**TUGAS TEORI**

**UAS KRIPTOGRAFI**

****

**NAMA : GIHFAR PRAMADAN**

**NIM : 201581098**

**PRODI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS ESA UNGGUL**

**JAKARTA**

**2018**

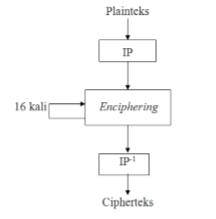
1. **DES (Data Encryption Standard)**

Algoritma DES dikembangkan di IBM dibawah kepemimpinan W.L.Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma LUCIFER yang dibuat oleh Horst Feistel. Algoritma ini telah disetujui oleh National Bureau of Standard (NBS) setelah penilaian kekuatannya oleh National Security Agency (NSA) Amerika Serikat.

DES termasuk ke dalam sistem kriptografi simetri dan tergolong jenis cipher blok. DES beroperasi pada ukuran blok 64 bit. DES mengenkripsikan 64 bit plainteks menjadi 64 bit cipherteks dengan menggunakan 56 bit kunci internal (internal key) atau lupa-kunci (subkey). Kunci internal dibangkitkan dari kunci eksternal (external key) yang panjangnya 64 bit.

Skema global dari algoritma DES adalah sebagai berikut :

1. Blok plainteks dipermutasi dengan matriks permutasi awal (initial permutation atau IP).
2. Hasil permutasi awal kemudian di-enciphering- sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
3. Hasil enciphering kemudian dipermutasi dengan matriks permutasi balikan (invers initial permutation atau IP -1 ) menjadi blok cipherteks.

****

**Fungsi DES**

Algoritma DES (Data Encryption Standard) adalah algoritma enkripsi yang paling banyak digunakan di dunia. Selama bertahun-tahun, dan di antara banyak orang, "pembuatan kode rahasia" dan DES telah serupa. Dan terlepas dari kudeta baru-baru ini oleh Electronic Frontier Foundation dalam menciptakan mesin $ 220.000 untuk memecahkan pesan terenkripsi DES, DES akan tinggal di pemerintahan dan perbankan selama bertahun-tahun yang akan datang melalui versi extended-extended. Sehingga, secara umum fungsi algoritma ini adalah diciptakan untuk merahasiakan suatu pesan agar tidak bisa dibaca oleh pihak yang tidak absah. berpedoman dari fungsi umum tersebut, sehinga pesan yang dikirimkan aman dan tidak dapat dibajak.

**Cara Kerja DES**

DES adalah blok cipher - artinya beroperasi pada blok plaintext dengan ukuran tertentu (64-bit) dan mengembalikan blok ciphertext dengan ukuran yang sama. Jadi, DES menghasilkan permutasi di antara 2 ^ 64 (baca ini sebagai: "2 sampai kekuatan ke-64") pengaturan yang mungkin dari 64 bit, yang masing-masing dapat berupa 0 atau 1. Setiap blok 64 bit dibagi menjadi dua blok masing masing 32 bit, setengah blok kiri L dan kanan setengah R. (Pembagian ini hanya digunakan dalam operasi tertentu.)

Contoh: Misalkan M adalah pesan teks biasa M = 0123456789ABCDEF, di mana M berada dalam format heksadesimal (basis 16). Menulis ulang M dalam format biner, kita mendapatkan blok teks 64-bit:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

L = 0000 0001 0010 0011 0100 0101 0110 0111

R = 1000 1001 1010 1011 1100 1101 1110 1111

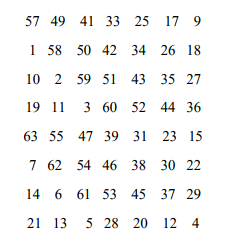
Bit pertama dari M adalah "0". Bit terakhir adalah "1". Kita membaca dari kiri ke kanan. DES beroperasi pada blok 64-bit dengan menggunakan ukuran kunci 56-bit. Kunci sebenarnya tersimpan 64 bit, tapi setiap bit ke 8 pada kunci tidak digunakan (yaitu bit bernomor 8, 16, 24, 32, 40, 48, 56, dan 64). Namun, kita tetap akan menghitung bit dari 1 sampai 64, ke kiri ke kanan, dalam perhitungan berikut. Tapi, seperti yang akan Anda lihat, delapan bit yang baru disebutkan bisa dieliminasi saat kita membuat subkunci. Contoh: Misalkan K adalah kunci heksadesimal K = 133457799BBCDFF1. Ini memberi kita kunci biner (setting 1 = 0001, 3 = 0011, dll, dan mengelompokkan bersama setiap delapan bit, yang terakhir tidak ada dalam kelompok masing-masing):

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Algoritma DES menggunakan langkah-langkah berikut:

Langkah 1: Buat 16 subkunci, masing-masing panjang 48-bit.

Kunci 64-bit dimodifikasi menurut tabel berikut, PC-1. Karena entri pertama dalam tabel adalah "57", ini berarti bahwa bit ke-57 dari kunci asli K menjadi bit pertama dari kunci permutasi K +. Bit ke 49 dari kunci asli menjadi bit kedua dari kunci yang dimodifikasi. Bit ke 4 dari kunci asli adalah bit terakhir dari kunci yang dimodifikasi. Perhatikan hanya 56 bit dari tombol asli yang muncul pada tombol permutasi.

****

Contoh: Dari tombol 64-bit asli

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001 kita mendapatkan permutasi 56-bit

K + = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Selanjutnya, pisahkan kunci ini ke bagian kiri dan kanan, C0 dan D0, di mana masing-masing bagian memiliki 28 bit.

Contoh: Dari kunci permutasi K +, kita dapatkan C0 = 1111000 0110011 0010101 0101111 D0 = 0101010 1011001 1001111 0001111

**Kelebihan & Kekurangan DES**

Data Encryption Standard ( DES ) pun seperti sistem yang lainnya, memiliki kekurangan dan kelebihan. Salah satu kekurangan DES adalah proses yang lebih lama dalam melakukan proses dekripsi dan enkripsi.

Kelebihan DES yaitu, sistem sandi lebih kompleks sehingga membutuhkan waktu yang tidak cepat untuk menembus enkripsi DES dan sulit untuk diketahui dari pihak luar. Tetapi setelah berkembangannya jaman DES tidak digunakan karena ukuran kunci yang terlalu kecil, sehingga mudah untuk ditembus

1. **Triple DES**

Fungsi Triple DES

Pada awalnya, ukuran kunci sandi DES yaitu 56 bit sudah mencukupi pada saat algoritme ini dibuat. Namun, dengan meningkatnya kemampuan komputasi, serangan brutal telah mungkin terjadi. Triple DES menyediakan metode yang sederhana dengan menambah ukuran kunci DES untuk mencegah serangan tersebut, tanpa memerlukan perancangan sandi blok (block cipher) yang sama sekali baru.

Triple-DES hanya DES dengan dua kunci 56-bit yang diaplikasikan. Dengan pesan plaintext, kunci pertama digunakan untuk mengenkripsi pesan. Kunci kedua digunakan untuk DESmendekripsi pesan terenkripsi. (Karena kunci kedua bukanlah kunci yang benar, dekripsi ini hanya mengacak data lebih jauh.) Pesan dua kali tergesa-gesa kemudian dienkripsi lagi dengan kunci pertama untuk menghasilkan cipherteks terakhir. Prosedur tiga langkah ini disebut triple-DES.

Triple-DES hanya DES dilakukan tiga kali dengan dua kunci yang digunakan dalam urutan tertentu. (Triple-DES juga bisa dilakukan dengan tiga kunci terpisah, bukan hanya dua. Dalam kedua kasus, ruang kunci yang dihasilkan adalah sekitar 2 ^ 112.).

**Cara Kerja Triple DES**

Tahap pertama, plainteks yang diinputkan dioperasikan dengan kunci eksternal pertama (K1) dan melakukan proses enkripsi dengan menggunakan algoritma DES. Sehingga menghasilkan pra-cipherteks pertama. Tahap kedua, pra-cipherteks pertama yang dihasilkan pada tahap pertama, kemudian dioperasikan dengan kunci eksternal kedua (K2) dan melakukan proses enkripsi atau proses dekripsi (tergantung cara pengenkripsian yang digunakan) dengan menggunakan algoritma DES. Sehingga menghasilkan prs-cipherteks kedua. Tahap terakhir, pra-cipherteks kedua yang dihasilkan pada tahap kedua, dioperasikan dengan kunci eksternal ketiga (K3) dan melakukan proses enkripsi dengan menggunakan algoritma DES, sehingga menghasilkan cipherteks (C). Dalam kriptografi, Triple DES adalah nama umum untuk Algoritma Data Encryption Triple (TDEA atau Triple DEA) blok cipher, yang menerapkan Standar Enkripsi Data (DES) algoritma cipher tiga kali untuk setiap blok data. Ukuran kunci DES asli cipher dari 56 bit pada umumnya cukup ketika algoritma yang dirancang, tetapi ketersediaan daya komputasi semakin membuat serangan brute force layak. Triple DES menyediakan metode yang relatif sederhana meningkatkan ukuran kunci DES untuk melindungi terhadap serangan tersebut, tanpa perlu merancang sebuah algoritma blok cipher baru.

Seperti semua blok cipher, enkripsi dan dekripsi dari beberapa blok data dapat dilakukan dengan menggunakan berbagai modus operasi, yang secara umum dapat didefinisikan secara independen dari algoritma blok cipher. Namun, ANS X9.52 menentukan secara langsung, dan NIST SP 800-67 menentukan melalui SP 800-38A bahwa beberapa mode hanya dapat digunakan dengan batasan tertentu pada mereka yang belum tentu berlaku untuk spesifikasi umum dari mode. Sebagai contoh, ANS X9.52 menetapkan bahwa untuk chaining blok cipher, vektor inisialisasi akan berbeda setiap kali, sedangkan ISO / IEC 10116 tidak. PUB FIPS 46-3 dan ISO / IEC 18033-3 mendefinisikan hanya algoritma blok tunggal, dan tidak menempatkan batasan pada mode operasi untuk beberapa blok

**Kelebihan & Kekurangan Triple DES**

Triple DES memiliki kekurangan pada performansi yang lambat dalam software. Kelebihan yang dimiliki oleh Triple DES adalah membuat menjebol enkripsinya sangat lama.

**PROGRAM TRIPLE DES**

CODINGAN TRIPLE DES DENGAN PHP

Namam file nya controller.php

<?php

/\*

inspirasi itu datang ketika kepepet .... hahahaha

\*/

require\_once 'kriptoopenssl.php';

require\_once'kripto.php';

if (!empty($\_POST["hash"])) {

$hash = $\_POST["hash"];

echo "kunci = <b>{$hash}</b>";

$kripto = new kripto($hash);

$kriptoopenssl = new kriptoopenssl($hash);

if (!empty($\_POST["data"])) {

$data = $\_POST["data"];

echo "<br/>plaintext = <b>{$data}</b>";

echo "<br/>pilihan= <b>{$\_POST["option"]}</b><br/>";

if ($\_POST["option"] == "encrypt") {

//menggunakan mcrypt

$start\_time = microtime(true);

$result = $kripto->Encrypt($data);

$stop\_time = microtime(true);

$time\_mcrypt = $stop\_time - $start\_time;

//menggunakan openssl

$start\_time = microtime(true);

$result = $kriptoopenssl->Encrypt($data);

$stop\_time = microtime(true);

$time\_openssl = $stop\_time - $start\_time;

CheckResult($result);

echo "<br/>hasil enkripsi= <b> {$result} </b>";

ShotTime($time\_mcrypt,$time\_openssl);

}

if ($\_POST["option"] == "decrypt") {

//menggunakan mcrypt

$start\_time = microtime(true);

$result = $kripto->Decrypt($data);

$stop\_time = microtime(true);

$time\_mcrypt = $stop\_time - $start\_time;

//menggunakan openssl

$result = $kriptoopenssl->Decrypt($data);

$stop\_time = microtime(true);

$time\_openssl = $stop\_time - $start\_time;

CheckResult($result);

$start\_time = microtime(true);

echo "<br/>hasil dekrip = <b> {$result} </b>";

ShotTime($time\_mcrypt,$time\_openssl);

}

} else {

throw new Exception("{{ plain text kosong }}");

}

} else {

throw new Exception("{{ kunci kosong bro }}");

}

function CheckResult($result) {

if ($result == "" || $result == false) {

throw new Exception("{{ plaintext tidak valid}}");

}

}

function ShotTime($time\_mcrypt, $time\_openssl)

{

echo "<br/>Time (mcrypt) :{$time\_mcrypt} ms";

echo "<br/>Time (openssl) :{$time\_openssl} ms";

}

1. **AES & RSA**

**Fungsi AES & RSA**

Advanced Encryption Standard (AES) merupakan algoritma cryptographic yang dapat digunkan untuk mengamakan data. Algoritma AES adalah blokchipertext simetrik yang dapat mengenkripsi (encipher) dan dekripsi (decipher) infoermasi. Enkripsi merubah data yang tidak dapat lagi dibaca disebut ciphertext; sebaliknya dekripsi adalah merubah ciphertext data menjadi bentuk semula yang kita kenal sebagai plaintext. Algoritma AES is mengunkan kunci kriptografi 128, 192, dan 256 bits untuk mengenkrip dan dekrip data pada blok 128 bits. Sedangkan Algoritma RSA masih digunakan secara luas dalam protocol electronic commerce dan dipercaya dalam pengamanan dengan kunci yang sangat panjang. Algoritma RSA disebut sebagai kunci publik karena kunci enkripsi dapat dibuat public yang berarti semua orang dapat mengetahuinya. Walaupun dibuat public key, keamanan algoritma RSA sangat terjaga. Hal itu dikarenakan kunci yang digunakan untuk enkripsi pada algoritma RSA berbeda dengan kunci yang digunakan untuk dekripsinya. Keamanan enkripsi dan dekripsi algoritma RSA terletak pada kesulitan untuk memfaktorkan modulus n yang sangat besar. Penamaan algoritma RSA diambil dari nama penemunya, yaitu Rivest, Shamir dan Adleman yang diplubikasikan pada tahun 1977 di MIT yang bertujuan untuk menjawab tantangan dari Algoritma Pertukaran Kunci Diffie Helman.

**Cara Kerja AES & RSA**

Berikut ini adalah operasi Rijndael (AES) yang menggunakan 128 bit kuncii

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) Pencampuran subkey (dengan XOR).
4. Transformasi : ByteSub dan ShiftRow
5. Pencampuran subkey Kesimpulan yang didapat adalah :

* AES terbukti kebal menghadapi serangan konvensional (linear dan diferensial attack) yang menggunakan statistik untuk memecahkan sandi.
* Kesederhanaan AES memberikan keuntungan berupa kepercayaan bahwa AES tidak ditanami trapdoor.
* Namun, kesederhanaan struktur AES juga membuka kesempatan untuk mendapatkan persamaan aljabar AES yang selanjutnya akan diteliti apakah persamaan tersebut dapat dipecahkan
* Bila persamaan AES dapat dipecahkan dengan sedikit pasangan plaintext/ciphertext, maka riwayat AES akan berakhir.
* AES didesain dengan sangat hati-hati dan baik sehingga setiap komponennya memiliki tugas yang jelas
* AES memiliki sifat cipher yang diharapkan yaitu : tahan menghadapi analisis sandi yang diketahui, fleksibel digunakan dalam berbagai perangkat keras dan lunak, baik digunakan untuk fungsi hash karena tidak memiliki weak(semi weak) key, cocok untuk perangkat yang membutuhkan key agility yang cepat, dan cocok untuk stream cipher.

Sedangkan untuk cara kerja RSA. 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.

**Kelebihan & Kekurangan AES & RSA**

1. Kelebihan AES menggunakan struktur SPN (Substitution Permutation Network) yang memiliki derajat paralelisme yang lebih besar, sehingga diharapkan lebih cepat dari pada Feistel.
2. Kekurangan AES 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.

**Kelebihan & Kekurangan Algoritma RSA**

1. Kekuatan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan menjadi faktor primanya, dalam hal ini memfaktorkan n menjadi p dan q. Karena sekali n berhasil difaktorkan, maka menghitung nilai m adalah perkara mudah. Selanjutnya, walau nilai e diumumkan, perhitungan kunci d tidaklah mudah pula karena nilai m yang tidak diketahui.
2. Kelebihan lain algoritma RSA terletak pada ketahanannya terhadap berbagai bentuk serangan, terutama serangan brute force. Hal ini dikarenakan kompleksitas dekripsinya yang dapat ditentukan secara dinamis dengan cara menentukan nilai p dan q yang besar pada saat proses pembangitkan pasangan kunci, sehingga dihasilakan sebuah key space yang cukup besar, sehingga tahan terhadap serangan.

**CONTOH PEROGRAM RSA**

Contoh Algoritma RSA |--------------------------------------------------------------------------------------------------|

Jika diketahui : p = 3 q = 7

Penyelesaian : n = p.q = 3 x 7 = 21

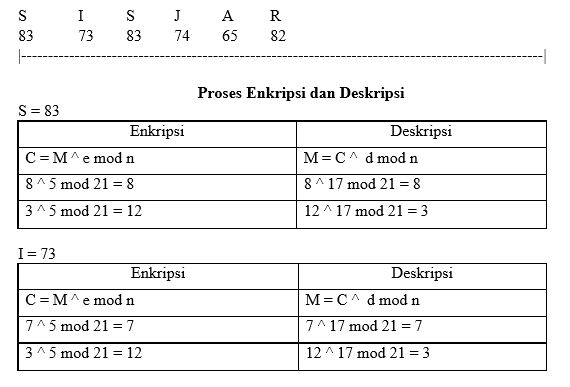
M = (p-1) (q-1) = (3 -1)(7 – 1) = 12

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)



1. **MD5, SHA1**

**Fungsi MD5, SHA1**

Algoritma yang menggunakan fungsi hash satu arah yang diciptakan oleh Ron Rivest. Algoritma merupakan pengembangan dari algoritma-algoritma sebelumnya yaitu algoritma MD2 dan algoritma MD4 karena kedua algoritma ini berhasil diserang para cryptanalyst

**Cara Kerja MD5**

Algoritma MD5 adalah algoritma yang menggunakan fungsi hash satu arah yang diciptakan oleh Ron Rivest. Algoritma merupakan pengembangan dari algoritma-algoritma sebelumnya yaitu algoritma MD2 dan algoritma MD4 karena kedua algoritma ini berhasil diserang para *cryptanalist*.

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 3.1 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 MD

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.

Cara Kerja SH1

Pesan diberi tambahan untuk membuat panjangnya menjadi kelipatan 512 bit ( l x 512 ). Jumlah bit asal adalah k bit. Tambahkan bit secukupnya sampai 64 bit kurangnya dari kelipatan 512 ( 512 – 64 = 448 ), yang disebut juga kongruen dengan 448 ( mod 512 ). Kemudian tambahkan 64 bit yang menyatakan panjang pesan. Inisiasi 5 md variable dengan panjang 32 bit yaitu a,b,c,d,e. Pesan dibagi menjadi blok-blok berukuran 512 bit dan setiap blok diolah. Kemudian keluaran setiap blok digabungkan dengan keluaran blok berikutnya, sehingga diperoleh output ( diggest ). Fungsi kompresi yang digunakan oleh algoritma sha-1 adalah sebagai berikut : A,b,c,d,e ← ( e + f (t,b,c,d) + s5 (a) + wt + kt),a,s30(b),c,d.]

**Kelebihan & Kekurangan MD5, SHA1**

1. Keamanan terhadapserangan brute-force. Hal yang paling penting adalah bahwa SHA-1 menghasilkan diggest 32-bit lebih panjang dari MD5. Dengan brute-force maka SHA-1 lebihkuat dibanding MD5.
2. Keamanan terhadap kriptanalisis. Kelemahan MD5 ada pada design sehingga lebih mudah dilakukan kriptanalisis dibandingkan SHA-1
3. Kecepatan. Kedua algoritma bekerja pada modulo 232 sehingga keduanya dibandingkan MD5 ( 80 dibanding MD5 64 ) dan harus memproses 160 bit buffer dibanding DM5 128 bit buffer, sehingga SHA-1 bekerja lebih lambat dibanding MD5 pada perangkat keras yang sama.
4. Simplicity. Kedua algoritma simple untuk dijelaskan dan mudah untuk diiemplementasikan karena tidak membutuhkan program yang besar atau table subtitusi yang besar pula.
5. **DIGITAL SIGNATURE**

**Fungsi Digital Signature**

Sistem ini berfungsi untuk memberikan jaminan bahwa dokumen digital adalah otentik. Menambahkan sistem Digital signature proses transfer dokumen akan terjamin otentikasi dan kerahasiannya. Dengan demikian dapat menghindari kemungkinan terjadinya pemalsuan dokumen terkait.

**Cara kerja dari digital signature**

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 algorithm, 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 keaman 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.

**Kelebihan & Kekurangan Digital Signature**

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

**Kelemahan :**

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.

**Kekurangan :**

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.

Dikumentasi Codingan Program DES, Triple DES, AES & RSA, MD5, SHA1, dan Digital Signature : - DES https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/des.js - Triple DES https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/triple-des.js - AES & RSA AES : https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/aes.js - MD5, SHA1 https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/sha1.js https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/md5.js https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/sha1-core.js - Digital Signature https://github.com/NaufalHSyahputra/CryptoTools/tree/master/proses/ds