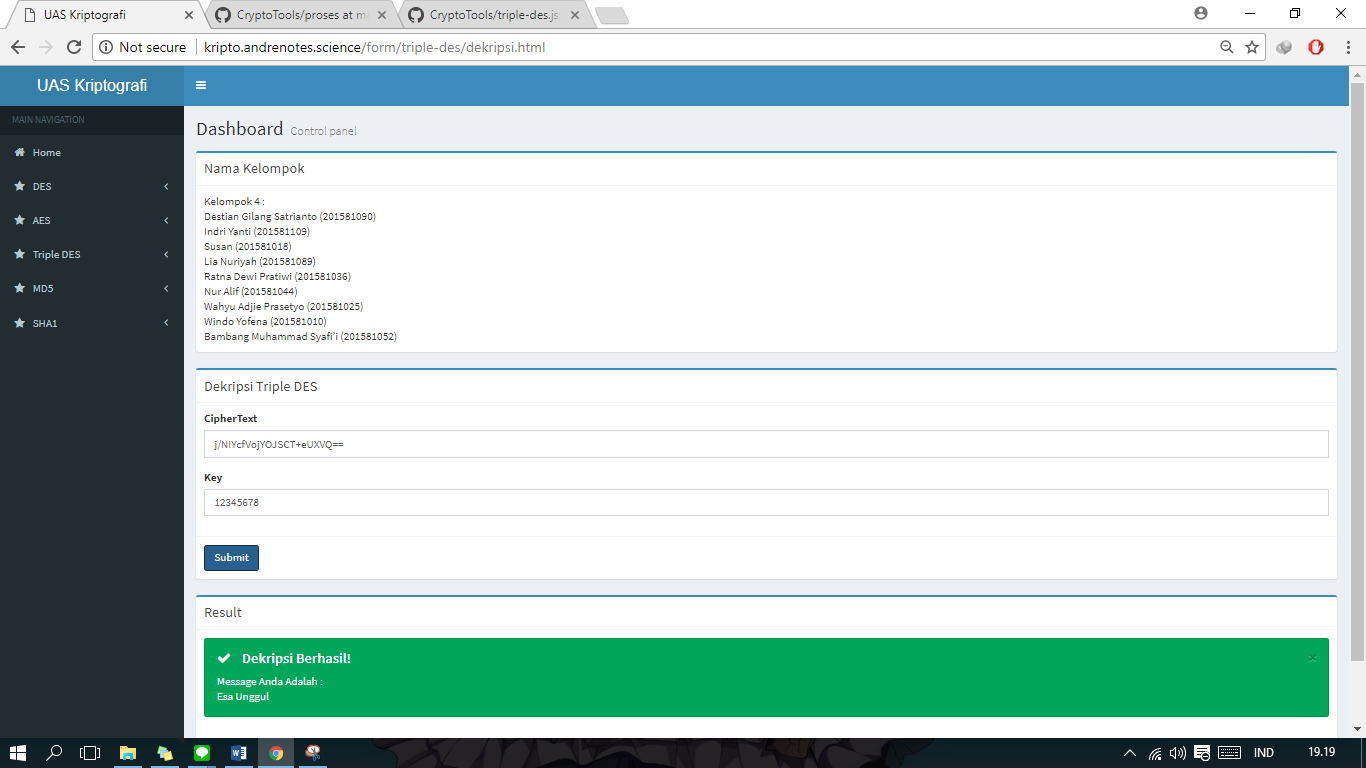




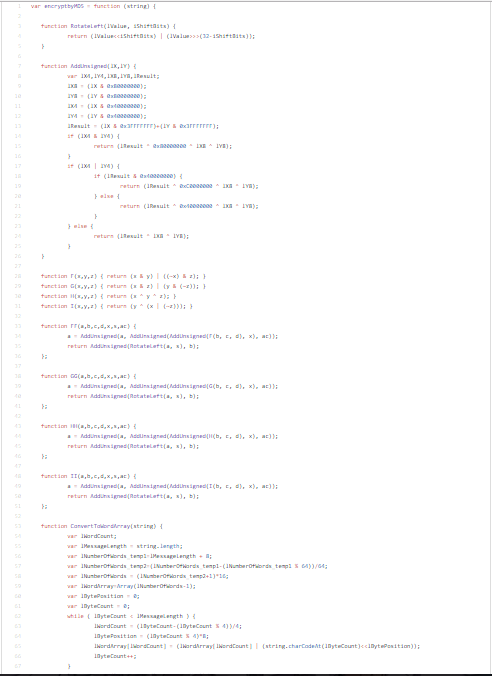
1. **Triple DES (*Triple Data Encryption Standard*)**

****

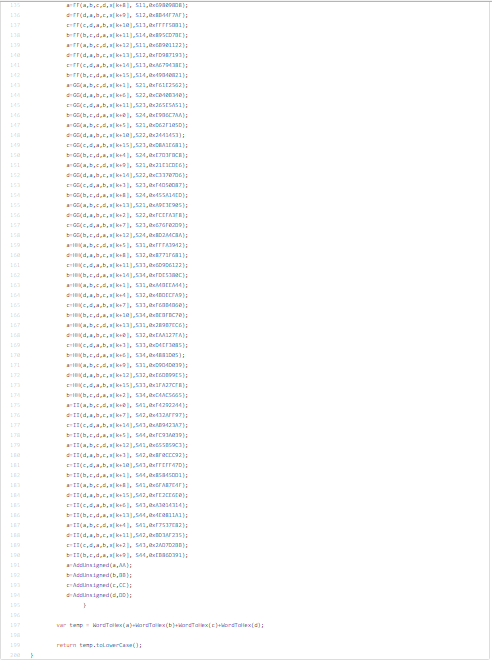


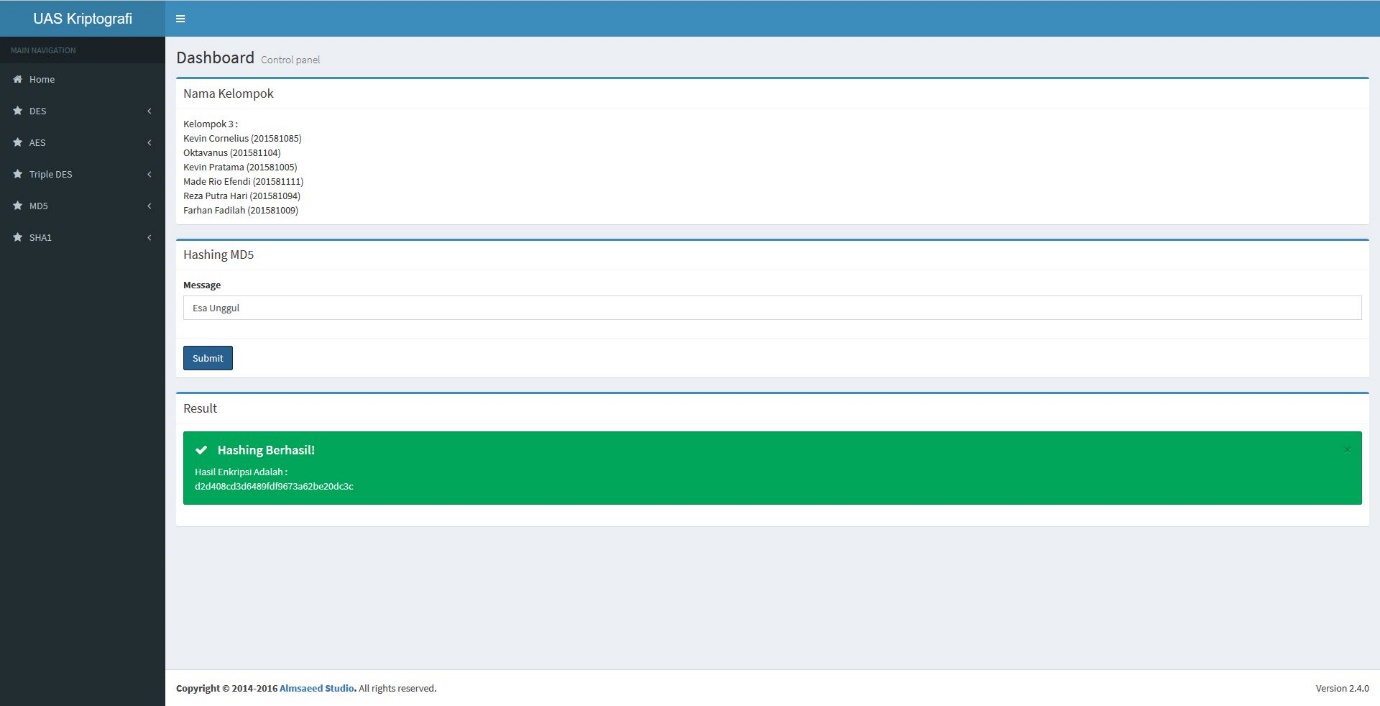


1. **MD5 (*Message Digest 5*)**

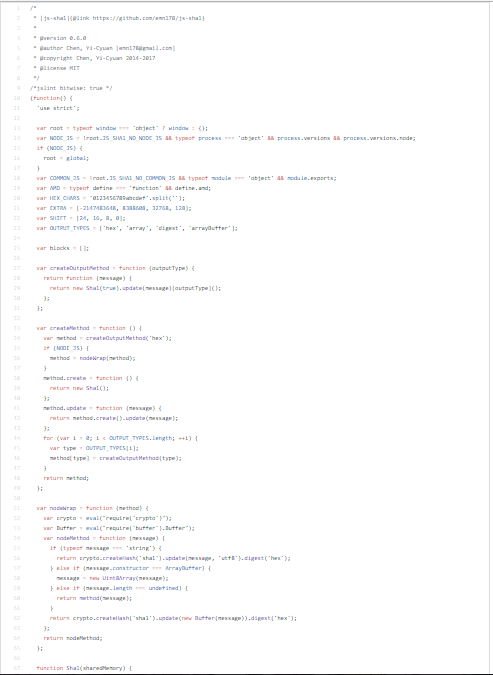
****

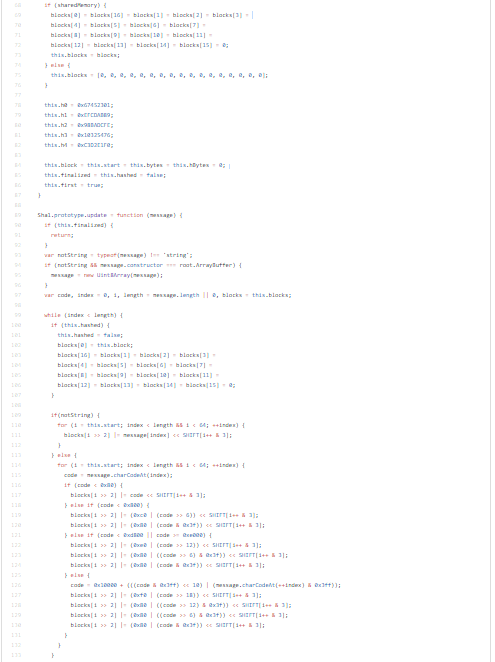
****

****

****

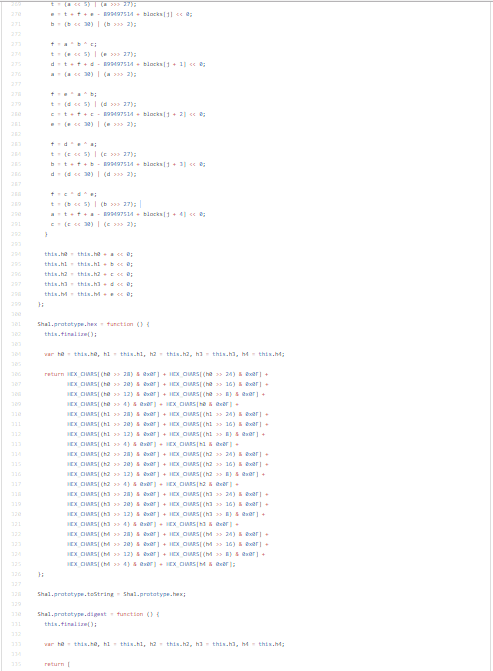
1. **SHA-1 (*Secure Hash Algorithm*)**

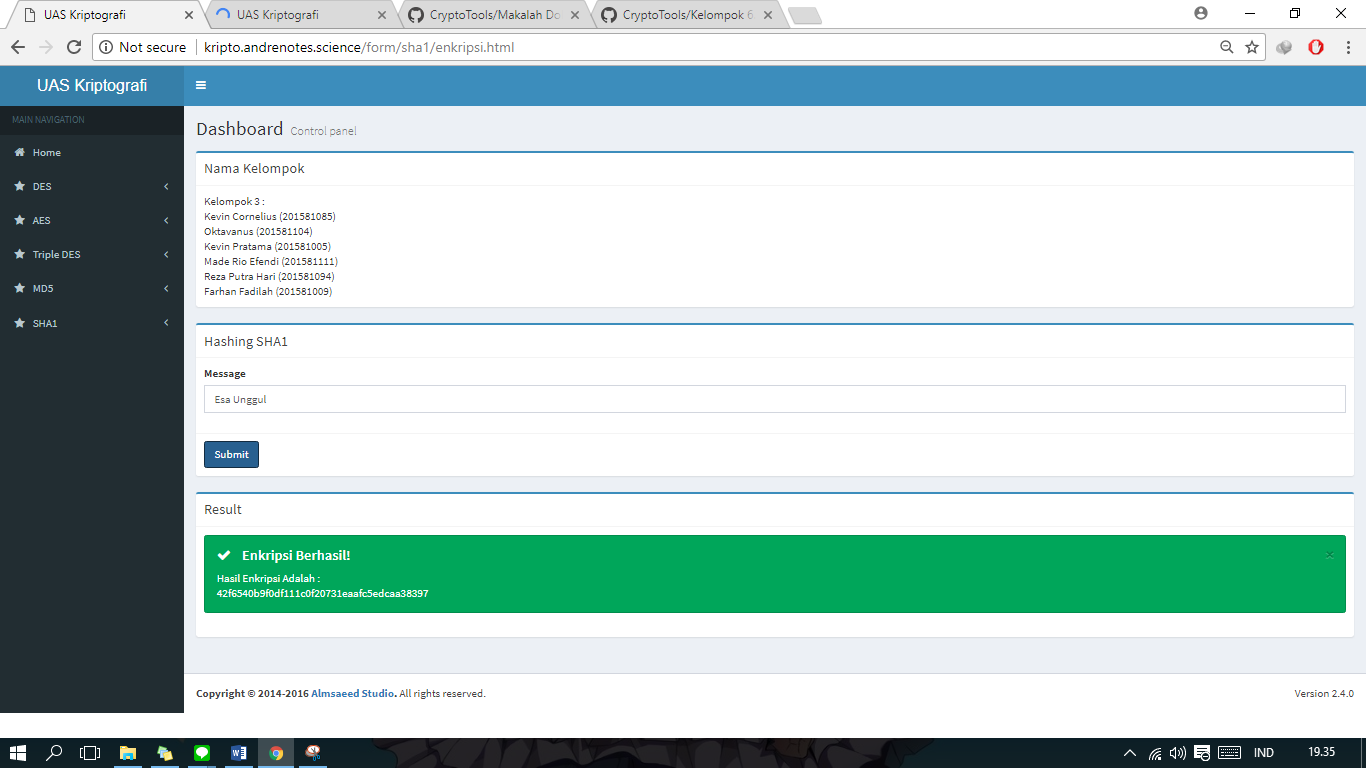
****

****

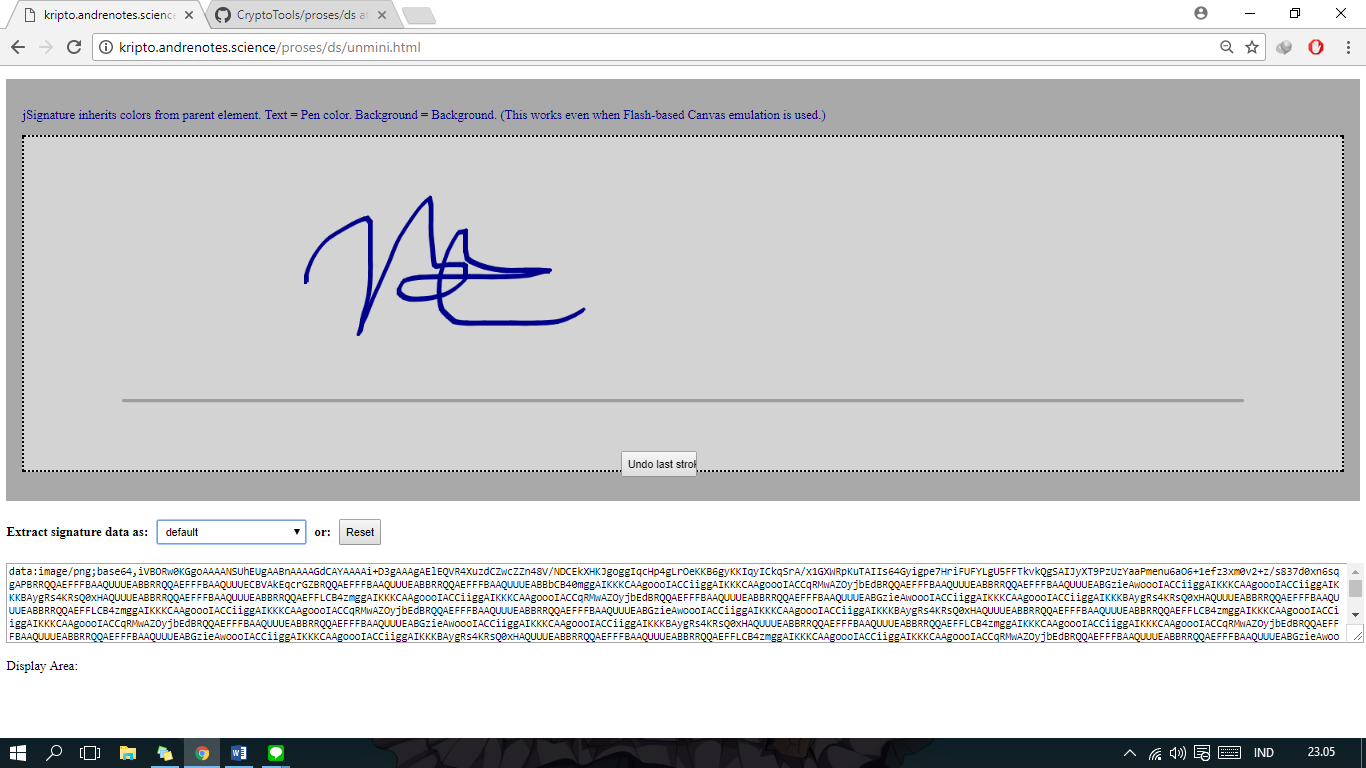
****

****

****



1. **DSA (Digital Signature Algorithm)**



**BAB III**

**KESIMPULAN**

1. **Kesimpulan**

Algoritma merupakan urutan langkah-langkah logis untuk menyelesaikan masalah yang disusun secara matematis dan benar. Sedangkan kriptografi (cryptography) berasal dari kata “crypto” yang berarti “secret” (rahasia) dan “graphy” yang berarti “writing” (tulisan). Kriptografi merupakan suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan, dari pengirim ke penerima tanpa mengalami gangguan dari pihak ketiga. Sehingga algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut.

1. **Saran**

Adapun Saran penulis sehubungan dengan bahasan makalah ini, kepada rekan-rekan mahasiswa agar lebih meningkatkan, menggali dan mengkaji lebih dalam tentang algoritma kriptografi.