**Dokumentasi Kriptografi**

****

**Disusun oleh :**

**Destian Gilang Satrianto**

**201581090**

**FAKULTAS ILMU KOMPUTER**

**JURUSAN TEKNIK INFORMATIKA**

**UNIVERSITAS ESA UNGGUL**

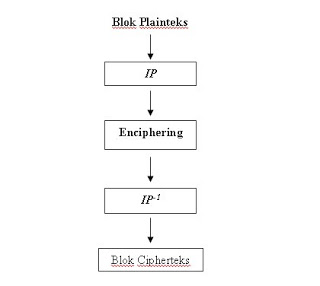
**2017**

1. Penjelasan fungsi jenis-jenis Encripsi

1. **DES (Data Encryption Standard)** adalah algoritma cipher blok yang populer karena dijadikan standard algoritma enkripsi kunci-simetri, meskipun saat ini standard tersebut telah digantikan dengan algoritma yang baru, AES, karena DES sudah dianggap tidak aman lagi. Sebenarnya DES adalah nama standard enkripsi simetri, nama algoritma enkripsinya sendiri adalah DEA (Data Encryption Algorithm), namun nama DES lebih populer daripada DEA. 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 upa-kunci (subkey). Kunci internal dibangkitkan dari kunci eksternal (external key) yang panjangnya 64 bit.

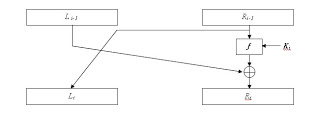
Skema global dari algoritma DES adalah sebagai berikut :

1. Blok plainteks dipermutasi dengan matriks permutasi awal (initial permutation atau IP).
2. Hasil permutasi awal kemudian di-enciphering- sebanyak 16 kaH (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.

[](https://3.bp.blogspot.com/-9ZwZdImz9Ec/VvaQnvLr4AI/AAAAAAAAADE/b8sH2OW8b2EnyQe2lFpkyvQORdjKjzE3w/s1600/des.jpg)

Di dalam proses enciphering, blok plainteks terbagi menjadi dua bagian, kiri (L) dan kanan R), yang masing-masing panjangnya 32 bit. Kedua bagian ini masuk ke dalam 16 putaran DES. Pada setiap putaran i, blok R merupakan masukan untuk fungsi transformasi yang ;isebut f. Pada fungsi f, blok R dikombinasikan dengan kunci internal K,. Keluaran dai =angsi f di-XOR-kan dengan blok L untuk mendapatkan blok R yang baru. Sedangkan blok - yang baru langsung diambil dari blok R sebelumnya. Ini adalah satu

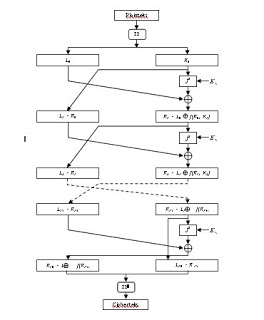
**Gambar skema global algoritma des**

[](http://1.bp.blogspot.com/_gwlUjKa1KGI/SgkiNdYcsiI/AAAAAAAAABA/ZD7HS7A4sJQ/s1600-h/des1.bmp)putaran DES. Secara watematis, satu putaran:DES dinyatakan sebagai

**Li=Ri-1 (6.1)**

**R i=L i-1 f(Ri-1, K i) (6.2)**

**Jaringan Feistel untuk satu putaran DES**

[](http://4.bp.blogspot.com/_gwlUjKa1KGI/SgkjWE7obeI/AAAAAAAAABI/6_aNR1PJQiA/s1600-h/des2.bmp)

**Algoritma Enkripsi dengan DES Permutasi Awal**

Diagram diatas memperlihatkan skema algoritma DES yang lebih rinci. Satu putaran DES merupakan model jaringan Feistel (lihat Gambar 6.2). Perlu dicatat dari Gambar 6.2 bahwa ika (L,6, R,6) merupakan keluaran dari putaran ke-16, maka (R,6, L,s) merupakan pra­:ipherteks (pre-ciphertext) dari enciphering ini. Cipherteks yang sebenarnya diperoleh dengan melakukan permutasi awal balikan, IP-1, terhadap blok pra-cipherteks.

Sebelum putaran pertama, terhadap blok plainteks dilakukan permutasi awal (initial-permutation atau IP). Tujuan permutasi awal adalah mengacak plainteks sehingga urutan bit-bit di dalamnya berubah. Pengacakan dilakukan dengan menggunakan matriks permutasi awal berikut ini:

[https://1.bp.blogspot.com/_gwlUjKa1KGI/SgktQY5g4VI/AAAAAAAAABQ/UMrXoWWIRDk/s320/des3.bmp](http://1.bp.blogspot.com/_gwlUjKa1KGI/SgktQY5g4VI/AAAAAAAAABQ/UMrXoWWIRDk/s1600-h/des3.bmp)

Cara membaca tabel/matriks: dua entry ujung kiri atas (58 dan 50) artinya:

"pindahkan bit ke-58 ke posisi bit 1"

"pindahkan bit ke-50 ke posisi bit 2", dst

**Pembangkitan Kunci Internal**

Karena ada 16 putaran, maka dibutuhkan kunci internal sebanyak 16 buah, yaitu K,, Kz, ...,K16. Kunci-kunci internal ini dapat dibangkitkan sebelum proses enkripsi atau bersamaan dengan proses enkripsi. Kunci internal dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Kunci eksternal panjangnya 64 bit atau 8 karakter.

Misalkan kunci eksternal yang tersusun dari 64 bit adalah K. Kunci eksternal ini menjadi masukan untuk permutasi dengan menggunakan matriks permutasi kompresi PC- 1 sebagai berikut:

[https://4.bp.blogspot.com/_gwlUjKa1KGI/SgktnpC2hfI/AAAAAAAAABY/dD3cAgcjc2g/s320/des4.bmp](http://4.bp.blogspot.com/_gwlUjKa1KGI/SgktnpC2hfI/AAAAAAAAABY/dD3cAgcjc2g/s1600-h/des4.bmp)

Dalam permutasi ini, tiap bit kedelapan (parity bit) dari delapan byte kunci diabaikan. Hasil 7-ermutasinya adalah sepanjang 56 bit, sehingga dapat dikatakan panjang kunci DES adalah 56 bit. Selanjutnya, 56 bit ini dibagi menjadi 2 bagian, kiri dan kanan, yang masing-masing nanjangnya 28 bit, yang masing-masing disimpan di dalam Co dan DO:

CO: berisi bit-bit dari K pada posisi

57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18

10, 2, 59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44, 36

Do: berisi bit-bit dari K pada posisi

63, 55, 47, 39, 31, 23, 15, 7, 62, 54, 46, 38, 30, 22

14, 6, 61, 53, 45, 37, 29, 21, 13, 5, 28, 20, 12, 4

Selanjutnya, kedua bagian digeser ke kiri (left shift) sepanjang satu atau dua bit bergantung pada tiap putaran.

**Kelebihan:**

• Sistem sandi lebih kompleks

• Sulit diketahui pihak luar

**Kelemahan:**

• Proses lebih lama

1. **3DES (Triple Data Encryption Standard)**

merupakan salah satu algoritma simetris pada kriptografi yang digunakan untuk mengamankan data dengan cara menyandikan data. Proses yang dilakukan dalam penyandian datanya, yaitu proses enkripsi dan proses dekripsi. Algoritma 3DES adalah suatu algoritma pengembangan dari algoritma DES (Data Encryption Standard). Perbedaan DES dengan 3DES terletak pada panjangnya kunci yang digunakan. Pada DES menggunakan satu kunci yang panjangnya 56-bit, sedangkan pada 3DES menggunakan 3 kunci yang panjangnya 168- bit (masing-masing panjangnya 56-bit). Pada 3DES, 3 kunci yang digunakan bisa bersifat saling bebas (K1 ≠ K2 ≠ K3) atau hanya dua buah kunci yang saling bebas dan satu kunci lainnya sama dengan kunci pertama (K1 ≠ K2 dan K3 = K1). Karena tingkat kerahasiaan algoritma 3DES terletak pada panjangnya kunci yang digunakan, maka penggunaan algoritma 3DES dianggap lebih aman dibandingkan dengan algoritma 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.

[](http://3.bp.blogspot.com/-BbPPuMRqcw0/T-bu1K9lNwI/AAAAAAAAAGU/j7qmHMEQNIs/s1600/3.gif)

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.

**Keuntungan**

* 3DES mudah diterapkan (dan dipercepat) baik hardware dan software.
* 3DES ada di mana-mana: kebanyakan sistem, perpustakaan, dan protokol termasuk dukungan untuk itu.

**Kekurangan**

* 3DES rentan terhadap kriptanalisis
* 3DES's hanya support 112 or 168 bits.

1. **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.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Panjang Kunci  (Nk words) | Ukuran Blok  (Nb words) | Jumlah Putaran  (Nr) |
| AES-128 | 4 | 4 | 10 |
| AES-192 | 6 | 4 | 12 |
| AES-256 | 8 | 4 | 14 |

Catatan: 1 word = 32 bit

**Gambar - Panjang Kunci Algoritma Kriptografi AES**

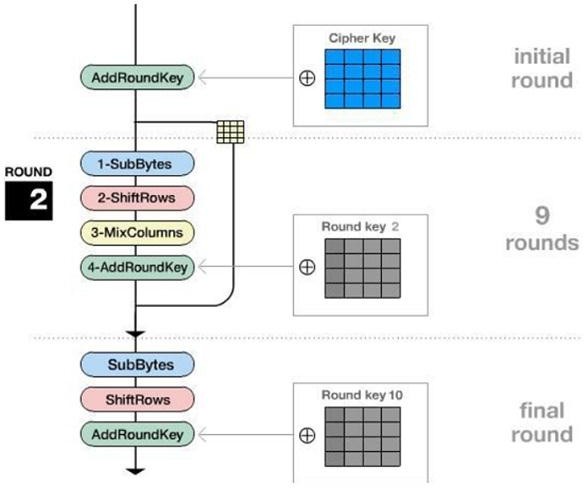
**AES** ini merupakan algoritma block cipher dengan menggunakan sistem permutasi dan substitusi (P-Box dan S-Box) bukan dengan jaringan Feistel sebagaimana block cipher pada umumnya. Jenis AES terbagi menjadi 3, yaitu:

**1.** AES-128

**2.** AES-192

**3.** AES-256

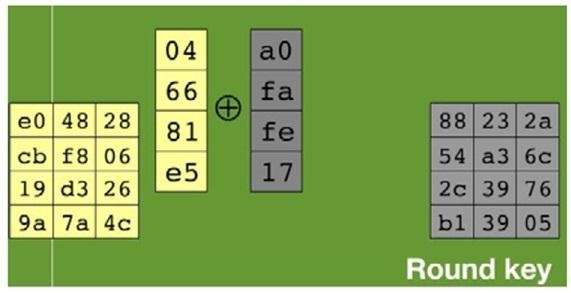
Pengelompokkan jenis AES ini adalah berdasarkan panjang kunci yang digunakan. Angka-angka di belakang kata AES menggambarkan panjang kunci yang digunakan pada tiap-tiap AES. Selain itu, hal yang membedakan dari masing-masing AES ini adalah banyaknya round yang dipakai. **AES-128 menggunakan 10 round,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

**2 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 diatas

## **Add Round Key**

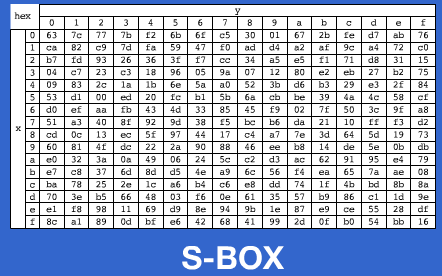
Add Round Key merupakan transformasi yang melakukan operasi XOR terhadap sebuah

round key dengan array state dan hasilnya disimpan di array state.

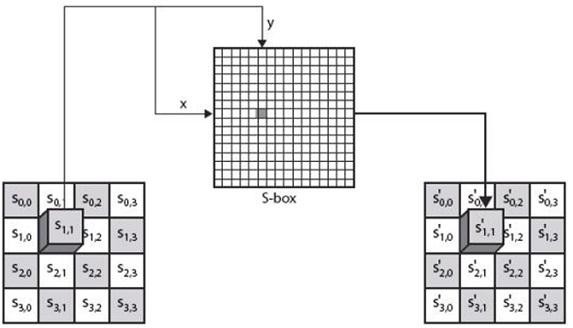


## **Sub Bytes**

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**

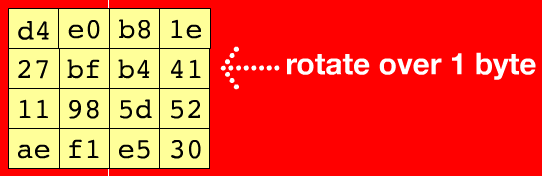


**Keterangan:** Gambar 4 adalah contoh dari Rijndael S-*Box*, di sana terdapat nomor kolom dan nomor baris. Seperti yang telah disebutkan sebelumnya, tiap isi kotak dari blok chiper berisi informasi dalam bentuk *heksadesimal* yang terdiri dari dua digit, bisa angka-angka, angka-huruf, ataupun huruf-angka 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.

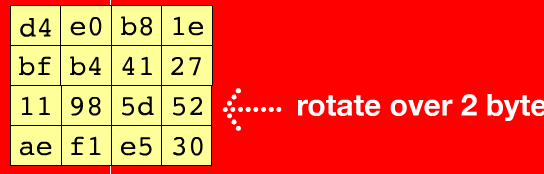
**Shift Rows**

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.

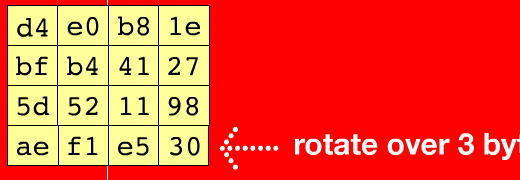
Geser baris ke-1:



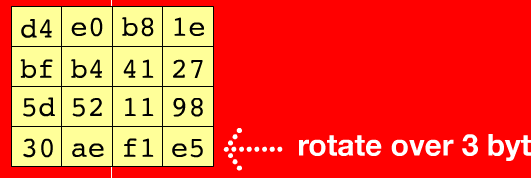
Hasil pergeseran baris ke-1 dan geser baris ke-2:



Hasil pergeseran baris ke-2 dan geser baris ke-3:



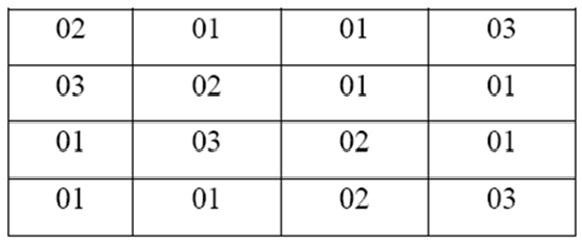
Hasil pergeseran baris ke-3:

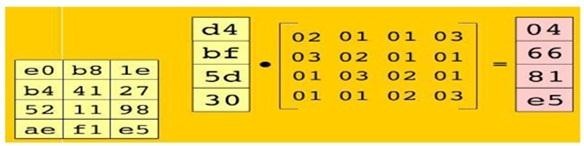


Seperti yang terlihat pada Gambar 5, tahap Shift Row sama sekali tidaklah rumit, karena ini adalah proses standar yang hanya berupa pergeseran. Langkah terakhir adalah Mix Column.

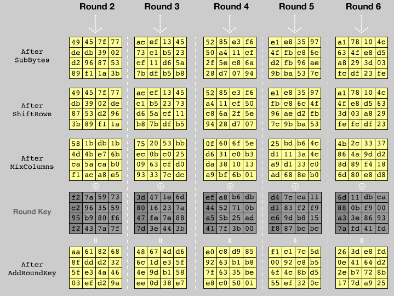
**Mix Column**

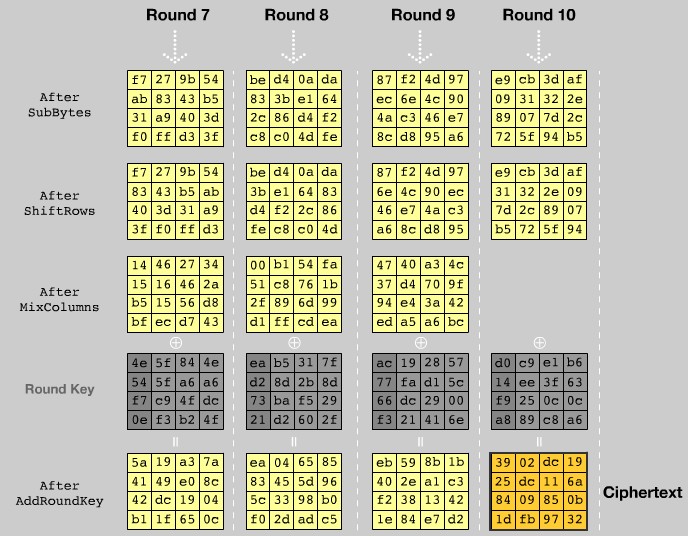
Yang terjadi saat *Mix Column* adalah mengalikan tiap elemen dari blok *chiper* dengan matriks yang ditunjukkan oleh Gambar 6. Tabel sudah ditentukan dan siap pakai. Pengalian dilakukan seperti perkalian matriks biasa yaitu menggunakan *dot product* lalu perkalian keduanya dimasukkan ke dalam sebuah 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**





**Gambar 11. 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. **RSA** adalah salah satu teknik kriptografi dimana kunci untuk melakukan enkripsi berbeda dengan kunci untuk melakukan dekripsi. Kunci untuk melakukan enkripsi disebut sebagai kunci publik, sedangkan kunci untuk melakukan dekripsi disebut sebagai kunci privat. Orang yang mempunyai kunci publik dapat melakukan enkripsi tetapi yang dalam melakukan dekripsi hanyalah orang yang memiliki kunci privat. Kunci publik dapat dimiliki oleh sembarang orang, tetapi kunci privat hanya dimiliki oleh orang tertentu saja.

Untuk pembangkitan pasangan kunci RSA, digunakan algoritma sebagai berikut:

1. Dipilih dua buah bilangan prima sembarang yang besar, p dan q. Nilai p dan q harus dirahasiakan.
2. Dihitung n = p x q. Besaran n tidak perlu dirahasiakan.
3. Dihitung m = (p – 1)(q – 1). Besaran m perlu dirahasiakan.
4. Dipilih sebuah bilangan bulat sebagai kunci publik, disebut namanya e, yang relatif prima terhadap m. e relatif prima terhadap m artinya faktor pembagi terbesar keduanya adalah 1, secara matematis disebut gcd (e,m) = 1. Untuk mencarinya dapat digunakan algoritma Euclid. Nilai e bersifat tidak rahasia.
5. Dihitung kunci privat, disebut namanya d sedemikian agar (d x e) mod m = 1. Untuk mencari nilai d yang sesuai dapat juga digunakan algoritma Extended Euclid. Nilai D bersifat rahasia.

Maka hasil dari algoritma tersebut diperoleh :

1. kunci publik adalah pasangan (e,n). Bersifat tidak rahasia.
2. kunci private adalah pasangan (d,n). Bersifat rahasia

**Algoritma Enkripsi/Dekripsi**

Enkripsi

* Ambil kunci publik penerima pesan, e, dan n
* Nyatakan plainteks M menjadi blok-blok M1, M2, …, sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, n – 1].
* Setiap blok Mi dienkripsi menjadi blok Ci dengan rumus Ci = Mi ^ e mod n

Dekripsi

* Setiap blok cipherteks Ci didekripsi kembali menjadi blok Mi dengan rumus Mi = Ci ^ d mod n

**Algoritma Euclid**

Algoritma ini mencari FPB dengan cara melakukan pembagian berulang-ulang dimulai dari kedua bilangan yang hendak kita cari FPBnya sampai kita mendapatkan sisa 0 dari hasil pembagian.

Marilah kita lihat contoh yang lain, cari FPB dari 40 dan 64 :

* 64 ÷ 40 = 1 dengan sisa 24
* 40 ÷ 24 = 1 dengan sisa 16
* 24 ÷ 16 = 1 dengan sisa 8
* 16 ÷ 8 = 2 dengan sisa 0.

Kita berhenti di sini sebab kita sudah mendapat sisa 0. Bilangan terakhir yang kita gunakan untuk membagi adalah 8, jadi FPB dari 40 dan 64 adalah 8

Contoh lain, cari FPB 3337 dan 79 :

* 3337 ÷ 79 = 42 dengan sisa 19
* 79 ÷ 19 = 4 dengan sisa 3
* 19 ÷ 3 = 6 dengan sisa 1

Kita berhenti karena sudah mendapatkan sisa 1.

**Contoh RSA Sederhana**

1. p = 47 dan q = 71 (keduanya prima).
2. n = p ⋅ q = 3337
3. m = (p – 1)(q – 1) = 3220
4. Pilih e yg relativ prime terhadap m, gcd(e,m) = 1. e = 79 => gcd(79, 3337) = 1
5. Cari nilai d, d\*e = 1 mod (m)

          d\*79 = 1 mod 3220

          d\*79 mod 3220 = 1

          d = 1019

Sehingga didapatkan :

1. Public key : (79, 3337)
2. Private key : (1019, 3337)

**Proses Enkripsi**

Setelah didapat perhitungan di atas, maka akan dilakukan enkripsi plaintext M = AKU. Pertama-tama plaintext tersebut diubah menjadi format ASCII sebagai berikut

Karakter          A                K           U

ASCII             65               75          85

M1 = 657 M2 = 585

Setelah dibagi perblock, maka akan dihitung menggunakan rumus Ci = Mi ^ e mod n.

* C1 = 657 ^ 79 mod 3337 = 2349
* C2 = 585 ^ 79 mod 3337 = 685

Maka, chipertext yang didapatkan adalah C = 320328

**Proses Dekripsi**

Setelah chipertext dari kata AKU didapat, untuk mengubahnya kembali jadi plaintext menggunakan dekripsi dengan rumus Mi = Ci ^ d mod n.

* M1 = 2349 ^ 1019 mod 3337 = 657
* M2 = 685 ^ 1019 mod 3337 = 585

Maka, setelah di dekripsi hasilnya akan sama. Yaitu 657585.

**Kelebihan dan Kelemahan RSA**

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. 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.

Namun demikian, kelebihan tersebut sekaligus menjadi kelemahan dari sistem ini. Ukuran kunci privat yang terlalu besar akan mengakibatkan proses dekripsi yang cukup lambat, terutama untuk ukuran pesan yang besar. Oleh karena itu, RSA umumnya digunakan untuk meng-enkripsi pesan berukuran kecil seperti kata kunci dari enkripsi simetris seperti DES dan AES yang kemudian kunci tersebut dikirim secara bersamaan dengan pesan utama.

1. **SHA** adalah secure Hash Algorithm

SHA adalah serangkaian fungsi cryptographic hash yang dirancangoleh National Security Agency (NSA) dan diterbitkan oleh NIST sebagai US Federal Information Processing Standard.

SHA adalah Secure Hash Algoritma. Jenis-jenis SHA yaitu SHA-0, SHA-1, dan SHA-2

Untuk SHA-2 menggunakan algoritma yang identik dengan ringkasan ukuran variabel yang terkenal sebagai SHA-224, SHA-256, SHA-384, dan SHA-512.

**CARA KERJA SHA - 1**

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.]

1. **Pengertian MD5** (Message-Digest algortihm 5) adalah fungsi hash kriptografik yang digunakan secara luas dengan hash value 128-bit (sumber: wikipedia). MD5 itu adalah salah satu dari one-way hashing algorithms yg bisa menerima input dg arbitrary length lalu menghasilkan digest/output 128-bit. Penjelasan MD5 secara sederhana : dari input yg panjangnya terserah bisa dicerna menjadi suatu “kode” yang panjangnya selalu sama (128 bit; kalo ditulis dlm hex jadi 32 characters). Yang dimaksud hash atau digest adalah sifatnya yg men-“cerna” menjadi “kode” pendek itu. Inputnya cuma beda satu bit aja maka outputnya berbeda dari yang aslinya ; sifat ini yang cocok untuk penerapan MD5 sebagai checksum (misalnya utk ngecek/verify untuk download corrupt apa nggak, bukan cuma ISO image aja, bisa file/data apa aja). Yang dimaksud one-way adalah proses MD5 itu adalah dari input ke output saja, jadi kalo cuma diketahui outputnya saja tidak bisa diketahui inputnya. Sifat ini menyebabkan MD5 cocok juga utk menyimpan hash dr password utk authentication (yg disimpan cukup hashnya, tidak perlu passwordnya, ini juga salah satu alasan kenapa kalo passwordnya lupa adminnya juga gak bisa recover jadi harus direset/ganti password baru). Jadi dengan fasilitas Anda hanya bisa mengacaknya saja, namun tidak bisa menterjemahkan apa arti dari Hasil MD5 tersebut. Makanya jangan heran, sebuah situs yang menggunakan password MD5 sebagai penyimpan passwordnya tidak mampu menrecover password yang hilang, hanya bisa mengganti / mereset password yang hilang tersebut dengan password baru. Password MD5 merupakan salah satu perlindungan kepada user dalam menggunakan fasilitas internet di dunia maya, terutama yang berhubungan dengan password, karena sebuah password adalah kunci yang sangat berharga bagi kita yang sering melakukan aktifitas di dunia maya, bisa kita bayangkan apabila seorang cracker mampu menjebol database website misalnya situs pemerintah yang sifatnya sangat rahasia kemudian cracker tersebut mencari bug dari situs targetnya dengan berbagai macam metode/teknik Hacking (seperti : SQL Injection, Keylogger, Social Engineering, Trojan Horse, DDOS d.l.l) supaya cracker bisa menembus ke database dan mendapatkan password korbannya dalam bentuk hash, dan kalau berhasil mencuri passwordnya dalam bentuk hash yang totalnya berjumlah 32bit (contoh hash : fdf0ef0ea5c1620f77107f3f1047fb4c) maka dengan mudah password hash hasil encrypt MD5 tersebut tinggal di decrypt ke dalam bentuk plain text (teks biasa) dengan menggunakan tools/software yang bisa didownload gratis dari google maupun website penyedia layanan decrypt password hash MD5 ke password yang sebenarnya, dengan demikian secara otomatis sang cracker pun dengan leluasa melakukan semua hal yang dia inginkan seperti mencuri data, merubah data, mengganti tampilan suatu website (Deface) dan bahkan ada yang hanya mendiamkannya saja karena maksud utamanya adalah untuk mengetes security dari situs targetnya saja dan untuk memenuhi rasa penasarannya sampai dia bisa menembus databasenya dan tidak berniat untuk merusak, setelah bisa ditembus databasenya ditinggalkan begitu saja , biasanya hal tersebut dilakukan oleh seorang hacker golongan putih (WhiteHat).

**Cara Kerja MD5**

Langkah-langkah pembuatan *message digest* secara garis besar:

Penambahan bit-bit pengganjal (*padding bits*).

Penambahan nilai panjang pesan semula.

Inisialisasi penyangga (*buffer*) *MD*.

Pengolahan pesan dalam blok berukuran 512 bit.

**Penambahan Bit-bit Pengganjal**

* Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512.
* Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512.
* Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

**Penambahan Nilai Panjang Pesan**

* Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula.
* Jika panjang pesan > 264 maka yang diambil adalah panjangnya dalam modulo 264. Dengan kata lain, jika panjang pesan semula adalah *K* bit, maka 64 bit yang ditambahkan menyatakan *K* modulo 264.
* Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

**3. Inisialisai Penyangga MD**

* *MD5* membutuhkan 4 buah penyangga (*buffer*) yang masing-masing      panjangnya 32 bit. Total panjangpenyangga adalah 4 ´ 32 = 128 bit.
* Keempat penyangga ini menampung hasil antara dan hasil akhir.Keempat penyangga ini diberi nama *A*, *B*, *C*, dan D. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut :

*A* = 01234567

*B* = 89ABCDEF

*C* = FEDCBA98

*D* = 76543210

**4. Pengolahan Pesan dalam Blok Berukuran 512 bit**

* Pesan dibagi menjadi *L* buah blok yang masing-masing panjangnya 512 bit (*Y*0 sampai *YL* – 1).
* Setiap blok 512-bit diproses bersama dengan penyangga *MD* menjadi keluaran 128-bit, dan ini disebut proses *H*MD5.

1. **Tanda tangan digital** adalah tanda tangan elektronik yang dapat digunakan untuk mengotentikasi identitas pengirim pesan atau penandatangan dokumen, dan mungkin untuk memastikan bahwa konten asli dari pesan atau dokumen yang telah dikirim tidak berubah. Tanda tangan digital mudah diangkut dan tidak dapat ditiru oleh orang lain, juga bisa dicap secara otomatis di setiap waktu. Kemampuan untuk memastikan bahwa pesan asli yang ditandatangani tiba berarti pengirim tidak dapat menyangkalnya nanti.

**Keunggulan Digital Signature**

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

**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.

**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.

**Otentikasi 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.

**Source Code Aplikasi**

[**https://github.com/NaufalHSyahputra/CryptoTools/tree/master/proses**](https://github.com/NaufalHSyahputra/CryptoTools/tree/master/proses)