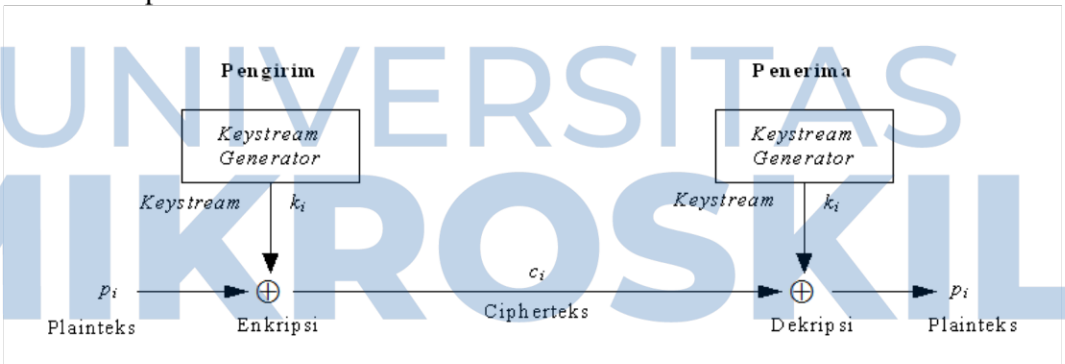
BAB II
TINJAUAN PUSTAKA

2.1. Kriptografi

Kata kriptografi berasal dari bahasa Yunani, "*kryptós*" yang berarti tersembunyi dan "*gráphein*" yang berarti tulisan. Sehingga kata kriptografi dapat diartikan sebagai "tulisan tersembunyi". Kriptografi merupakan seni dan ilmu untuk menjaga keamanan pesan (Munir, 2006). Dalam menjaga keamanan pesan, kriptografi mentransformasikan pesan asli (*plaintext*) ke dalam bentuk pesan sandi (*ciphertext*). Proses penyandian pesan ini disebut proses enkripsi. Enkripsi menjadikan pesan yang telah disandikan tersebut tidak dapat dimengerti dan dipahami isinya oleh pihak lain. *Ciphertext* ini yang kemudian dikirimkan oleh pengirim kepada penerima. Setelah sampai ke penerima, *ciphertext* tersebut ditranformasikan kembali ke dalam bentuk *plaintext* agar dapat membaca pesan yang dikirim. Proses pengembalian *ciphertext* menjadi *plaintext* disebut dengan proses dekripsi.



Gambar 2. 1. Proses enkripsi dan dekripsi. Sumber : (Ariyus, D., 2008)

Tujuan dari proses enkripsi selain untuk meningkatkan keamanan data tetapi juga berfungsi untuk melindungi data agar tidak dapat dibaca oleh pihak lain dan mencegah agar orang-orang yang tidak berhak, menyisipkan atau menghapus data.

Algoritma kriptografi yang baik tidak ditentukan oleh kerumitan dalam mengelola data atau pesan yang akan disampaikan. Yang terpenting adalah algoritma tersebut memiliki 4 persyaratan berikut:

1. Kerahasiaan (*confidentiality*). *Plaintext* hanya dapat dibaca oleh dua pihak berwenang.
2. Otentikasi (*authentication*). Pesan yang dikirimkan sebaiknya dapat dipastikan sumbernya, keasliannya, muatannya, waktu pembuatannya, dan lain-lain. Pengirim pesan harus dapat diindentifikasi dengan pasti, penyusup harus dipastikan tidak bisa berpura – pura menjadi orang lain.
3. Integritas Pesan (*message integrity*). Penerima pesan harus dapat memastikan bahwa pesan yang dia terima tidak dimodifikasi dari pesan dalam proses transmisi data sampai saat pesan dibuka.
4. Tidak Dapat Disangkal (*non-repudiation*). Pengirim pesan harus tidak bisa menyangkal pesan yang dikirim. (Bruce Scheiner,1996).

Berdasarkan kunci yang dipakai, algoritma kriptografi dapat dibedakan atas dua jenis yaitu algoritma simetrik (*symmetric*) dan asimetrik (*asymmetric*).

Algoritma Simetrik

Algoritma simetrik dapat pula disebut sebagai algoritma konvensional, dimana kunci dekripsi dapat ditentukan dari kunci enkripsinya, begitu pula sebaliknya. Pada algoritma simetrik, kunci enkripsi dan kunci dekripsinya sama.

Keamanan dari algoritma ini terletak pada kuncinya, jika kunci diberitahukan atau dibocorkan maka siapa saja dapat mengenkrip dan mendekrip data, jadi kunci harus benar-benar rahasia dan aman. Proses enkripsi dan dekripsi dari fungsi algoritma ini dapat dinotasikan sebagai berikut

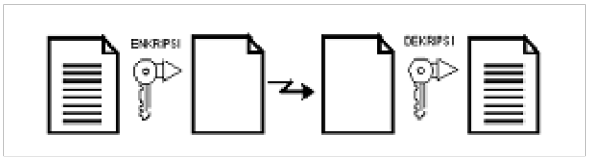
E_k (P) = C (Proses Enkripsi)

D_k (C) = P (Proses Dekripsi)

Dimana E adalah fungsi enkripsi, D adalah fungsi dekripsi, k adalah kunci enkripsi dan dekripsi, P adalah *plaintext* (pesan yang sebenarnya) dan C adalah *ciphertext* (hasil enkripsi dari *plaintext*).

Proses enkripsi dan dekripsi dengan algoritma

simetrik dapat digambarkan sebagai berikut :



Gambar 2. 2. Gambar Enkripsi dan dekripsi Algoritma Simetris

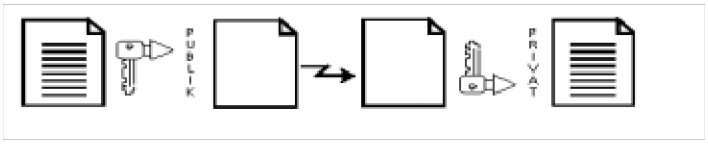
Algoritma Asimetrik

Algoritma Asimetrik (*Asymmetric* atau *Public Key*) adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsi. Algoritma ini disebut juga algoritma kunci umum (*public key algorithm*) karena kunci untuk enkripsi dibuat umum atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang mengetahui data yang disandikan. Jadi hanya orang tertentu saja yang berhak terhadap kunci dekripsi, walaupun kunci enkripsi dapat diketahui dan digunakan oleh orang lain. Kunci enkripsi pada algortima ini disebutkunci publik (*public key*) dan kunci dekripsi sering disebut dengan kunci pribadi atau kunci rahasia (*private key*). Proses enkripsi dengan kunci publiknya (misalkan "ek") dan proses dekripsi dengan kunci rahasianya (misalkan "dk") akan menghasilkan persamaan sebagi berikut:

Eek (P) = C (Proses Enkripsi)

Ddk (C) = P (Proses Dekripsi)

Proses enkripsi dan dekripsi dengan algoritma asimetrik dapat digambarkan sebagai berikut :



Gambar 2. 3. Enkripsi dan dekripsi algoritma Asimetris

2.2. Block Chiper

Pada *blok cipher*, plaintext yang akan disandikan dipecah menjadi blok-blok dengan panjang yang sama. Blok cipher menyandikan setiap plaintext tersebut menjadi blok ciphertext dengan proses enkripsi yang identik dan keseluruhan blok plaintext disandikan dengan kunci yang sama. Dalam sub bab ini metode blok cipher meliputi teknik penyandian dan mode operasi yang digunakan untuk menyandikan sebuah plaintext. Pada blok cipher, penyandian dilakukan pada sebuah blok plaintext dan berorientasi pada satu bit atau satu karakter pada blok plaintext tersebut. Algoritma kriptografi modern merupakan algoritma berbasis kunci. Teknik penyandian pada algoritma blok cipher modern juga mengandalkan kunci untuk kerahasiaannya.

Chiper berulang (iterated cipher) mengenkripsi blok plaintext dengan sebuah proses yang mengalami beberapa putaran (round) atau iterasi untuk mendapatkan blok ciphertext. Pada masing-masing putaran diterapkan transformasi atau fungsi putaran (round function) yang sama pada plaintext dengan menggunakan sub kunci. Fungsi tersebut biasanya merupakan gabungan dari proses substitusi, permutasi, transposisi, atau ekspansi terhadap blok plaintext. Kumpulan subkunci biasanya dihasilkan dari secret key dengan fungsi khusus dan biasanya disebut dengan daftar kunci (key schedule). Jumlah putaran dalam cipher berulang bergantung pada tingkat keamanan yang diinginkan. Dalam banyak kasus, penambahan jumlah putaran akan memperbaiki keamanan, tetapi untuk beberapa penyandian, jumlah putaran untuk mencapai keamanan yang memadai akan menjadikan penyandian tidak efisien.

Blok cipher merupakan algoritma kriptografi simetrik yang mengenkripsi satu blok *plaintext* dengan jumlah bit tertentu dan menghasilkan blok ciphertext dengan jumlah bit yang sama. Misalkan ukuran blok *plaintext* yang dienkripsi adalah 64 bit, maka akan menghasilkan *ciphertext* yang berukuran 64 bit. Pada umumnya, setiap blok *plaintext* yang diproses berukuran 64 bit. Namun, seiring dengan kemajuan teknologi, ukuran blok plaintext berkembang menjadi 128 bit, 256 bit bahkan menjadi 512 bit. Dalam proses enkripsinya, block cipher

menggunakan beberapa fungsi matematika, diantaranya fungsi permutasi dan fungsi substitusi, sehingga konfusi (confussion) dan difusi (diffusion) pada block cipher terpenuhi.

Misalkan blok plainteks (P) yang berukuran m bit dinyatakan sebagai vektor

$$P = (p_1, p_2, \ldots, p_m)$$

yang dalam hal ini p_i adalah 0 atau 1 untuk i = 1, 2, …, m, dan blok cipherteks (C) adalah :

$$C = (c_1, c_2, \ldots, c_m)$$

yang dalam hal ini c_i adalah 0 atau 1 untuk i = 1, 2, …, m.

Bila plainteks dibagi menjadi n buah blok, barisan blok-blok plainteks dinyatakan sebagai,

$$(P_1, P_2, \ldots, P_n)$$

Untuk setiap blok plainteks P_i, bit-bit penyusunnya dapat dinyatakan sebagai vektor

$$P_i = (p_{i1}, p_{i2}, \ldots, p_{im})$$

Enkripsi dan dekripsi dengan kunci K dinyatakan berturut-turut dengan persamaan

$$E_K(P) = C$$

untuk enkripsi, dan

$$D_K(C) = P$$

Fungsi E haruslah fungsi yang berkoresponden satu-ke-satu, sehingga $E^{-1} = D$. (A . Menezes, P. va n Oorschot, a nd S. Vanstone, CRC Press, 1996).

2.3. Key Dependent Secure Messenger

Key Dependent Secure Messenger adalah adalah algoitma kriptografi baru untuk tujuan enkripsi serta dekripsi yang bergantung pada kunci. Algoritma ini menggunakan teknik simetris, dimana dalam setiap proses enkripsi dan dekripsi menggunakan kunci yang sama. Kunci dibentuk pada saat pengiriman pesan dan setiap prosesnya kunci bernilai *random*. Algoritma menggunakan fungsi logika seperti XOR dan *chircular shift,* yang dapat menyebabkan perubahan dalam satu bit untuk menghasilkan perubahan bit yang signifikan.

Semua proses enkripsi dan dekripsi dilakukan pada sisi *client,* memastikan pengiriman data yang berlangsung dalam format terenkripsi melalui jaringan. Algoritma bekerja dalam 2 tahapan pada data 128 bit dan diperlukan kunci 128 bit untuk enkripsi atau dekripsi. Jika kunci atau data kurang dari 128 bit, maka sisa bit ditambahkan dengan angka nol. (Aniket D. Nandanwar et al., 2014).

2.4.1. Tahap Enkripsi

Seperti yang dinyatakan diatas, enkripsi bekerja dalam 2 tahapan sebagai berikut:

Tahap I :

1. Mengubah teks menjadi 128 bit biner.
2. Membagi 128 bit biner menjadi 2 bagian, masing-masing 64 bit.
3. Membalikkan setiap bagian dan kemudian menukar kedua bagian.
4. Menerapkan operasi *circular shift* di kedua bagian dua kali dan menggabungkan kedua bagian untuk mendapatkan data 128 bit.
5. Ambil nilai kunci, kemudian ubah menjadi 128 bit biner.
6. Melakukan operasi XOR antara teks dan nilai kunci.

Tahap II :

1. Membagi 128 bit dari hasil tahap 1 menjadi 16 bagian yang sama, masing-masing dari 8 bit.
2. Membagi setiap 8 bit blok menjadi 2 bagian, masing-masing 4 bit.
3. Menggabungkan semua 4 bit blok bagian kiri untuk mendapatkan 64 bit blok dan melakukan yang sama pada bagian kanan untuk mendapatkan blok 64 bit juga.
4. Melakukan XOR pada kedua blok, yaitu blok kiri 64 bit dengan blok kanan 64 bit. Dan *output* dikombinasikan dengan blok kanan 64 bit, untuk mendapatkan data 128 bit.
5. Ulangi proses 1 sampai 4, dimana N adalah jumlah siklus/kitaran.
6. Kemudian data 128 bit dibagi menjadi 16 blok, masing-masing 8 bit.

7. Setiap 8 bit blok kemudian dibagi menjadi 2 bagian, yang terdiri 2 bit dan 6 bit, dan lakukan operasi *circular shift* pada 6 bit dari setiap blok.
8. Menggabungkan 2 bit blok dengan 6 bit blok yang diubah untuk mendapatkan 8 bit blok (semua 16 blok).
9. Gabungkan semua blok untuk akhirnya mendapatkan 128 bit *ciphertext*.

2.4.2. Tahap Dekripsi

Seperti yang dinyatakan di atas, deskripsi juga bekerja dalam 2 tahapan sebagai berikut:

Tahap I :

1. Pilih 128 bit *chipertext.*
2. Membagi 128 bit *chipertext* menjadi 16 blok, masing-masing 8 bit.
3. Setiap 8 bit blok dibagi menjadi 2 bagian, terdiri dari 2 bit blok dan 6 bit blok.
4. Menerapkan kebalikan *circular shift* pada 6 bit blok.
5. Menggabungkan 2 bit blok dan 6 bit blok yang diubah untuk mendapatkan 8 bit blok.
6. Menggabungkan semua blok untuk mendapatkan 128 bit blok.
7. Membagi 128 bit menjadi 2 bagian, masing-masing 64 bit blok kiri dan 64 bit blok kanan.
8. Melakukan operasi XOR pada kedua blok, yaitu blok kiri 64 bit dengan blok kanan 64 bit. *output* adalah 64 blok kiri.
9. Membagi 64 bit blok kiri dan 64 bit blok kanan menjadi 8 bagian, masing-masing 4 bit blok kiri dan 4 bit blok kanan.
10. Gabungkan 4 bit blok kiri pada bagian kiri dengan 4 bit blok kiri pada bagian kanan, kemudian gabungkan juga 4 bit blok kanan bagian kiri dengan 4 bit blok kanan bagian kanan.
11. Menggabungkan semua blok untuk mendapatkan 128 bit blok.
12. Ulangi proses 7 sampai 12, dimana N adalah jumlah siklus/kitaran.
13. Kemudian menghasilkan 128 bit blok.

Tahap II :

1. Ambil nilai kunci, kemudian ubah menjadi 128 bit biner.
2. Melakukan operasi XOR antara 128 bit nilai kunci dengan 128 bit *chipertext* (hasil akhir tahap 1).
3. Membangi *chipertext* menjadi 2 bagian, masing-masing 64 bit.
4. Melakukan kebalikan dari operasi *circular shift* pada kedua bagian-bagian dua kali.
5. Menukarkan kedua bagian dan membalikkan setiap bagian.
6. Menggabungkan kedua bagian untuk mendapatkan 128 bit *plaintext*.

2.4. Instant Messaging

Instant Messaging (IM) adalah salah satu jenis layanan komunikasi yang memungkinkan seseorang untuk melakukan percakapan dengan orang lain secara *real time* melalui media internet maupun intranet. Percakapan melalui *instant messaging* umumnya berupa pesan teks, atau sering disebut "*chatting*". Namun semakin berkembangnya kemajuan teknologi, fitur pada *instant messaging* bahkan bisa digunakan untuk berbagi data, foto, video, bahkan dapat melakukan sesi telepon layaknya menggunakan telepon biasa. Dengan kemudahan tersebut maka semakin banyak orang yang menggunakan layanan *instant messaging* untuk sarana komunikasi (Aji Setiyo Sukarno, 2011).

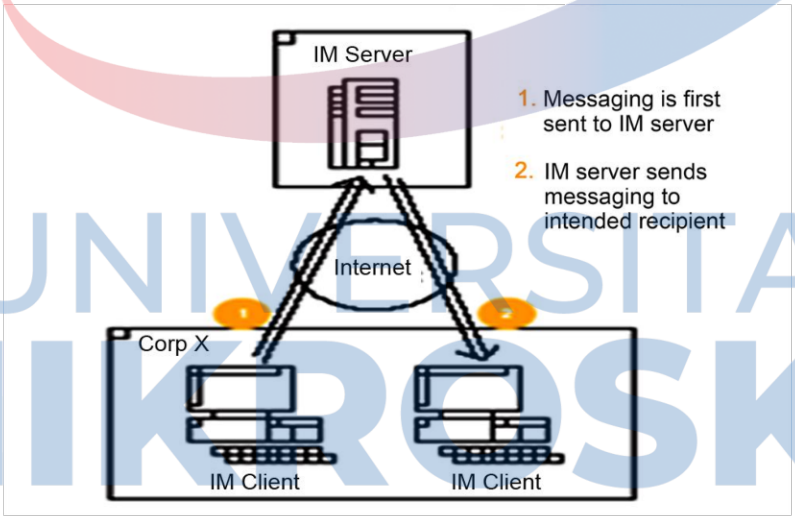
Instant messenger adalah sebuah aplikasi berbasis *Internet Protocol* (IP) yang memberikan kemudahan antar *user* yang memakai *device* berbeda dalam hal berkomunikasi. *Instant Messenger* yang paling sering digunakan adalah *instant messenger* antar komputer (Rittinghouse dan Ransome, 2005). Selain itu, sekarang ini sudah banyak *user* memakai *instant messenger* di *mobile phone* yang mendukung suara dan *video*.

Kegunaannya *instant messenger* dimulai dari adanya sebuah kebutuhan dimana *email* dinilai tidak cukup cepat sedangkan *user* ingin sebuah aplikasi yang dapat secara *instant* diterima oleh *user* lain (Rittinghouse dan Ransome, 2005). Hal tersebut yang membuat *instant messenger* semakin dikenal dan diterima dalam

lingkungan masyarakat dalam hal bekerja dan berkomunikasi antar *user,* sehingga *instant messenger* sangat berguna sebagai alat untuk komunikasi *real time* yang membuat *user* dapat secara cepat melihat siapa saja yang *online* dan dapat berkomunikasi secara langsung (Rittinghouse dan Ransome, 2005).

Meskipun memiliki bermacam-macam fitur yang disediakan oleh layanan *instant messaging*, namun tetap dapat disimpulkan secara garis besar tentang kelemahan yang dimiliki oleh *instant messaging* adalah keamanan interaksi data (Nugraha, 2010).

Pada umumnya, sistem *instant messaging* bekerja berdasarkan arsitektur *client-server*. Pengguna menjalankan *instant messaging client* pada sebuah *device* seperti komputer maupun ponsel dan kemudian program *client* tersebut akan berkomunikasi dengan *server instant messaging* yang telah disediakan oleh pengelola layanan untuk bertukar pesan atau informasi tentang pengguna dengan pengguna lain (Arie Karhendana, 2006).

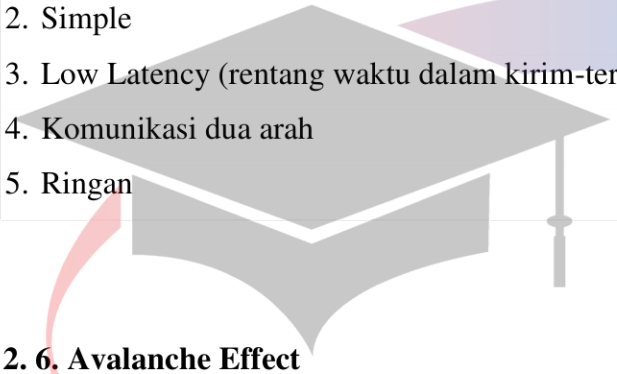


Gambar 2. 4. Instant Messaging dengan Arsitektur Client – Server (Arie Karhendana, 2006)

Web Socket adalah cara baru untuk berkomunikasi antara klien dan server tanpa mengirimkan informasi tambahan yang tidak perlu melalui protokol HTTP. Web Socket menggunakan protokolnya sendiri yang dipaparkan oleh IETF, yang mana versi terakhirnya adalah RFC 6455 (Ochtman, D., 2011).

Selain memiliki protokol sendiri, Web Socket juga memiliki API yang dapat digunakan oleh aplikasi web untuk membuka dan menutup hubungan dan untuk mengirim dan menerima pesan yang disebut sebagai WebSocket API. Dengan WebSocket API komunikasi dua arah penuh antara server dan klien yang lebih ringan dibandingkan dengan metode HTTP tradisional. Beberapa kelebihan dari Web Socket adalah sebagai berikut:

1. Native Javascript, tidak perlu tambahan library
2. Simple
3. Low Latency (rentang waktu dalam kirim-terima data)
4. Komunikasi dua arah
5. Ringan



2. 6. Avalanche Effect

Salah satu karakteristik untuk menentukan baik atau tidaknya suatu algoritma kriptografi adalah dengan melihat *avalanche effect*-nya. *Avalanche effect* adalah perubahan satu bit pada *plaintext* maupun kunci yang menyebabkan perubahan yang signifikan terhadap *ciphertext* yang dihasilkan (Schneier, B., 1996). Dengan kata lain, perubahan satu bit pada *plaintext* maupun kunci akan menghasilkan perubahan banyak bit pada *ciphertext*.

Suatu *avalanche effect* dikatakan baik jika perubahan bit yang dihasilkan berkisar antara 40%-60% dan akan sangat baik bila 50% atau lebih. Hal ini dikarenakan perubahan tersebut berarti membuat perbedaan yang cukup sulit untuk kriptanalis melakukan serangan (Schneier, B., 1996). Semakin besar *avalanche effect* akan semakin baik algoritma kriptografi tersebut. Cara menghitung *avalanche effect* sebagai berikut :

$$\textit{Avalanche Effect}\ (\text{AE}) = \frac{\text{besar perubahan bit}}{\text{jumlah keseluruhan bit}} \text{ x } 100\% \quad \text{(Schneier, B., 1996).}$$