Perancangan Aplikasi Pengelolaan Dokumen Menggunakan Prinsip Kriptografi

Eka Mukti Arifah - 13507100[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*if17100@students.if.itb.ac.id*

***Abstract* — Aplikasi untuk melakukan pengelolaan dokumen, terutama untuk berbagi dokumen (*file sharing*), perlu memperhatikan keamanan data dokumen dari berbagai serangan. Salah satu serangan ini adalah pencurian data. Untuk menghasilkan aplikasi yang tahan terhadap serangan tersebut, dapat digunakan enkripsi yang merupakan bagian dari kriptografi. Prinsip kriptografi lain seperti tanda tangan digital juga dapat digunakan dalam perancangan aplikasi untuk menyediakan aspek keamanan lainnya seperti otentikasi, integritas, serta anti-penyangkalan. Pada makalah ini akan dianalisis dan dirancang aplikasi pengelolaan dokumen yang memberikan faktor keamanan dengan menggunakan prinsip-prinsip kriptografi, seperti enkripsi, tanda tangan digital, serta skema pembagian data rahasia.**

Index Terms* — *pengelolaan dokumen, file sharing, enkripsi, tanda tangan digital

I. PENDAHULUAN

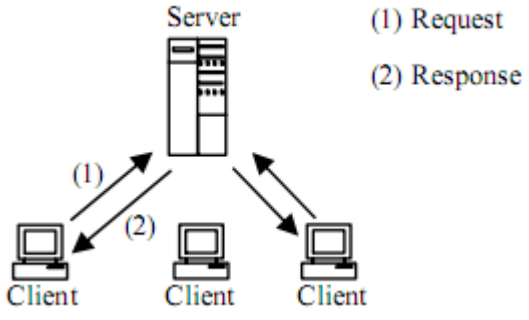
Saat ini terdapat banyak aplikasi yang memungkinkan pengguna untuk dapat melakukan pengelolaan dokumen. Dengan aplikasi ini pengguna dapat berbagi dokumen (*file sharing*) dengan pengguna lainnya secara mudah. Hal ini tentu berguna bagi organisasi atau perusahaan yang dalam keberjalanannya tidak dapat lepas dari adanya dokumen. Dokumen tersebut ada yang sifatnya rahasia ataupun tidak. Hak pengaksesannya pun ada yang terbatas hanya untuk orang-orang tertentu saja atau bersifat umum. Untuk itulah diperlukan suatu aplikasi yang dapat melakukan pengelolaan dokumen dengan mengutamakan faktor keamanan, termasuk pula kerahasiaan, dari dokumen yang dikelola tersebut.

Aplikasi pengelolaan dokumen ini biasanya menitikberatkan pada penyimpanan dokumen dan pembagiannya (*file sharing*). Kriptografi dapat digunakan untuk memberikan aspek keamanan di dalam aplikasi. Perancangan aplikasi dapat mempertimbangkan prinsip kriptografi seperti enkripsi, skema pembagian data rahasia, tanda tangan digital, serta pengelolaan kunci untuk memberikan aspek keamanan yang diinginkan. Beberapa aspek yang perlu dianalisis dalam perancangan aplikasi antara lain adalah arsitektur aplikasi, otentikasi pengguna, pengelolaan kunci, pengenkripsian dokumen, pengembalian dokumen, integritas data, serta ketahanannya terhadap serangan.

II. APLIKASI PENGELOLAAN DOKUMEN

Aplikasi pengelolaan dokumen yang dimaksud dalam makalah ini adalah aplikasi yang dapat melakukan penyimpanan (*file storage*) dan pembagian dokumen (*file sharing*). Aplikasi *file sharing* dapat dikelompokkan dalam dua jenis arsitektur, yaitu *client-server* dan *peer-to-peer*.

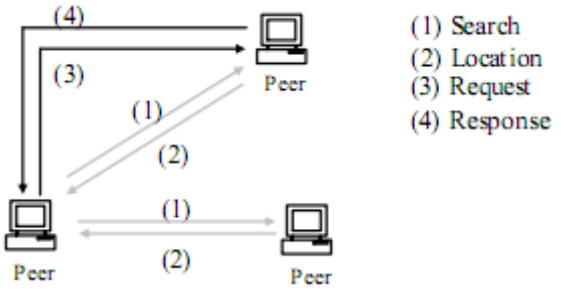
Arsitektur *client-server* menggunakan *file system* terdistribusi yang memungkinkan penyimpanan file dilakukan pada satu *server* atau lebih. Gambar 1 mengilustrasikan model ini. Pada gambar tersebut digambarkan sistem yang berbasis pada *server* pusat yang bertanggung jawab untuk memberikan layanan file secara *remote* kepada *client*. Saat menerima *request* yang valid dari *client*, *server* melakukan operasi yang sesuai kemudian mengirimkan balasan kepada *client*. Interaksi seperti ini disebut sebagai *request/response* atau *interrogation*. Layanan *file hosting* seperti Mediafire, Megaupload, Hotfile, Dropbox dan Windows Live SkyDrive menggunakan arsitektur *client-server* ini.



Gambar 1 Arsitektur Client-Server

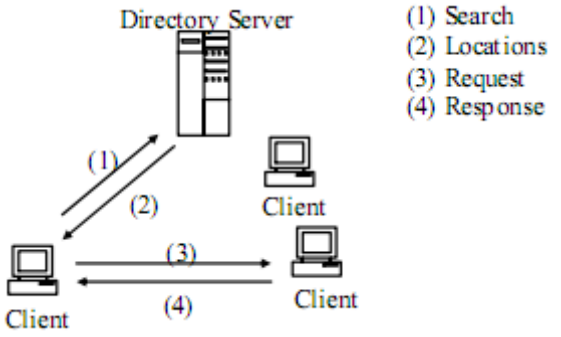
Arsitektur *peer-to-peer* menggunakan jaringan yang tidak terpusat dimana komputer *client* dapat melakukan komunikasi satu sama lain secara langsung tanpa bantuan *server* terpusat. Pada arsitektur ini, penyimpanan *file* dan informasi mengenai *client* tidak

disimpan pada *server* terpusat namun masing-masing *client* menggunakan sebuah aplikasi yang menghubungkannya secara langsung dengan *client* lainnya. Salah satu layanan *file sharing* dengan arsitektur *peer-to-peer* yang terkenal adalah Gnutella. Arsitektur yang digunakan pada Gnutella diilustrasikan pada Gambar 2. Saat seorang *client* terhubung dengan jaringan Gnutella, semua *file* yang dapat dibagikan (*sharable files*) dibuat menjadi *file* publik ke *client* lainnya melalui sebuah set folder lokal yang dispesifikasikan *client*. Transfer *file* dilakukan berdasarkan koneksi yang HTTP-*like*.



Gambar 2 Arsitektur Peer-To-Peer Gnutella

Napster, yang saat ini merupakan layanan toko musik *online*, pada awalnya adalah sebuah layanan *file sharing* dengan arsitektur gabungan antara *client-server* dan *peer-to-peer*. Napster menggunakan *server* direktori terpusat yang membangun dan memelihara daftar *file*, menambah serta menghapus data saat *client* terhubung dan tidak terhubung dengan *server*. Arsitektur ini diilustrasikan melalui Gambar 3. *Server* pusat menawarkan fasilitas pencarian terhadap suatu *file* untuk kemudian mengidentifikasi *file* yang sesuai dengan pencarian dan memberikan lokasi *file* tersebut kepada *client*. Direktori terpusat seperti ini menyebabkan pencarian dapat dilakukan dengan cepat. Setelah lokasi *file* yang dicari diketahui, *client* dapat melakukan *request* kepada *client* yang memiliki *file* tersebut melalui TCP/IP. Arsitektur hibrid yang digunakan oleh Napster memungkinkan adanya skalabilitas yang sulit dilakukan dengan arsitektur *peer-to-peer*.



Gambar 3 Arsitektur Hybrid Napster

Dalam menentukan layanan yang akan digunakan untuk pengelolaan *file*, ada beberapa hal yang biasanya menjadi pertimbangan pengguna, yaitu:

1. Fitur
   Layanan memberikan fitur yang secara praktis dapat memudahkan pengguna untuk melakukan pengunggahan, penyimpanan, pengaksesan, serta pembagian *file*. Misalnya fasilitas yang menawarkan *remote acces*, *file sharing* publik dan privat, *backup* terjadwal, serta *drag-and-drop window.*
2. Keamanan
   Dalam penyimpanan data, keamanan merupakan hal yang sangat penting. Layanan perlu memberikan fasilitas yang memungkinkan transfer *file* terenkripsi dan proteksi *password*.
3. Harga dan kapasitas penyimpanan
   Layanan pengelolaan *file* yang ada saat ini sangat beragam. Dalam memilih layanan, pengguna biasanya mempertimbangkan kesesuaian harga serta kapasitas yang disediakan. Layanan yang populer biasanya menawarkan kapasitas yang besar dengan harga yang sesuai, akun *free trial*, dan tidak menarik bayaran untuk *setup*.
4. Kemudahan penggunaan
   Pengguna dapat menggunakan layanan dengan mudah karena aplikasi bersifat *user-friendly*. Siapapun dapat belajar untuk menggunakan layanan dengan cepat.
5. *Help/Suppor*t
   Adanya fasilitas yang membantu pengguna dalam menggunakan layanan yang disediakan baik berupa *self-help* seperti FAQ, tutorial, dan *user manual* atau layanan *support* melalui email, telepon, atau *online chat*.

III. KRIPTOGRAFI

Kriptografi merupakan ilmu dan seni untuk menjaga keamanan pesan. Aspek keamanan yang disediakan kriptografi, antara lain adalah

1. Kerahasiaan pesan
2. Otentikasi
3. Integritas data
4. Anti-penyangkalan (*non-repudiation*)

Dari keempat aspek tersebut, aspek 1 dilakukan dengan melakukan enkripsi/dekripsi. Sedangkan 3 aspek lainnya dapt dilakukan dengan menggunakan tanda tangan digital. Pada bagian ini akan dijelaskan teori-teori yang berhubungan dengan prinsip kriptografi yang digunakan dalam tahap perancangan aplikasi pengelolaan dokumen selanjutnya.

A. Enkripsi

Enkripsi adalah proses untuk mengubah informasi (plainteks) menggunakan algoritma (cipher) agar tidak dapat dibaca oleh semua orang kecuali yang memiliki kunci. Hasil endkripsi disebut sebagai cipherteks.

Algoritma enkripsi dapat dibagi menjadi 2, yaitu algoritma kriptografi klasik dan modern. Algoritma

kriptografi klasik berbasis karakter. Algoritma kriptografi klasik termasuk dalam algoritma kunci simetris, yaitu algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsinya.

Algoritma kriptografi klasik dapat digolongkan ke dalam 2 jenis, yaitu cipher subtitusi dan cipher transposisi. Cipher subtitusi ada 4 macam, yaitu:

1. Cipher abjad tunggal (*monoalphabetic cipher*)
   Sebuah karakter pada plainteks diganti dengan karakter yang bersesuaian. Penggantian karaker dapat dilakukan dengan menggunakan tabel substitusi. Jumlah kemungkinan tabel substitusi yang dibuat adalah sebanyak 26!. Contoh cipher abjad tunggal adalah Caesar Cipher.
2. Cipher substitusi homofonik (*homophonic substitution cipher*)
   Algoritma ini memetakan karakter pada plainteks ke dalam salah satu cipherteks yang mungkin dari satu ke banyak dengan tujuan menyembunyikan hubungan statistik antara plainteks dan cipherteks.
3. Cipher abjad-majemuk (*polyalphabetic cipher*)
   Kunci yang digunakan untuk setiap karakter berbeda sehingga sebuah karakter pada plainteks dipetakan menjadi beberapa karakter berbeda tergantung kuncinya. Contoh algoritmanya adalah Vigenere Cipher.
4. Cipher subtitusi poligram (polygram substitution *cipher*)
   Dengan cipher ini, subtitusi dilakukan pada blok yang terdiri dari beberapa karakter. Contoh algoritmanya adalah Playfair Cipher.

Sesuai dengan namanya, untuk menghasilkan cipherteks, cipher transposisi melakukan perubahan posisi (*transpose*) pada plainteks.

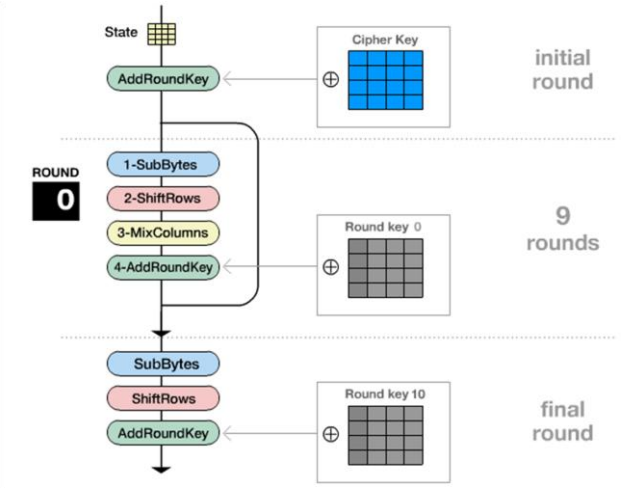
Algoritma kriptografi modern beroperasi dalam mode bit. Baik plainteks, cipherteks, maupun kunci diproses dalam rangkaian bit. Algoritma enkripsi berbasis bit dapat dikategorikan menjadi 2, yaitu:

1. Cipher Aliran (*Stream Cipher*)
   Pada cipher aliran, operasi dilakukan pada bit tunggal. Enkripsi dan dekripsi dilakukan bit per bit. Bit-bit kunci yang digunakan disebut sebagi *keystream*. *Keystream* ini dibangkitkan oleh *keystream generator*. Keamanan cipher aliran bergantung sepenuhnya pada *keystream generator* ini. Semakin acak keluaran yang dihasilkan oleh pembangkit aliran-bit-kunci, maka akan semakin sulit pula bagi kriptanalis untuk memecahkan cipherteks.
2. Cipher Blok (*Block Cipher*)
   Pada cipher blok, operasi dilakukan per blok. Ukuran blok yang dapat digunakan misalnya 64 bit karakter/blok (8 karakter/blok). Enkripsi dan dekripsi dilakukan per blok. Panjang kunci enkripsi sama dengan panjang blok yang digunakan. Mode operasi blok yang digunakan ada 4 macam, yaitu *Electronic Code Book* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback* (CFB), dan *Output Feedback* (OFB).

Salah satu algoritma kriptografi modern yang digunakan dalam aplikasi pengelolaan dokumen adalah AES. Contohnya pada layanan Dropbox, dokumen yang disimpan adalah dokumen yang telah dienkripsi dengan algoritma AES-256.

Advanced Encryption Standard (AES) termasuk dalam algoritma kriptografi simetri berbasis cipher blok. Pada November 2001, algoritma Rjindael ditetapkan sebagai AES. Algoritma Rjindael mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit. Panjang kunci dan ukuran blok dapat dipilih secara independen. Setiap blok dienkripsi dalam sejumlah putaran tertentu sebagaimana halnya pada DES. Karena AES menetapkan panjang kunci adalah 128, 192, dan 256 bit, maka dikenal adanya AES-128, AES-192, dan AES-256. Namun, secara de fakto hanya ada dua varian AES, yaitu AES-128 dan AES-256 karena sangat jarang pengguna menggunakan kunci dengan panjang 192 bit.

Rjindael beroperasi dalam orientasi *byte*. Setiap putaran menggunakan kunci internal yang berbeda yang disebut sebagai *round key*. Cara kerja algoritma Rjindael dapat dijelaskan melalui gambar berikut:



Gambar 4 Algoritma Rjindael

Garis besar Algoritma Rjindael yang beroperasi pada blok 128 bit dengan kunci 128 bit adalah sebagai berikut:

1. *AddRoundKey*, yaitu melakukan XOR antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga sebagai *initial round*.
2. Putaran sebanyak Nr – 1 kali, proses yang dilakukan dalam setiap putaran adalah:
   a. *SubBytes*: substitusi *byte* dengan menggunakan tabel substitusi (S-box).
   b. *ShiftRows*: pergeseran baris-baris *array state* secara *wrapping*.
   c. *MixColumns*: mengacak data di masing-masing kolom *array state*.

d. *AddRoundKey*: melakukan XOR antara *state* sekarang *round key*.

3. *Final Round*, untuk proses putaran terakhir:
   a. *SubBytes*
   b. *ShiftRows*
   c. *AddRoundKey*

B. Tanda Tangan Digital

Untuk melakukan pemberian tanda tangan digital pada dokumen, dapat dilakukan dengan 2 cara, yaitu:

1. Enkripsi pesan
   a. Enkripsi dengan kriptografi simetri

      Pesan yang dienkripsi dengan algoritma simetri sudah memberikan solusi untuk otentikasi pengirim dan keaslian pesan karena kunci simetri hanya diketahui oleh pengirim dan penerima. Namun cara ini tidak menyediakan mekanisme untuk anti-penyangkalan sehingga diperlukan pihak ketiga yang dipercaya oleh pengirim/penerima. Pihak ketiga ini disebut penengah (arbitrase).

      Misalkan BB (*Big Brothers*) adalah otoritas arbitrase yang dipercaya oleh Alice dan Bob. BB memberikan kunci rahasia K_A kepada Alice dan kunci rahasia K_B kepada Bob. Hanya Alice dan BB yang mengetahui K_A, begitu juga hanya Bob dan BB yang mengetahui K_B. Jika Alice bekirim pesan P kepada Bob, maka langkah-langkahnya adalah sebagai berikut:

      1. Alice mengenkripsi pesan M untuk Bob dengan K_A, lalu mengirim cipherteksnya ke BB.
      2. BB melihat bahwa pesan dari Alice, lalu mendekripsi pesan dari Alice dengan K_A.
      3. BB membuat pernyataan S bahwa ia menerima pesan dari Alice, lalu menambahkan pernyataan tersebut pada plainteks dari Alice.
      4. BB mengenkripsi bundel pesan (M + S) dengan K_B, lalu mengirimkannya kepada Bob.
      5. Bob mendekripsi bundel pesan dengan K_B. Ia dapat membaca pesan dari Alice (M) dan pernyataan (S) dari BB bahwa Alice yang mengirim pesan tersebut.

      Jika Alice menyangkal telah mengirim pesan tersebut, maka pernyataan dari BB pada pesan yang diterima oleh Bob digunakan untuk menolak penyangkalan Alice. Dan karena hanya BB dan Alice yang mengetahui kunci rahasia K_A, maka hanya Alice yang dapat mengenkripsi pesan dengan kunci terebut. Sehingga BB mengetahui bahwa pesan tersebut memang berasal dari Alice, bukan orang lain.

   b. Enkripsi dengan kriptografi kunci publik

      Pada enkripsi biasa, yang dilakukan adalah pengenkripsian pesan dengan kunci publik penerima dan dekripsi dengan kunci privat penerima. Cara ini tidak memberikan sarana otentikasi karena kunci publik diketahui oleh banyak orang. Untuk melakukan enkripsi sebagai tanda-tangan, dilakukan pengenkripsian pesan dengan kunci privat pengirim dan dekripsi pesan dengan kunci publik pengirim. Dengan cara ini, kerahasiaan pesan dan otentikasi dapat dicapai sekaligus. Dengan algoritma kunci-publik, penandatanganan pesan tidak membutuhkan lagi pihak penengah (arbitrase). Ide ini ditemukan oleh Diffie dan Hellman. Tahapan yang dilakukan adalah:

      - Proses menandatantangani pesan (oleh pengirim): $\boldsymbol{S = E_{SK}(M)}$
      - Proses membuktikan otentikasi pesan (oleh penerima): $\boldsymbol{M = D_{PK}(S)}$

      Beberapa agoritma kunci-publik dapat digunakan untuk menandatangani pesan dengan cara mengenkripsinya, asalkan algoritma tersebut memenuhi sifat:

      $$\boldsymbol{D_{SK}(E_{PK}(M)) = M} \text{ dan } \boldsymbol{D_{PK}(E_{SK}(M)) = M}$$

      Keterangan:
      SK = *secret key* = kunci privat pengirim
      PK = *public key* = kunci publik pengirim
      E = fungsi enkripsi
      D = fungsi dekripsi
      M = pesan semula
      S = *signature* = hasil enkripsi pesan

2. Kombinasi fungsi *hash* dan kriptografi kunci publik

Penandatanganan pesan dengan cara enkripsi memberikan dua fungsi berbeda, yaitu kerahasiaan dan otentikasi pesan. Pada beberapa kasus, seringkali otentikasi diperlukan tetapi kerahasiaan pesan tidak. Maksudnya, pesan tidak perlu dienkripsikan karena yang dibutuhkan hanya keotentikan pesan saja. Algoritma kunci-publik dan fungsi hash dapat digunakan untuk kasus seperti ini.

Berikut ini adalah langkah-langkah pemberian tanda-tangan:

1. Pengirim menghitung nilai *hash* dari pesan M yang akan dikirim, misalkan nilai *hash* dari M adalah h.
2. Pengirim mengenkripsi h dengan kunci privatnya menggunakan persamaan enkripsi RSA: $S = \boldsymbol{h^{SK} \bmod n}$. *SK* adalah kunci privat pengirim dan n adalah modulus ($n = pq$, p dan q adalah dua buah bilangan prima).
3. Pengirim mentransmisikan $M + S$ ke penerima

Sedangkan langkah-langkah untuk melakukan verifikasi tanda tangan digital adalah sebagai berikut:

1. Penerima menghitung nilai *hash* dari pesan M yang akan dikirim, misalkan nilai *hash* dari M adalah h'.
2. Penerima melakukan dekripsi terhadap tanda-tangan S dengan kunci publik si pengirim menggunakan persamaan dekripsi RSA: $\boldsymbol{h = S^{PK} \bmod n}$. *PK* adalah kunci privat pengirim dan n adalah modulus ($n = pq$, p dan q adalah dua buah bilangan prima).
3. Penerima membandingkan h dengan h'. Jika $h = h$'

maka tanda-tangan digital adalah otentik. Jika tidak sama, maka tanda-tangan tidak otentik sehingga pesan dianggap tidak asli lagi atau pengirimnya

Keotentikan pesan dilihat dari verifikasi tanda tangan digital, hal ini dapat dijelaskan sebagai berikut:

a. Apabila pesan M yang diterima sudah berubah, maka MD' yang dihasilkan dari fungsi hash berbeda dengan MD semula. Ini berarti pesan tidak asli lagi.
b. Apabila pesan M tidak berasal dari orang yang sebenarnya, maka message digest MD yang dihasilkan dari persamaan 3 berbeda dengan message digest MD' yang dihasilkan pada proses verifikasi karena kunci publik yang digunakan oleh penerima pesan tidak berkoresponden dengan kunci privat pengirim.
c. Bila MD = MD', ini berarti pesan yang diterima adalah pesan yang asli (*message authentication*) dan orang yang mengirim adalah orang yang sebenarnya (*user authentication*).

Dua algoritma tanda tangan digital yang digunakan secara luas adalah RSA dan ElGamal. Pada RSA, algoritma enkripsi dan dekripsi identik sehingga proses *signature* dan verifikasi juga identik. Selain *RSA*, terdapat algoritma yang dikhususkan untuk tanda-tangan digital, yaitu *Digital Signature Algorithm* (DSA), yang merupakan standar untuk *Digital Dignature Standard* (DSS). Pada *DSA*, algoritma *signature* dan verifikasi berbeda

IV. Analisis Dan Perancangan Aplikasi

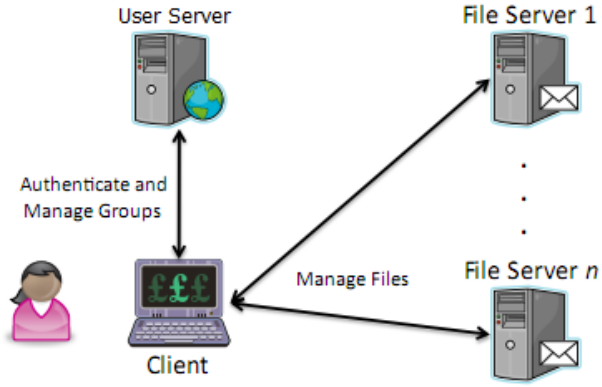
Fungsi yang diharapkan pada aplikasi yang dirancang pada makalah ini adalah sebagai berikut:

1. Penyimpanan dokumen (*file storage*)
   Aplikasi dapat melakukan penyimpanan dokumen dengan menjaga keamanan data dokumen yang disimpan terhadap berbagai serangan. Serangan ini antara lain adalah pencurian data oleh orang yang tidak berhak.
2. Pembagian dokumen (*file sharing*)
   Aplikasi dapat melakukan *sharing file*, baik bersifat publik ataupun privat. Pengguna dapat menentukan apakah dokumen tersebut dapat dilihat oleh seseorang, grup, atau semua orang yang mengaksesnya.

Dalam makalah ini, arsitektur aplikasi yang diusulkan adalah arsitektur *client-server* dimana layanan bekerja pula sebagai layanan *file hosting*. Hal ini disebabkan karena keamanan data dan otentikasi pengguna pada arsitektur *peer-to-peer* lebih sulit untuk dikelola. Pada arsitektur yang diusulkan, seluruh dokumen akan disimpan pada *server* terpusat. *Server* terpusat ini dapat terdiri dari satu server atau lebih.

Agar memudahkan proses *file sharing*, didesain adanya sebuah *user server* yang bertugas untuk mengautentikasi pengguna atau grup yang berhak mengakses dokumen yang disimpan pada *file server*.

Jika digambarkan, maka arsitektur dapat diilustrasikan sebagai berikut:



Gambar 5 Arsitektur Aplikasi

Untuk menjaga keamanan data, layanan hanya menyimpan dokumen yang telah dienkripsi. Agar terhindar dari penyadapan data selama transfer data dari *client* ke *server* maupun sebaliknya, maka enkripsi dan dekripsi data dilakukan pada sisi *client*. Hal ini juga dilakukan agar kerja *server* tidak terlalu berat saat diakses oleh banyak orang. Enkripsi dilakukan dengan menggunakan AES-256. Kunci yang digunakan dalam proses enkripsi maupun dekripsi adalah kunci yang disimpan oleh pengguna. Kunci ini dapat dikembalikan oleh aplikasi jika pengguna masuk ke sistem dengan *username* dan *password* yang sesuai. Hal ini juga diatur oleh *user server*.

Untuk melakukan otentikasi serta integritas data, setiap dilakukan penyimpanan dokumen juga dilakukan penyimpanan data lain seperti tanggal penyimpanan dan tanda tangan digital. Aplikasi memungkinkan validasi tanda tangan digital dokumen yang diunduh oleh pengguna. Jika tanda tangan digital dokumen tidak valid, maka aplikasi dapat memperingatkan pengguna mengenai hal ini sehingga pengguna dapat memilih untuk terus melanjutkan pengunduhan atau tidak.

Tanda tangan digital pada aplikasi juga dapat digunakan untuk melakukan pengecekan terhadap perubahan jika dokumen tersebut sudah ada pada server. Untuk mengecek adanya perubahan, aplikasi tidak perlu mencocokkan keseluruhan isi file, namun dengan mengecek tanda tangan digitalnya. Dengen pengecekan ini, aplikasi dapat melakukan *update* penyimpanan *file* ke *server* tanpa perlu mengunggah dokumen secara keseluruhan. Hal ini juga dapat menghemat *bandwith*. Penerapan hal ini telah dilakukan pada layanan Dropbox.

V. Kesimpulan

1. Keamanan data adalah hal yang sangat penting dalam aplikasi pengelolaan dokumen.
2. Aplikasi pengelolaan dokumen dapat menggunakan prinsip kriptografi, seperti enkripsi dan tanda tangan digital untuk memberikan aspek keamanan.
3. Arsitektur yang diusulkan untuk memberikan aspek

keamanan adalah arsitektur *client-server* dengan *server* khusus untuk otentikasi penggun. Server ini terpisah dengan *file server* untuk penyimpanan dokumen.

4. Untuk mengurangi kerja server, enkripsi dan dekripsi dilakukan pada pihak *client*.
5. Selain untuk memberikan aspek otentikasi dan integritas, tanda tangan digital ataupun *message digest* dapat digunakan sebagai alat bantu yang memudahkan pembaharuan *file* pada sisi *server* sehingga tidak perlu dilakukan pengunggahan ulang yang mengonsumsi *bandwith* lebih besar.

REFERENCES

[1] Brown, D.J., *A Secure and Distributed Architectural Model For File Sharing*, Napier University: 2002.

[2] Fu, K. E., *Group Sharing and Random Access in Cryptographic Storage File Systems*. MIT: 1998.

[3] Parashar, M, et. al. "Evaluating Security Mechanisms in Peer To Peer Applications", Rutgers University

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Mei 2011

Eka Mukti Arifah
13507100