Program Penelitian Pemula

LAPORAN PENELITIAN PEMULA

APLIKASI KRIPTOGRAFI UNTUK SISTEM KEAMANAN DATA REKAM MEDIS PASIEN PADA WEB RUMAH SAKIT MENGGUNAKAN *XML DATABASE XINDICE*

Oleh:
Edy Winarno, S.T., M.Eng.
Veronica Lusiana, S.T., M.Kom.
Wiwien Hadikurniawati, S.T., M.Kom.

Dibiayai Dinas Pendidikan
Provinsi Jawa Tengah
Tahun Anggaran 2010

FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS STIKUBANK SEMARANG
TAHUN 2010

Program Penelitian Pemula

LAPORAN PENELITIAN PEMULA

APLIKASI KRIPTOGRAFI UNTUK SISTEM KEAMANAN DATA REKAM MEDIS PASIEN PADA WEB RUMAH SAKIT MENGGUNAKAN *XML DATABASE XINDICE*

Oleh:
Edy Winarno, S.T., M.Eng.
Veronica Lusiana, S.T., M.Kom.
Wiwien Hadikurniawati, S.T., M.Kom.

Dibiayai Dinas Pendidikan
Provinsi Jawa Tengah
Tahun Anggaran 2010

FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS STIKUBANK SEMARANG

TAHUN 2010

HALAMAN PENGESAHAN LAPORAN PENELITIAN PEMULA

1.	a. Judul Penelitian	: Aplikas Kriptografi untuk Sistem Keamanan Data Rekam Medis Pasien pada Web Rumah Sakit menggunakan *XML Database Xindice.*
	b. Bidang Ilmu	: Teknologi Informasi
	c. Kategori Penelitian	: Rekayasa
2.	Ketua Peneliti	
	a. Nama Lengkap dan Gelar	: Edy Winarno, S.T., M.Eng.
	b. Jenis Kelamin	: Laki-laki
	c. Golongan/Pangkat/NIP	: Penata Muda / III A / Yu.2.04.10.071
	d. Jabatan Fungsional	: Asisten Ahli
	e. Jabatan Struktural	: -
	f. Fakultas/Jurusan	: Teknologi Informasi / Sistem Informasi
	g. Pusat Penelitian	: LPPM UNISBANK Semarang
3.	Alamat Ketua Peneliti	
	a. Alamat kantor	: Jl. Tri Lomba Juang No. 1 Semarang
	Telepon/Faks	: (024)8311668 Fax: (024)8443240
	E-Mail	: info@unisbank.ac.id
	b. Alamat rumah	: Perum Klipang Pesona Asri III E-46 Tembalang Semarang
	Telepon/Faks	: 085865206752
	E-Mail	: winarnoedy@gmail.com
4.	Jumlah Anggota Peneliti	: 2 orang
	a. Nama Anggota	: 1. Veronica Lusiana, S.T., M.Kom. 2. Wiwien Hadikurniawati, S.T., M.Kom.
5.	Lokasi Penelitian	: Laboratorium Komputer FTI-UNISBANK
6.	Kerjasama dengan institusi lain	: -
7.	Lama Penelitian	: 7 (tujuh) bulan
8.	Biaya yang diperlukan	
	a. Dinas Pendidikan Provinsi Jateng	: Rp. 7.000.000,-
	b. Sumber lain	: Rp. 0,-
	Jumlah	: Rp. 7.000.000,- *(tujuh juta rupiah rupiah)*

Semarang, Oktober 2010

Menyetahui ,
Dekan Fakultas Teknologi Informasi

(Dwi Agus Diartono, M.Kom)
NIY: Y.2000.03.054

Ketua Peneliti,

(Edy Winarno, S.T., M.Eng.)
NIY: YU.2.04.10.071

Menyetujui,
Ketua LPPM UNISBANK,

(Dr. Dra. Ec. Liana, M.MSI)
NIY: Y.2.92.07.085

ABSTRAK

Untuk mengirimkan data atau informasi yang diolah menggunakan teknik kriptografi salah satunya menggunakan data XML yang merupakan singkatan dari *eXtensible Markup Language*. Data XML merupakan file teks biasa yang berisikan berbagai *tag* yang didefinisikan sendiri oleh pembuat dokumen XML. Dalam penelitian ini data XML diolah menggunakan teknik kriptografi *Password Based Encryption*. Metode ini merupakan metode enkripsi dan dekripsi untuk mengubah data *plaintex* menjadi *chipertex* dan menghasilkan kembali data *chipertex* dari data *plaintex* menggunakan kunci *password*.

Dalam menyimpan dan mengolah data digunakan XML *database* Xindice yaitu sebuah sistem basis data yang dapat mengakomodasi data-data dalam format XML. XML *database* Xindice memberikan kemudahan dalam memberikan perintah atau *query* terhadap struktur data XML yang telah disimpan. XML *Database* Xindice dibuat untuk pengembangan dokumen XML.

Dalam penelitian ini dibuat sebuah implementasi program berupa sebuah *prototype web* rumah sakit yang dilengkapi dengan menu enkripsi dan dekripsi data untuk mengamankan data-data yang disimpan agar dapat diakses dan diolah oleh pihak yang mempunyai kepentingan terhadap data tersebut. Implementasi program dibuat dalam sebuah aplikasi *web* menggunakan *Java Server Page* (JSP) pada NetBeans 6.0 yang dilengkapi dengan enkripsi dan dekripsi metode *Password Based Encryption* yang telah disediakan oleh *Java Cryptography Extension* (JCE).

Kata Kunci :
XML, XML *database* Xindice, *Password Base Encryption*, enkripsi, dekripsi.

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunia yang telah diberikan sehingga kami dapat menyelesaikan laporan penelitian pemula yang dibiayai oleh Diknas Pendidikan Pemprov Jawa Tengah dengan lancar dan baik. Laporan ini adalah hasil penelitian yang kami kerjakan dengan judul "Aplikasi Kriptografi Untuk Sistem Keamanan Data Rekam Medis Pasien Pada Web Rumah Sakit Menggunakan *XML Database Xindice*.

Pada kesempatan ini kami ingin mengucapkan terima kasih yang sedalam-dalamnya atas bimbingan, arahan, bantuan dan petunjuk yang telah diberikan kepada kami, kepada :

1. Dinas Pendidikan Pemerintah Provinsi Jawa Tengah yang telah memfasilitasi proyek penelitian ini.
2. Rektor Universitas Stikubank (UNISBANK) Semarang, atas izin dan kesempatan yang diberikan kepada penulis untuk mengikuti penelitian program fasilitasi Diknas Pemprov Jateng ini.
3. Ketua Lembaga Penelitian dan Pengabdian Universitas Stikubank Semarang atas dorongan dan dukungannya pada penelitian ini.
4. Dekan Fakultas Teknologi Informasi Universitas Stikubank Semarang yang telah memberikan izin untuk mengikuti proyek penelitian ini.
5. Teman-teman kerja Fakultas Teknologi Informasi Universitas Stikubank yang telah mendukung, membantu dan memberikan *support* kepada kami.
6. Semua pihak yang tidak dapat kami sebutkan satu-persatu, yang telah membantu di dalam menyelesaikan laporan penelitian ini.

Semoga laporan penelitian ini dapat memberikan manfaat kepada para pembaca dan menambah wacana khususnya tentang enkripsi data XML menggunakan kriptografi PBE. Laporan penelitian ini masih banyak kekurangan dan masih jauh dari kesempurnaan sehingga penulis mengharapkan saran dan kritik yang akan membantu bagi perbaikan dan pengembangan penelitian selanjutnya.

Semarang, Oktober 2010

Tim Peneliti

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar belakang

Untuk mengirimkan informasi atau data yang diolah menggunakan teknik kriptografi salah satunya menggunakan data XML yang merupakan singkatan dari *eXtensible Markup Language* (Noprianto,2004). Sebuah data XML merupakan sebuah file teks biasa yang berisikan berbagai tag yang didefinisikan sendiri oleh pembuat dokumen XML tersebut.

Data XML sering disimpan sebagai file data dalam sebuah *filesystem* atau sebagai teks atau karakter dalam *relational databases*. Namun, data XML dalam ukuran yang besar akan sulit dalam pengolahan datanya. Penggunaan teks dalam sebuah *relational databases* dapat dengan mudah disimpan sebagai sebuah data, tapi akan kesulitan ketika memberikan perintah atau *query* terhadap struktur data XML yang telah disimpan. *Database* yang mampu mengolah dan bisa mengakomodasi data-data XML adalah XML *Database* Xindice.

Dibandingkan dengan RDBMS, penggunaan XML *Database* Xindice lebih memudahkan dalam pengolahan data aplikasi yang berupa data XML. Karena dengan dipergunakannya *server* basis data *native* XML, aplikasi tidak perlu melakukan pemetaan dari data XML ke struktur data tertentu. Aplikasi dapat langsung menyimpan data XML, dan mendapatkan kembali data XML dalam format yang tidak memerlukan perubahan.

Data XML ini kemudian diolah menggunakan proses enkripsi dan dekripsi menggunakan teknik kriptografi. Data XML digunakan untuk proses enkripsi dan dekripsi yang dilakukan pada tag tertentu sesuai yang diperlukan. Enkripsi dan dekripsi tidak dilakukan pada sebuah kesatuan dokumen XML, tapi dilakukan dengan memilih pada tag-tag tertentu sesuai yang diinginkan.

Teknik kriptografi adalah teknik pengiriman data yang dikirimkan ke tempat tujuan lain dan disamarkan sedemikian rupa sehingga data itu tidak bisa dimengerti oleh pihak yang tidak berhak. Data asli yang akan dikirimkan dan belum mengalami penyandian disebut dengan istilah *plaintext*, dan setelah disamarkan dengan suatu cara penyandian maka *plaintext* ini akan berubah

menjadi *ciphertext*. Teknik untuk mengubah *plaintext* menjadi *ciphertext* disebut sebagai enkripsi *(encryption)*, sedangkan teknik untuk mengubah kembali *chipertext* menjadi *plaintext* disebut dekripsi *(decryption)*.

Salah satu contoh aplikasi dari teknik kriptografi dalam penelitian ini digunakan pada proses penyandian data rekam medis pada sebuah web rumah sakit yang memiliki beberapa bagian unit kerja dimana masing-masing bagian tersebut memiliki kewenangan masing-masing untuk mengetahui, menyimpan dan memproses data pasien sehingga tingkat kerahasiaan dan keamanan data pasien dapat dijaga. Setiap bagian dari rumah sakit tersebut memiliki kunci untuk dapat mengenkripsi dan mendekripsikan data yang diterima, sehingga tiap bagian mempunyai kewenangan masing-masing terhadap kerahasiaan dan keamanan data pasien.

Dalam proses enkripsi dan dekripsi data terdapat beberapa metode, salah satunya menggunakan metode *Password Based Encryption*. Metode *Password Based Encryption* merupakan metode enkripsi dengan algoritma kunci simetrik yang menggunakan kesepakatan kunci yang sama untuk proses enkripsi dan dekripsinya.

1.1.1 Perumusan masalah

Berdasarkan latar belakang diatas, maka dapat dirumuskan permasalahan penelitian sebagai berikut:

a. Membuat contoh data rekam medis pasien dari sebuah rumah sakit menggunakan XML
b. Membuat sebuah basis data XML menggunakan XML *database* Xindice untuk pemrosesan data XML yang digunakan dalam aplikasi web.
c. Membuat program menggunakan Java untuk melakukan proses enkripsi dan dekripsi data XML menggunakan kriptografi *Password Based* Encryption untuk melakukan proses enkripsi dan dekripsi pada tag yang diperlukan.
d. Mengaplikasikan seluruh program dalam sebuah aplikasi web menggunakan *Java Server Page* (JSP) pada NetBeans 6.0

e. Menguji proses enkripsi dan dekripsi data tersebut menggunakan model *client* dan *server* pada suatu jaringan komputer

1.1.2 Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat memberikan manfaat dan menjadi referensi bagi kegiatan penelitian yang berhubungan dengan:

a. Membuat data XML yang aman dengan cara enkripsi
b. Membuat aplikasi dari program kriptografi menggunakan XML database Xindice
c. Membuat pendistribusian data XML yang terenkripsi melalui internet, intranet, atau fasilitas pertukaran data yang diakses oleh orang banyak, tanpa dikhawatirkan adanya pihak yang tidak berkepentingan untuk membacanya.

1.2.Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Dapat menggunakan XML *Database* Xindice sebagai basis data untuk menyimpan dan mengolah data-data berformat XML.
2. Dapat melakukan proses enkripsi dan dekripsi (Kriptografi) pada tag tertentu pada data XML yang digunakan
3. Dapat membuat dan mengaplikasikan data XML yang telah terenkripsi pada *database XML* tersebut pada sebuah implementasi web.

1.3. Batasan Masalah

Batasan masalah yang diberikan dalam penelitian ini adalah:

1. Penelitian hanya membahas penggunaan XML *Database* Xindice sebagai basis data yang digunakan untuk menyimpan dan mengolah data XML.
2. Penelitian hanya membahas implementasi enkripsi dan dekripsi pada tag-tag data XML pada sebuah contoh data rekam medis pasien di sebuah rumah sakit.
3. Penelitian ini tidak membahas masalah sistem keamanan jaringan secara luas.

BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Beberapa penelitian yang berhubungan dengan penggunaan data XML dalam proses aplikasi dan penggunaan teknik kriptografi diberikan sebagai berikut.

Triono (2006), melakukan penelitian mengenai penerapan XML *Web Service* dalam sistem informasi akademik. Penelitian ini bertujuan untuk merancang suatu sistem informasi akademik dengan menggunakan data XML dalam proses aplikasinya. Penelitian ini dilakukan dengan membuat suatu rancangan model sistem informasi akademik sebagai layanan data pada sebuah *web service* untuk memudahkan dalam proses pengolahan data karena menggunakan format data XML yang terstruktur. Dalam penelitian ini belum dibuat sistem keamanan untuk distribusi data tiap bagian.

Hartono (2003), melakukan penelitian mengenai pemakaian kriptografi kunci publik dengan algoritma RSA untuk keamanan data XML. Penelitian bertujuan untuk membuat rancangan sistem distribusi data XML yang aman menggunakan proses kriptografi kunci publik dengan algoritma RSA. Data XML digunakan sebagai data yang terstruktur untuk dapat diproses menggunakan metode enkripsi dan dekripsi. Metode enkripsi yang digunakan menggunakan metode kriptografi kunci publik sehingga data XML yang telah dibuat dapat dienkripsi dan didekripsi sesuai dengan *tag* yang dikehendaki. Namun dalam penelitian ini belum dibuat suatu model implementasi program dengan fasilitas yang baik dan interaktif untuk digunakan oleh pemakai.

Wijayanto (2002), melakukan penelitian mengenai implementasi *public key cryptography* dan skema e-*voting* untuk e-*questionaire* pada *internet*. Penelitian bertujuan untuk membuat sebuah implementasi kriptografi dan skema voting pada kuesioner menggunakan *internet*. Dalam penelitian ini digunakan kriptografi metode kunci publik dalam proses enkripsi dan dekripsi data. Namun

dalam aplikasi kriptografi belum menggunakan format data XML sebagai data implementasinya.

Dari beberapa penelitian diatas dapat dikembangkan oleh penulis menjadi sebuah penelitian mengenai implementasi teknik kriptografi untuk keamanan distribusi data XML. Implementasi yang akan digunakan adalah penerapan kriptografi *password based encryption* untuk keamanan distribusi data menggunakan XML *database* Xindice pada data rekam medis sebuah rumah sakit. Implementasi program berupa *prototype* web sebuah rumah sakit yang dilengkapi dengan menu enkripsi dan dekripsi data untuk mengamankan data-data yang disimpan agar dapat diakses dan diolah oleh pihak yang hanya mempunyai kepentingan terhadap data tersebut.

2.2. Landasan Teori

2.2.1. *Extensible Markup Language* (XML)

XML atau *Extensible Markup Language* merupakan bahasa *markup* yang dikembangkan oleh W3C (*World Wide Consorsium*), yaitu organisasi yang menetapkan standarisasi untuk Web.Yang dimaksud bahasa *markup* adalah bahasa yang dipergunakan untuk menandai (*markup*) data sehingga data tersebut bisa ditampilkan atau diolah untuk tujuan tertentu. Contoh bahasa *markup* yang lain misalnya HTML. XML bukanlah program, atau pustaka. XML adalah sebuah teknologi, sebuah standar dengan berbagai aturan tertentu. Dalam pengertian yang sederhana, sebuah dokumen XML hanyalah sebuah file teks biasa yang berisikan berbagai tag yang didefinisikan sendiri oleh pembuat dokumen XML tersebut. Sesuai dengan namanya, *eXtensible Markup Language*, sebuah dokumen XML adalah sebuah dokumen dengan *markup*, sama seperti halnya dengan HTML.

XML tidak didesain untuk menggantikan HTML, XML lebih dirancang untuk mendeskripsikan data dan memfokuskan diri pada data tersebut. Sementara, HTML didesain untuk menampilkan data dan memfokuskan diri pada bagaimana data ditampilkan. Jadi XML bukan pengganti HTML karena keduanya memang dirancang berbeda. Hubungan antara XML dan HTML lebih ke arah pelengkap. Sebuah dokumen XML dapat disimpan menjadi sebuah data dan dapat menggunakan HTML untuk menampilkan data tersebut.

Dokumen XML terdiri dari berbagai *tag* seperti pada HTML.Tag-tag pada XML tidak memiliki standar khusus. Untuk menjaga agar tag-tag tersebut tetap berada di dalam lingkup jalan yang benar, maka keseluruhan aturan tag disimpan di dalam *Document Type Definition* (DTD) atau XML *Schema*. Dengan adanya aturan tersimpan di DTD atau XML Schema, maka sebuah dokumen XML diharapkan akan mampu mendeskripsikan diri sendiri (*self descriptive*). Boleh disamakan dengan tag HTML yang telah memiliki standar, tetapi pada XML akan memiliki arti yang lebih luas.

Dengan menyimpan informasi tersebut dalam file XML, informasi dapat diteruskan ke siapa saja, dengan perantaraan apapun juga. Selama sistem tujuan bisa memproses XML tersebut, maka informasinya juga akan sampai kepada sistem tujuan. Untuk memroses tag-tag XML tersebut tidak perlu dilakukan secara manual. Terdapat banyak cara untuk melakukannya. Algoritma yang digunakan juga berbeda-beda.

Dengan prinsip ini, pertukaran informasi akan menjadi lebih mudah. Dapat dikatakan sebuah sistem menggunakan XML sebagai *format* filenya. Ketika akan dikirimkan ke sistem lain yang benar-benar tidak kompatibel (misal dari *Mainframe* ke PDA), format XML ini tetap bisa diandalkan karena informasi tetap akan terjaga.

2.2.1.1. Dokumen XML

Pada XML, semua tag yang terdapat didalamnya tidak satupun menjelaskan tentang presentasi data atau bagaimana data ditampilkan. Tag-tag yang terdapat di dalam sebuah dokumen XML hanya menjelaskan tentang data itu sendiri secara terstruktur. Dari contoh dokumen XML berikut, dapat dilihat bahwa elemen-elemen dapat dibuat dan diberi nama sendiri. *DATARUMAHSAKIT, DATAPASIEN, NOMORINDUKPASIEN, NAMAPASIEN, NOMORIDENTITAS, TANGGALLAHIR, JENISKELAMIN, GOLONGANDARAH, TINGGIBERATBADAN, ALAMAT, TELEPONPASIEN, PENYAKIT, MULAISAKIT* dan *ALERGIOBAT* merupakan elemen-elemen yang didefinisikan sendiri pada saat membuat dokumen XML berstruktur. Dokumen XML berstruktur seperti hieraki pohon, dengan elemen-elemen yang tersarang dalam elemen-elemen lain dan dengan suatu elemen

DATARUMAHSAKIT sebagai elemen tingkat atas atau sebagai elemen root. Dengan demikian, XML telah siap untuk mendefinisikan dokumen-dokumen berstruktur hirarki, seperti halnya suatu buku dengan bagian, bab, subbab dan seterusnya. Listing 2.1 merupakan contoh sebuah dokumen XML

```
<?xml version="1.0"?>
<!--Name File     : pasien.xml -->
<DATARUMAHSAKIT>
      <DATAPASIEN>
            <NOMORINDUKPASIEN>2008070087</NOMORINDUKPASIEN>
            <NAMAPASIEN>ANTONI YOHANES,ST</NAMAPASIEN>
            <NOMORIDENTITAS>KTP 037002355078</NOMORIDENTITAS>
            <TANGGALLAHIR>SEMARANG 21 JULI 1976</TANGGALLAHIR>
            <JENISKELAMIN>LAKI-LAKI</JENISKELAMIN>
            <GOLONGANDARAH>A</GOLONGANDARAH>
            <TINGGIDANBERATBADAN>163 / 76</TINGGIDANBERATBADAN>
            <ALAMAT>PERUM KORPRI J/37 KLIPANG SEMARANG</ALAMAT>
            <TELEPONPASIEN>024-76738878</TELEPONPASIEN>
            <PENYAKIT>TIPES</PENYAKIT>
            <MULAISAKIT>24 JUNI 2008</MULAISAKIT>
            <ALERGIOBAT>AMOXYCILIN</ALERGIOBAT>
      </DATAPASIEN>
      <DATAPASIEN>
            <NOMORINDUKPASIEN>2008070088</NOMORINDUKPASIEN>
            <NAMAPASIEN>BAMBANG SASMITO</NAMAPASIEN>
            <NOMORIDENTITAS>KTP 065447644321</NOMORIDENTITAS>
            <TANGGALLAHIR>YOGYAKARTA 24 MARET 1965</TANGGALLAHIR>
            <JENISKELAMIN>LAKI-LAKI</JENISKELAMIN>
            <GOLONGANDARAH>O</GOLONGANDARAH>
            <TINGGIDANBERATBADAN>172/ 66</TINGGIDANBERATBADAN>
            <ALAMAT>PERUM GRAHA WAHID C-34 SEMARANG</ALAMAT>
            <TELEPONPASIEN>08156686868</TELEPONPASIEN>
            <PENYAKIT>PARU-PARU</PENYAKIT>
            <MULAISAKIT>22 MEI 2008</MULAISAKIT>
            <ALERGIOBAT>AMPICILIN</ALERGIOBAT>
      </DATAPASIEN>
</DATARUMAHSAKIT>
```

Listing 2.1 Contoh Dokumen XML

2.2.1.1.1 Karakter Data dan *Markup*

Teks dapat berupa karakter data atau *markup*. *Markup* berwujud *start-tag, end-tag, empty-element tag, entity/character reference*, komentar (*comment*), *document type declaration* dan *Processing Instruction*. Semua teks yang bukan *markup* merupakan karakter data dokumen tersebut.

Dalam *content* suatu elemen, karakter data adalah setiap karakter string yang tidak berisi *start-delimiter* suatu *markup*. Dokumen XML mengenal istilah *white space* yang disimbolkan dengan huruf s dalam spesifikasinya. Penulisan dokumen XML terkadang lebih mudah mempergunakan *white space* (misal spasi dan tabulasi), agar dokumen lebih mudah dibaca.

2.2.1.1.2 Komentar (*Comment*)

Dapat muncul di manapun dalam dokumen diluar *markup* lain. Komentar bukanlah bagian dari karakter data suatu dokumen dan diberikan dengan *markup* <!--*Komentar*--> dan tidak boleh berisi *double hypens* (--) atau kurung sudut kanan (>), karena dapat menyebabkan salah interpretasi sebagai end tag. Berikut adalah contoh sebuah komentar.

<!--*Name file : Pasien.xml*-->

2.2.1.1.3 Prolog Dokumen

Dalam dokumen XML, ada beberapa deklarasi yang mungkin ada sebelum elemen yang pertama (*root element*) dalam dokumen itu. Deklarasi-deklarasi tersebut disebut *prolog*. Dokumen XML dapat, dan seharusnya dimulai dengan deklarasi XML yang mengekspresikan versi XML yang dipergunakan. Jika tidak ada deklarasinya maka dokumen tersebut tetap *well-formed*, namun tidak *valid*. Versi yang saat ini dipergunakan adalah 1.0. *Document Type Declaration* (Deklarasi Tipe Dokumen) berisi deklarasi *markup* yang memenuhi *grammar* suatu klas dokumen. *Grammar* ini disebut sebagai *Document Type Definition* (Definisi Tipe Dokumen) atau disingkat DTD. Suatu dokumen XML yang *valid* mempunyai Deklarasi Tipe Dokumen dan dokumen tersebut memenuhi aturan didalamnya. Deklarasi Tipe Dokumen harus muncul sebelum elemen pertama dalam dokumen dan dapat menunjuk langsung dalam *internal subset*, atau dapat juga keduanya.

2.2.1.2 Struktur Logikal

Setiap dokumen XML memuat satu atau lebih elemen, yaitu suatu batasan yang dibatasi oleh *start-tag* dan *end-tag* atau untuk *empty element* (elemen kosong) dibatasi oleh *empty-element-tag*. Setiap elemen mempunyai tipe, diidentifikasi dengan nama yang kadang disebut "*generic identifier*" (GI), dan boleh jadi mempunyai sekumpulan atribut. Setiap atribut mempunyai nama (*name*) dan nilai (*value*).

2.2.1.2.1 Elemen

Elemen adalah suatu komponen dokumen yang dapat berisi *content* dan dimaksudkan untuk memberi struktur suatu dokumen. Elemen ditandai dengan *start-tag* dan *end-tag*. Selain itu tag dipergunakan untuk memberikan deskripsi tentang suatu elemen.

Ada juga elemen yang tidak mempunyai isi, disebut *empty element*. Meski kosong, tidak berarti elemen ini tidak dapat berisi informasi tentang elemen. *Empty element* tetap dapat mempunyai atribut yang dituliskan dalam *empty-element tag*.

2.2.1.2.2 *Start-Tag, End-Tag*, dan *Empty-Element Tag*

Di awal setiap elemen XML yang bukan *empty element* selalu ditandai dengan *start-tag*. Jika elemen tersebut mempunyai atribut, maka atribut dan nilainya ditulis dalam *start-tag* tersebut.

Start-tag ditulis dengan tanda < diikuti nama elemen tersebut, dan jika ada, dapat diikuti dengan *white space* atau atribut serta ditutup dengan tanda >. Jika memuat atribut maka nama atribut dalam satu *star-tag* tidak boleh ada yang sama. Nilai suatu atribut harus dalam tanda petik tunggal ('...') atau petik ganda ("..."). dan tidak boleh berisi tanda <.

Contoh start-tag yang mempunyai atribut :

<Sepatu Warna="merah" Ukuran="41">

Pada akhir tiap elemen yang dimulai dengan sebuah *start-tag* harus diakhiri dengan sebuah *end-tag* yang dimulai dengan tanda </ dan diikuti nama yang sama dengan tipe elemen yang diberikan pada *start-tag* serta diakhiri dengan tanda >. Perlu diperhatikan dalam penulisan nama pada *end-tag* ini bahwa XML

case-sensutive sehingga harus benar-benar sama dengan penulisan nama di *start-tag*.

Contoh end-tag : *</Sepatu>*

Teks diantara *start-tag* dan *end-tag* disebut sebagai *content* (isi) elemen. *Content* dapat berupa elemen, karakter data, *reference* (baik *entity* maupun *character reference*), ataupun komentar.

Jika suatu elemen tidak mempunyai *content* maka disebut sebagai *empty element*. Elemen tersebut harus direpresentasikan baik dengan cara menulis *start-tag* yang kemudian diikuti dengan *end-tag* maupun dengan *empty-elemen tag*.

Contoh *empty-element tag*: *<Kosong?>*

Yang berarti sama dengan penulisan *<Kosong></Kosong>*

2.2.1.3 Atribut

Atribut merupakan properti suatu elemen atau informasi tentang elemen tersebut. Atribut dalam XML dituliskan dalam bentuk *name atribut = nilai_atribut*. Nilai atribut haruslah di dalam tanda petik dua, misal "*nilai*" atau petik tunggal, misal *'nilai'*. Atribut hanya boleh muncul dalam *start-tag* dan *empty-elemet tag*.

2.2.2 *Database* XML Apache Xindice

Data XML sering disimpan sebagai file data dalam sebuah *filesystem* atau sebagai teks atau karakter dalam *relational databases*. Namun, data XML dalam ukuran yang besar akan sulit dalam pengolahan datanya. *Relational Databases* tidak cocok untuk pengolahan data-data XML yang mempunyai struktur tertentu. Penggunaan teks dalam sebuah *relational databases* dapat dengan mudah disimpan sebagai sebuah data, tapi akan kesulitan ketika memberikan perintah atau *query* terhadap struktur data XML yang telah disimpan. *Database* yang mampu mengolah dan bisa mengakomodasi data-data XML adalah XML *Databases* Xindice, dimana database ini dibuat untuk pengembangan dokumen XML.

2.2.2.1. Manfaat Xindice

Apache Xindice adalah server basis data yang didesain dari awal untuk menyimpan data XML, atau yang saat ini lebih dikenal dengan *native* XML. Xindice dibaca dengan bahasa Italy, zeen-dee-chay. Xindice diawali dari kode proyek yang bernama dbXML Core, yang kemudian pada bulan Desember tahun 2001 kode sumber aplikasi tersebut didonasikan kepada yayasan Apache Software Foundation.

Keuntungan utama penggunaan basis data native XML, adalah jika data yang diolah aplikasi merupakan data XML. Karena dengan dipergunakannya server basis data native XML, aplikasi tidak perlu melakukan pemetaan dari data XML ke struktur data tertentu. Aplikasi dapat langsung menyimpan data XML, dan mendapatkan kembali data XML dalam format yang tidak memerlukan perubahan.

Pada saat ini, Xindice menggunakan perintah XPath untuk bahasa query-nya dan XML:DB Update untuk bahasa update-nya. Selain itu, fungsionalitas Xindice juga bisa diakses melalui aplikasi lain, tidak hanya Java, dengan menggunakan API XML-RPC, yang merupakan salah satu mekanisme komunikasi antar aplikasi menggunakan protokol HTTP.

2.2.2.2. Fitur Xindice

Sebagai database native XML, Xindice memiliki fitur-fitur yang memudahkan bekerja dengan data XML. Fitur-fitur ini dirancang agar aplikasi dapat bekerja dengan data XML secara fleksibel. Beberapa fitur tersebut di antaranya adalah:

1. Koleksi dokumen

   Dokumen-dokumen XML disimpan di dalam koleksi yang dapat menerima perintah *query* secara keseluruhan. Aplikasi dapat membuat satu koleksi untuk satu jenis dokumen, atau satu koleksi untuk banyak jenis dokumen. Xindice mendukung penuh pilihan yang manapun yang dipilih.

2. Mesin *query XPath*

   Untuk melakukan *query* atau pencarian pada dokumen, dipergunakan sintaksis XPath sebagaimana yang didefinisikan oleh W3C (*World Wide*

Web Consortium). Fitur ini menyediakan mekanisme yang fleksibel untuk melakukan pencarian pada dokumen dengan melakukan navigasi dan membatasi output dari *tree* yang dihasilkan.

3. *Indexing* XML

   Untuk mempercepat kinerja pencarian, aplikasi dapat mendefinisikan indeks terhadap elemen atau atribut. Fitur ini akan meningkatkan performa pencarian secara signifikan.

4. Implementasi XML:DB XUpdate

   XUpdate adalah mekanisme penyegaran dokumen berbasis XML, yang dilakukan disisi *server*. Modifikasi dari hasil perintah XUpdate dapat diaplikasikan pada satu dokumen, atau juga pada keseluruhan dokumen.

5. Implementasi API XML:DB

   Untuk pengembang aplikasi Java, Xindice mengimplementasikan dukungan penuh kepada standar pengolahan data XML yang disebut dengan XML:DB *Application Programming Interface*. Sebagaimana JDBC merupakan standar pengembangan aplikasi basis data, maka XML:DB merupakan standar pengembangan aplikasi XML.

6. Perangkat manajemen berbasis konsol

   Untuk mendukung administrasi basis data, Xindice menyediakan seperangkat alat bantu berbasis konsol, yang merupakan implementasi dari semua fungsionalitas yang dapat diprogram melalui XML:DB API. Sehingga, admin dari basis data dapat melakukan perawatan terhadap basis data tanpa perlu membuat program Java tersendiri.

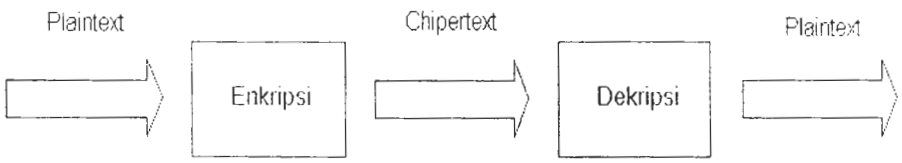
7. Arsitektur modular

   Server Xindice dibangun secara modular, sehingga sangat mudah untuk menambah atau menghapus komponen agar server sesuai dengan kebutuhan.

2.2.3 Kriptografi

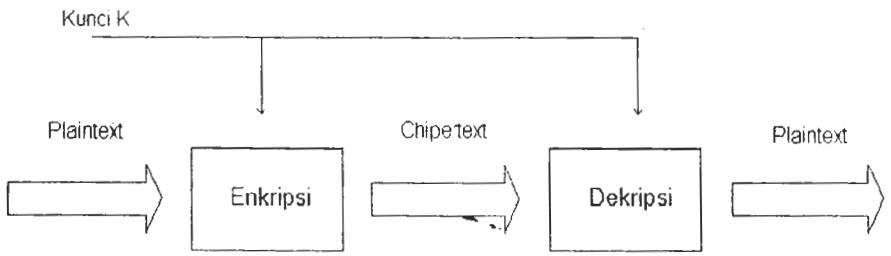
Ilmu kriptografi adalah ilmu yang mempelajari tentang penyembunyian huruf atau tulisan sehingga membuat tulisan tersebut tidak dapat dibaca oleh orang yang tidak berkepentingan. Kriptografi sudah dipakai sejak jaman Julius Caesar dimana akan mengirimkan pesan kepada panglimanya tetapi tidak mempercayai kurir pembawa pesan tersebut. Kriptografi mempunyai 2 (dua) bagian yang penting, yaitu enkripsi dan dekripsi. Enkripsi adalah proses dari penyandian pesan asli menjadi pesan yang tidak dapat diartikan seperti aslinya.

Dekripsi sendiri berarti merubah pesan yang sudah disandikan menjadi pesan aslinya. Pesan asli biasanya disebut *plaintext*, sedangkan pesan yang sudah disandikan disebut *ciphertext*. Pada Gambar 2.1 dapat dilihat bahwa masukan berupa *plaintext* akan masuk ke dalam blok enkripsi dan keluarannya akan berupa *ciphertext*, kemudian *ciphertext* akan masuk ke dalam blok dekripsi dan keluarannya akan kembali menjadi *plaintext* semula.



Gambar 2.1 Proses Enkripsi dan Dekripsi

Ada 2 (dua) model algoritma enkripsi yang menggunakan kunci, yaitu kunci simetrik dan kunci asimetrik. Enkripsi kunci simetrik yang biasanya disebut enkripsi konvensional adalah enkripsi yang menggunakan kunci yang sama untuk enkripsi maupun dekripsi, dari Gambar 2.2 terlihat bahwa untuk mengenkripsi maupun mendekripsi pesan hanya menggunakan satu buah kunci (K) saja.



Gambar 2.2 Enkripsi-dekripsi Kunci Simetrik

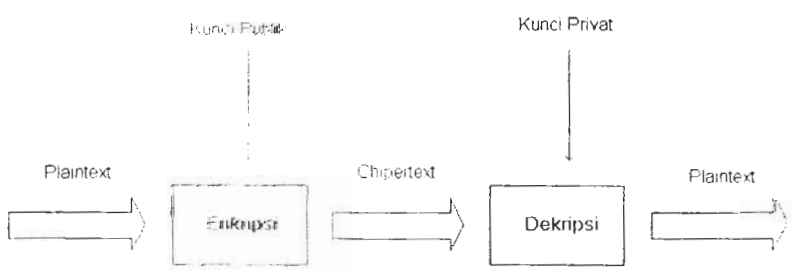
Penggunaan metode ini membutuhkan persetujuan antara pengirim dan penerima tentang kunci sebelum mereka saling mengirim pesan. Keamanan dari kunci simetrik tergantung pada kerahasiaan kunci, apabila seorang penyusup dapat menemukan kunci maka dengan mudah dapat membaca pesan yang sudah dienkripsi. Enkripsi kunci simetrik dapat dibagi kedalam 2 (dua) kelompok yaitu metode *stream cipher* dan metode *block cipher*.

Enkripsi kunci asimetrik (biasa disebut enkripsi kunci publik) dibuat sedemikian rupa sehingga kunci yang dipakai untuk enkripsi berbeda dengan kunci yang dipakai untuk dekripsi. Enkripsi kunci publik disebut demikian karena kunci untuk enkripsi boleh disebarluaskan kepada umum sedangkan kunci untuk mendekripsi hanya disimpan oleh orang yang bersangkutan. Enkripsi asimetrik dapat ditulis seperti berikut:

Ek (P) = C

Dk (C) = P

Contohnya seperti pada Gambar 2.3 bila seseorang ingin mengirim pesan kepada orang lain maka orang tersebut menggunakan kunci publik orang tersebut untuk mengenkripsi pesan yang kita kirim kepadanya lalu orang tersebut akan mendekripsi pesan tersebut dengan kunci privat miliknya.



Gambar 2.3 Enkripsi Kunci Asimetrik

2.2.3.1 Tujuan Dari Kriptografi

Seperti juga perkembangan ilmu kriptografi, tujuan-tujuan dari kriptografi teruslah berkembang. Bila pertama kali dibuat hanya untuk keamanan data saja, tetapi sekarang semakin banyak tujuan-tujuan yang ingin dicapai, yaitu:

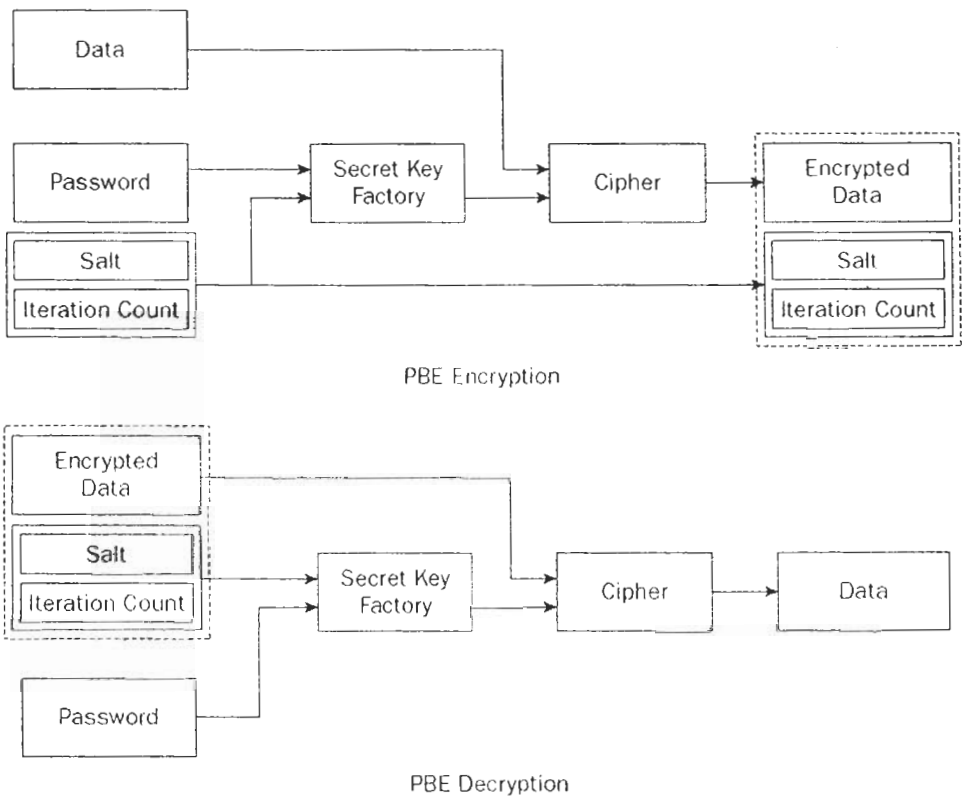
1. *Privasi*, Musuh tidak dapat membongkar tulisan yang kita kirim.
2. *Autentikasi*, Penerima pesan dapat meyakinkan dirinya bahwa pesan yang diterima tidak terjadi perubahan dan berasal dari orang yang diinginkan.
3. *Tanda tangan*, penerima pesan dapat meyakinkan pihak ketiga bahwa pesan yang diterima berasal dari orang yang diinginkan.
4. *Minimal*, Tidak ada yang dapat berkomunikasi dengan pihak lain kecuali berkomunikasi dengan pihak yang diinginkan.
5. *Pertukaran bersama,* suatu nilai (misalnya tanda tangan sebuah kontrak) tidak akan dikeluarkan sebelum nilai lainnya (misalnya tanda tangan pihak lain) diterima.
6. *Koordinasi,* di dalam komunikasi dengan banyak pihak, setiap pihak dapat berkoordinasi untuk tujuan yang sama walaupun terdapat kehadiran musuh.

2.2.3.2 *Password Based Encryption*

Password Based Encryption (PBE) adalah sebuah metode kriptografi simetrik yang menggunakan kunci seperti *password* dalam melakukan proses enkripsinya dan menggunakan kunci yang sama untuk melakukan proses dekripsinya sehingga akan dihasilkan data yang sama dengan data *plaintex* aslinya. Data *plaintex* yang telah dienkripsi akan menghasilkan sebuah *chipertex* yang tidak dapat dibaca oleh orang lain. *Chipertex* inilah yang akan dikirimkan ke pihak kedua sehingga akan memiliki kerahasiaan yang bisa diandalkan. Data *chipertex* yang dihasilkan akan berubah-ubah sesuai masukan data kunci *password* yang diberikan.

Kriptografi PBE dibuat berdasarkan mekanisme *hashing*. Sebuah *password* dan *salt* akan dikombinasikan sehingga akan menghasilkan data yang

random melalui proses fungsi aplikasi dan akan diolah oleh perhitungan iterasi (*iteration count*) sehingga ketika proses pencampuran telah selesai akan menghasilkan data berupa *chipertex*. Gambar 2.7 akan menunjukkan proses enkripsi menggunakan *Password Based Encryption* (PBE)



Gambar 2.4 Proses Enkripsi dan Dekripsi menggunakan PBE

Password, merupakan data yang selain harus dijaga kerahasiaannya juga merupakan data yang harus sulit ditebak oleh orang lain sehingga aplikasi yang dikerjakan akan menjadi sangat aman. Berapa lebar *bandwidth* yang didapatkan pada sebuah *password* dalam pemrograman java tergantung pada metode PBE yang digunakan. Pada umumnya penggunaan seperti PKC#5 hanya memperhitungkan jumlah karakter ASCII dan hanya didapatkan 8 bits pada masing-masing java karakter yang telah diproses ke dalam fungsi. Jika

menggunakan metode PKC#12 bisa didapatkan sampai 12 bits *full* untuk masing-masing karakter java.

Salt, merupakan sebuah nilai publik dan dapat dengan mudah untuk ditemukan oleh orang lain. Salt digunakan untuk menambah sebuah *string* dari *byte-byte* yang random pada *password*, *password* yang sama dapat digunakan sebagai sebuah sumber untuk nomor yang besar dari kunci-kunci yang berbeda. *Salt* yang bagus adalah *salt* yang besarnya bisa menyamai ukuran blok dari fungsi yang digunakan pada untuk memproses *password*.

Iteration Count, adalah juga merupakan sebuah nilai publik. Fungsi dari *iteratin count* adalah untuk menambah perhitungan waktu yang dibutuhkan untuk mengkonversi sebuah *password* menjadi sebuah kunci.

Beberapa algoritma yang digunakan dalam *Password Based Encryption adalah*:

1. PBEWithMD5AndDES
2. PBEWithSHA1AndDES
3. PBEWithSHA1AndRC2
4. PBEWithMD5AndRC2
5. PBEWithSHA1AndIDEA
6. PBEWithSHA1And3-KeyTripleDES
7. PBEWithSHA1And2-KeyTripleDES
8. PBEWithSHA1And40BitRC2
9. PBEWithSHA1And40BitRC4
10. PBEWithSHA1And128BitRC2
11. PBEWithSHA1And128BitRC4
12. PBEWithSHA1AndTwofish

2.2.4 PBE dengan MD5 dan DES

PBE dengan MD5 dan DES merupakan metode kriptografi menggunakan algoritma *Message Digest* 5 (MD5) dan *Data Encryption Standard* (DES). MD5 adalah algoritma *message digest* yang dikembangkan oleh Ronald Rivest pada tahun 1991. MD5 mengambil pesan dengan panjang sembarang dan menghasilkan *message digest* 128 bit. Pada MD5 pesan diproses dalam blok 512 bit dengan empat *round* berbeda.

DES, akronim dari *Data Encryption Standard*, adalah nama dari *Federal Information Processing Standard* (FIPS) 46-3, yang menggambarkan *data encryption algorithm* (DEA). DEA juga didefinisikan dalam ANSI *standard* X3.92. DEA merupakan perbaikan dari algoritma Lucifer yang dikembangkan oleh IBM pada awal tahun 70an. Meskipun algoritmanya pada intinya dirancang oleh IBM, NSA dan NBS (sekarang NIST (*National Institute of Standards and Technology*)) memainkan peranan penting pada tahap akhir pengembangan. DEA, sering disebut DES, telah dipelajari secara ekstensif sejak publikasinya dan merupakan algoritma simetris yang paling dikenal dan paling banyak digunakan.

DES memiliki ukuran blok 64-bit dan menggunakan kunci 56-bit kunci selama eksekusi (8 bit paritas dihilangkan dari kunci 64 bit). Saat digunakan untuk komunikasi, baik pengirim maupun penerima harus mengetahui kunci rahasia yang sama, yang dapat digunakan untuk mengenkrip dan mendekrip pesan, atau untuk proses *generate* dan verifikasi *message authentication code* (MAC).

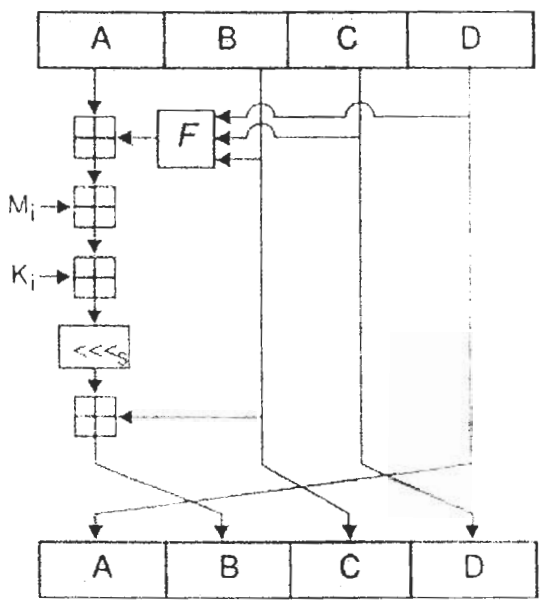
2.2.5 MD5 (*Message Digest* 5)

2.2.5.1 Prinsip Dasar MD5

Message Digest 5 (MD-5) adalah salah satu penggunaan fungsi *hash* satu arah yang paling banyak digunakan. MD-5 merupakan fungsi *hash* kelima yang dirancang oleh Ron Rivest. MD-5 merupakan pengembangan dari MD-4 dimana terjadi penambahan satu ronde. MD-5 memproses teks masukan ke dalam blok-

blok bit sebanyak 512 bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD-5 berupa 4 buah blok yang masing-masing 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai *hash*. Pada Gambar 2.5 terlihat simpul utama dari MD-5. Simpul utama MD5 mempunyai blok pesan dengan panjang 512 bit yang masuk ke dalam 4 buah ronde. Hasil keluaran dari MD-5 adalah berupa 128 bit dari *byte* terendah A dan tertinggi *byte* D.

2.2.5.2 Penjelasan Algoritma MD-5



Gambar 2.5 Operasi MD5

Gambar 2.5 merupakan operasi dari sebuah MD5. MD5 terdiri atas 64 operasi, dikelompokkan dalam empat putaran dari 16 operasi. *F* adalah fungsi nonlinear, satu fungsi digunakan pada tiap-tiap putaran. *Mi* menunjukkan blok 32-bit dari masukan pesan, dan *Ki* menunjukkan konstanta 32-bit, berbeda untuk tiap-tiap operasi. <<<s menunjukkan perputaran bit kiri oleh s, s bervariasi untuk tiap-tiap operasi. menunjukkan tambahan modulo 2^{32}.

4. Proses Pesan di dalam Blok 16 Word

Pada MD-5 juga terdapat 4 (empat) buah fungsi *nonlinear* yang masing masing digunakan pada tiap operasinya (satu fungsi untuk satu blok), yaitu:

$F(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$

$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge (\neg Z))$

$H(X,Y,Z) = X \oplus Y \oplus Z$

$I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$

($\oplus$ untuk XOR, $\wedge$ untuk AND, $\vee$ untuk OR dan $\neg$ untuk NOT).

Berikut dapat dilihat satu buah operasi dari MD-5 dengan operasi yang dipakai sebagai contoh adalah FF(a,b,c,d,Mj,s,ti) menunjukan $a = b + ((a + F(b,c,d) + Mj + ti) <<< s)$.

Bila Mj menggambarkan pesan ke-j dari sub blok (dari 0 sampai 15) dan <<<s menggambarkan bit akan digeser ke kiri sebanyak s bit, maka keempat operasi dari masing-masing ronde adalah:
FF(a,b,c,d,Mj,s,ti) menunjukan $a = b + ((a + F(b,c,d) + Mj + ti) <<< s)$
GG(a,b,c,d,Mj,s,ti) menunjukan $a = b + ((a + G(b,c,d) + Mj + ti) <<< s)$
HH(a,b,c,d,Mj,s,ti) menunjukan $a = b + ((a + H(b,c,d) + Mj + ti) <<< s)$
II(a,b,c,d,Mj,s,ti) menunjukan $a = b + ((a + I(b,c,d) + Mj + ti) <<< s)$
Konstanta ti didapat dari integer 232. abs(sin(i)), dimana i dalam radian.

5. Keluaran MD-5

Keluaran dari MD-5 adalah 128 bit dari *word* terendah A dan tertinggi *word* D masing-masing 32 bit.

2.2.6 DES (*Data Encryption Standard*)

2.2.6.1 Sejarah DES

Skema enkripsi yang paling umum digunakan saat ini adalah *Data encryption Standard* (DES). DES diadopsi pada tahun 1977 oleh *National Bureau of Standards*, atau sekarang disebut sebagai *National Institute* of *Standards and Technology* (NIST), sebagai *Federal Information Processing Standard* 46 (FIPS

PUB 46). Di dalam DES, data dienkripsi di dalam 64 bit blok dengan menggunakan 56 bit kunci. Algoritma DES mengubah 64 bit input di dalam berbagai langkah menjadi 64 bit output. Langkah yang sama, dengan kunci yang sama, digunakan untuk mendekripsi *ciphertext* yang dihasilkan.

Pada tahun 1960, IBM memulai sebuah proyek di dalam kriptografi komputer yang dipimpin oleh Horst Feistel. Proyek ini selesai pada tahu 1971 dengan pengembangan algoritma yang disebut sebagai LUCIFER [FEIS73], yang dijual kepada Lloyd's of London untuk digunakan pada sistem penyaluran uang tunai, yang juga dikembangkan oleh IBM. LUCIFER adalah blok *cipher* yang beroperasi pada 64 bit blok, dengan menggunakan ukuran kunci 128 bit. Karena hasil yang menjanjikan, IBM kemudian mengembangkan sistem ini secara komersial. Usaha ini dipimpin oleh Walter tuchman dan Carl Meyer, dan tidak hanya melibatkan IBM saja, tetapi juga konsultan luar dan nasehat yang bersifat teknikal dari NSA. Hasilnya, muncul LUCIFER versi baru yang lebih tahan terhadap *cryptoanalyst* tetapi dengan mengurangi ukuran kunci menjadi 56 bit sehingga dapat diimplementasikan pada sistem dengan prosesor tunggal.

Sementara itu, National Bureau of Standards (NBS) pada tahun 1973 mengeluarkan permintaan untuk standard chiper nasional. IBM mengirimkan hasil dari proye Tuchman-Meyer. Ini adalah algoritma terbaik yang diajukan dan diadopsi sebagai *Data Encryption Standard.*

2.2.6.2 Prinsip Kerja DES

DES bekerja dalam model bit , atau angka biner 0 dan 1. Setiap kelompok dari 4 bit membentuk *hexadesimal*, atau bilangan berbasis 16. Angka biner 0001 membentuk angka heksa 1, dan seterusnya. DES bekerja dengan mengenkripsi setiap group yang terdiri dari 64 bit data. Untuk melakukan enkripsi, DES membutuhkan kunci yang juga mempunyai ukuran 64 bit. namun dalam prakteknya bit ke 8 dari setiap kelompok 8 bit diabaikan, sehingga ukuran kunci menjadi 56 bit. Sebagai contoh, jika kita ingin mengenkripsi pesan "8787878787878787" dengan kunci "0E329232EA6D0D73", maka akan dihasilkan *ciphertext* "0000000000000000". Jika *ciphertext* tersebut didekripsi dengan menggunakan kunci yang sama, maka outputnya adalah pesan asli.

PUB 46). Di dalam DES, data dienkripsi di dalam 64 bit blok dengan menggunakan 56 bit kunci. Algoritma DES mengubah 64 bit input di dalam berbagai langkah menjadi 64 bit output. Langkah yang sama, dengan kunci yang sama, digunakan untuk mendekripsi *ciphertext* yang dihasilkan.

Pada tahun 1960, IBM memulai sebuah proyek di dalam kriptografi komputer yang dipimpin oleh Horst Feistel. Proyek ini selesai pada tahu 1971 dengan pengembangan algoritma yang disebut sebagai LUCIFER [FEIS73], yang dijual kepada Lloyd's of London untuk digunakan pada sistem penyaluran uang tunai, yang juga dikembangkan oleh IBM. LUCIFER adalah blok *cipher* yang beroperasi pada 64 bit blok, dengan menggunakan ukuran kunci 128 bit. Karena hasil yang menjanjikan, IBM kemudian mengembangkan sistem ini secara komersial. Usaha ini dipimpin oleh Walter tuchman dan Carl Meyer, dan tidak hanya melibatkan IBM saja, tetapi juga konsultan luar dan nasehat yang bersifat teknikal dari NSA. Hasilnya, muncul LUCIFER versi baru yang lebih tahan terhadap *cryptoanalyst* tetapi dengan mengurangi ukuran kunci menjadi 56 bit sehingga dapat diimplementasikan pada sistem dengan prosesor tunggal.

Sementara itu, National Bureau of Standards (NBS) pada tahun 1973 mengeluarkan permintaan untuk standard chiper nasional. IBM mengirimkan hasil dari proye Tuchman-Meyer. Ini adalah algoritma terbaik yang diajukan dan diadopsi sebagai *Data Encryption Standard.*

2.2.6.2 Prinsip Kerja DES

DES bekerja dalam model bit , atau angka biner 0 dan 1. Setiap kelompok dari 4 bit membentuk *hexadesimal*, atau bilangan berbasis 16. Angka biner 0001 membentuk angka heksa 1, dan seterusnya. DES bekerja dengan mengenkripsi setiap group yang terdiri dari 64 bit data. Untuk melakukan enkripsi, DES membutuhkan kunci yang juga mempunyai ukuran 64 bit. namun dalam prakteknya bit ke 8 dari setiap kelompok 8 bit diabaikan. sehingga ukuran kunci menjadi 56 bit. Sebagai contoh, jika kita ingin mengenkripsi pesan "8787878787878787" dengan kunci "0E329232EA6D0D73", maka akan dihasilkan *ciphertext* "0000000000000000". Jika *ciphertext* tersebut didekripsi dengan menggunakan kunci yang sama, maka outputnya adalah pesan asli.

DES adalah sebuah "*block cipher*", artinya, DES bekerja dalam *plaintext* dengan ukuran yang telah diberikan (64 bit) dan mengembalikan *ciphertext* dengan ukuran yang sama pula.

BAB III
CARA PENELITIAN

3.1 Alat dan Bahan Penelitian

3.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam membuat aplikasi pada penelitian ini adalah sebuah Laptop dengan spesifikasi Intel(R) Celeron(R) M CPU 430 @1,73 GHz, 1,49 GB of RAM, dan Harddisk 80GB.

3.1.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam membangun aplikasi Penggunaan XML *Database* Xindice pada Aplikasi Kriptografi menggunakan Data XML untuk Keamanan Distribusi Data adalah sebagai berikut.

1. *XML Database* Xindice sebagai basis data yang mengakomodasi data-data XML.
2. NetBeans IDE 6.0 dan Macromedia Dream Weaver 8 untuk membuat *prototype* program aplikasi kriptografi pada Web
3. JSP (*Java Server Page*) sebagai perangkat lunak pembantu untuk editor program
4. *Browser* Mozilla Firefox untuk menampilkan aplikasi program.
5. Microsoft Windows XP Professional sebagai sistem operasi yang digunakan untuk membangun sistem.

3.2. Subyek Penelitian

Yang menjadi subyek dalam penelitian ini adalah sistem distribusi data rekam medis pasien di sebuah contoh rumah sakit yang terdapat beberapa bagian penting yang memiliki kewenangan dan hak masing-masing untuk mengakses data dan untuk mengetahui mengenai informasi dari data yang dibutuhkan.

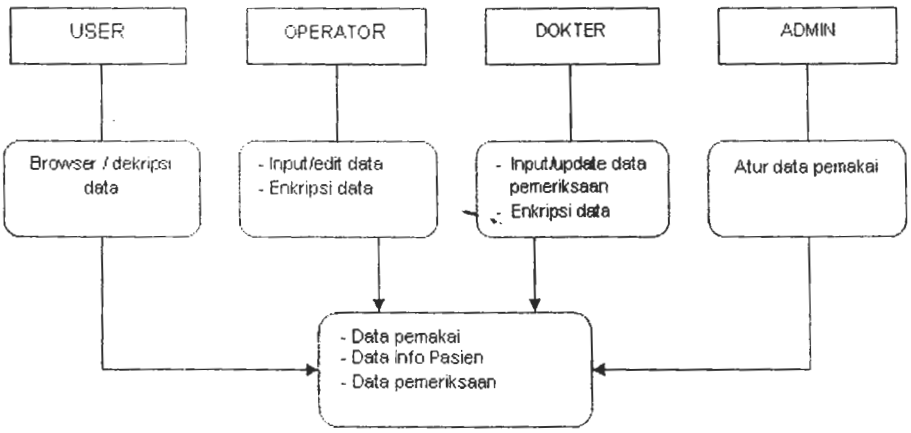
3.3. Spesifikasi Sistem

Aplikasi yang akan dibangun meliputi spesifikasi sistem sebagai berikut :

1. Aplikasi berupa pembuatan sebuah *prototype* web dari sebuah contoh rumah sakit yang dapat ditampilkan dalam sebuah *browser* yang berisi tentang informasi mengenai rumah sakit dan data – data yang bisa diakses oleh siapapun.
2. Aplikasi dilengkapi dengan informasi data rekam medis pasien rumah sakit yang diatur dengan sebuah model sistem sederhana. Sistem yang dirancang memiliki beberapa bagian yang terdiri dari administrator, dokter, operator dan *user* (pasien) dimana masing-masing bagian memiliki hak sendiri-sendiri sesuai dengan distribusi password yang telah disepakati untuk memasukkan data, menambah data, mengenkripsi dan mendekripsi data, dengan tujuan agar data dapat disampaikan kepada pihak yang berhak dan berwenang untuk mengolah dan mengakses informasi data.
3. Aplikasi dibuat dengan menambahkan sebuah basis data menggunakan XML *database* Xindice. Xindice merupakan sebuah *database* yang dapat mengakomodasi data-data XML yang dikembangkan dalam penelitian ini. Dibandingkan dengan RDBMS (*Relational Database Management System*) penggunaan *XML Database* dapat lebih memudahkan dalam proses aplikasi dan pengembangan data XML.
4. Aplikasi yang dibuat menggunakan proses enkripsi dan dekripsi dengan program java dengan metode *Password Based Encryption* (PBE) yang telah disediakan dalam JCE (*Java Cryptography Extension*). JCE merupakan sebuah ekstensi kriptografi yang dikembangkan oleh Java yang memiliki beberapa pilihan metode kriptografi yang dapat digunakan secara bebas untuk pengembangan kriptografi.

3.4. Perancangan Desain Sistem

Tahap perancangan desain sistem merupakan tahap penggambaran atau identifikasi komponen-komponen fungsional yang digunakan dalam perencanaan pengembangan sistem. Tahap perancangan sistem ini bertujuan untuk mendesain sistem yang lengkap dan jelas yang akan digunakan dalam implementasi yang ditujukan untuk memenuhi kebutuhan pemakai sistem.



Gambar 3.1 Rancangan Desain Sistem

Gambar 3.1 diatas merupakan rancangan desain dari sistem kriptografi PBE yang digunakan untuk keamanan distribusi data pada aplikasi web sebuah rumah sakit. Informasi pada sistem aplikasi dapat dijelaskan secara umum yang terdiri dari beberapa bagian entitas yaitu :

1. *Administrator*

   Administrator terlebih dahulu harus melakukan *login*, kemudian sistem akan memberikan konfirmasi login. Setelah login diterima, *administrator* dapat melakukan perubahan pada data pemakai dimana administrator mengatur administrasi perubahan dan penambahan data pemakai. Data pemakai dalam sistem ini adalah dokter dan *operator* yang masing-masing mempunyai kunci *password* masing-masing dalam membuka akses dalam sistem.

2. *Operator*

   Operator melakukan *login* terlebih dahulu, kemudian setelah mendapat hak akses maka operator mempunyai hak untuk mengisi data-data pasien, menambah dan menghapus data pasien, serta melakukan proses enkripsi dan dekripsi menggunakan kunci *password* yang sama dengan yang diberikan kepada tiap-tiap user (pasien), kemudian menyimpan data pasien untuk dipergunakan oleh dokter dalam penambahan data rekam medis, dan untuk diakses oleh *user* (pasien).

3. Dokter

Dokter juga melakukan *login* terlebih dahulu, kemudian setelah mendapat hak akses maka dokter dapat membuka data pasien sesuai dengan nomor identitas yang telah disepakati dengan operator dan *user*. Kemudian dokter juga dapat menambahkan data tambahan pemeriksaan yang bertujuan untuk memberikan *update* data rekam medis bagi seorang pasien. Data rekam medis ini merupakan data riwayat penyakit dan kesehatan pasien yang hanya dapat dibaca oleh si pasien melalui proses dekripsi. Oleh dokter data rekam medis ini kemudian dilakukan proses enkripsi dan disimpan untuk nantinya dapat diakses oleh pihak pasien. Proses enkripsi menggunakan *password* nomor identitas masing-masing pasien untuk memudahkan pasien pada saat membuka dan mengakses data riwayat rekam medis dirinya.

4. *User* (pasien)

Pasien merupakan entitas terakhir yang dapat membuka akses untuk mengetahui data-data rekam medis dirinya dengan cara memasukkan *login password* berupa nomor identitas pasien yang telah disepakati dengan operator. Pasien dapat mengetahui data informasi tentang dirinya dan dapat melakukan proses dekripsi menggunakan *password* yang telah diberikan untuk membuka data rekam medis yang terenkripsi.

3.5. Perancangan *Data Flow Diagram* (DFD)

Untuk menggambarkan informasi yang mengalir pada sistem atau aplikasi dapat digunakan DFD *(Data Flow Diagram /* Diagram Alir Data*)*. DFD juga dapat digunakan untuk menggambarkan sistem dan memberikan pandangan umum pada sistem untuk memperlihatkan proses interaksi antar bagian.

Data flow diagram juga dapat digunakan untuk menggambarkan sistem pada setiap tingkatan. Keuntungan dari penggunaan DFD adalah dapat menggambarkan sistem dari level yang paling tinggi ke level yang lebih rendah. Langkah awal dengan membuat diagram konteks atau DFD level 0 sebagai gambaran sistem secara keseluruhan.

3.5.1. DFD Level 0

Pada DFD level 0 (*diagram konteks*) menunjukkan rancangan proses pada suatu proses dasar dari sistem. Pada DFD level 0 ini terdapat empat entitas yaitu *user*, *operator*, dokter dan *admin*.

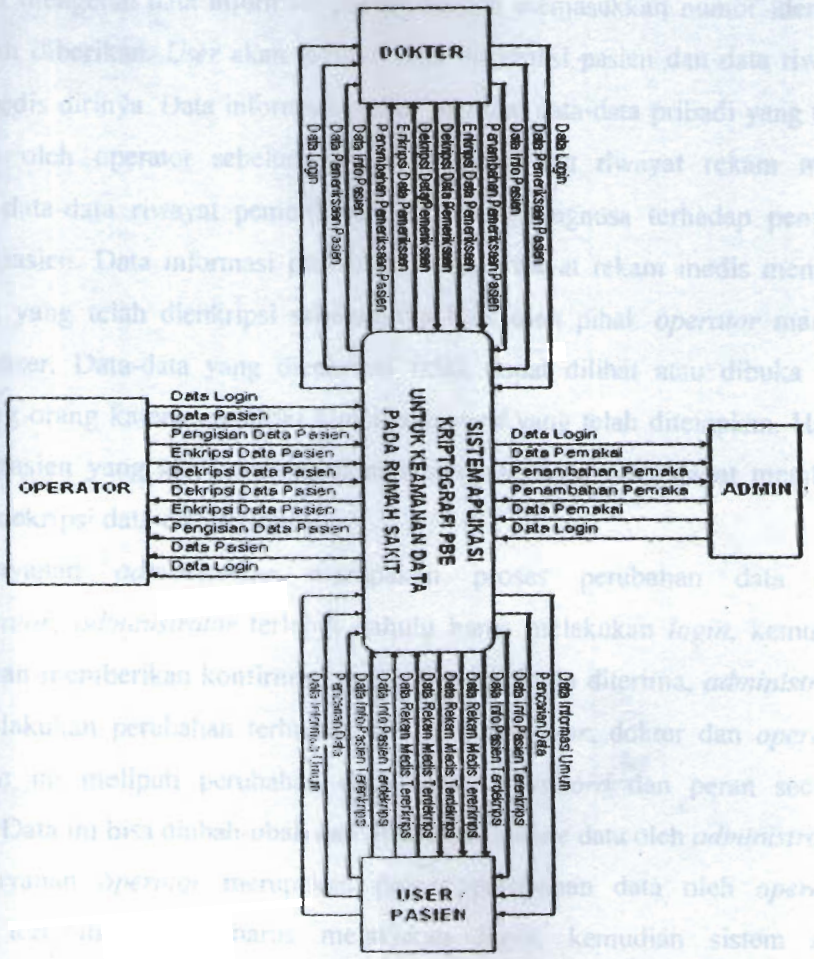
User dapat meminta data atau informasi yang diinginkan dengan memilih menu homepage pada sistem aplikasi, maka *user* akan mendapatkan informasi umum tentang rumah sakit dan informasi pencarian data dengan cara memasukkan identitas *user* (pasien). Selain itu *user* juga dapat memilih menu aksi sehingga *user* dapat melihat data informasi pasien dan data riwayat rekam medisnya, tetapi *user* memerlukan kunci *password* untuk mendekrip data informasi pasien dan data riwayat rekam medis. Untuk membuka data informasi pasien *user* membutuhkan kunci *password* yang telah dibuat oleh *operator*, dan untuk membuka data riwayat rekam medis user membutuhkan kunci *password* yang telah dibuat oleh dokter yaitu berupa nomor identitas pasien. Jadi dalam sistem ini terlebih dahulu harus dibuat kesepakatan kunci *password* antara *user* dengan operator dan *user* dengan dokter.

Operator harus melakukan login terlebih dahulu, kemudian sistem akan memberikan konfirmasi *login*. Setelah *login* diterima, *operator* dapat memasukkan data-data tentang informasi pasien sesuai dengan nomor identitas pasien, menambahkan data baru, meng-*update* data, menghapus data, mengenkripsi data yang diperlukan, kemudian menyimpan data-data tersebut untuk nantinya digunakan oleh dokter untuk penambahan data riwayat rekam medis pada data pasien. Dalam mengenkripsi data yang diperlukan, *operator* menggunakan kunci *password* tertentu yang telah diberikan atau disepakati dengan *user* untuk nantinya digunakan oleh *user* dalam mendekripsi data yang telah dienkripsi oleh *operator*.

Dokter juga harus melakukan *login* terlebih dahulu, setelah konfirmasi *login* diterima maka dokter dapat mengakses data informasi pasien dengan memasukkan nomor identitas pasien pada menu pemeriksaan pasien. Pada menu pemeriksaan pasien dokter dapat melihat data informasi pasien yang telah diolah oleh *operator* berupa data-data pribadi pasien dimana terdapat data yang masih

utuh maupun data yang telah dienkripsi oleh *operator*, sehingga dokter tidak akan bisa mengetahui data yang dienkripsi tersebut karena dokter tidak memiliki hak akses maupun kunci *password* untuk membukanya. Kemudian dokter dapat mengisi data pemeriksaan pasien, menambah data pemeriksaan, menghapus data pemeriksaan maupun meng-*update* data pemeriksaan pasien. Untuk yang terakhir dokter dapat mengenkripsi data riwayat pemeriksaan pasien dan kemudian kemudian menyimpannya. Pada proses enkripsi data riwayat pemeriksaan ini dokter menggunakan kunci *password* berupa nomor identitas pasien, sehingga akan memudahkan *user* (pasien) untuk mendekripsi data riwayat pemeriksaan yang dibutuhkan.

Sedangkan *admin* terlebih dahulu harus melakukan *login*, kemudian sistem akan memberikan konfirmasi *login*. Setelah *login* diterima, admin dapat menambah data pemakai sistem yang terdiri dari *user*, *operator* dan *admin*. *Admin* dapat meng-*edit* data pemakai yang diperlukan, menghapus data pemakai yang tidak diperlukan dan meng-*update* data pemakai. Pengisian data dilakukan dengan memasukkan nama pemakai, kunci *password* yang digunakan para pemakai dan pendeskripsian peran masing-masing pemakai apakah sebagai *admin*, *operator* atau *user*. Proses dari seluruh sistem secara garis besar dapat ditunjukkan pada Gambar 3.2 di bawah ini:



Gambar 3.2 Data Flow Diagram level 0

3.5.2. DFD Level 1

Pada DFD level 1 merupakan penjelasan secara lebih rinci dari proses yang terjadi pada DFD level 0. Pada DFD level 1 ini terdapat 3 buah tabel yaitu tabel data pemakai, tabel data informasi pasien dan tabel data pemeriksaan penyakit. Proses yang terjadi pada DFD level 1 ini adalah proses layanan *user*, layanan *operator*, layanan dokter dan layanan admin. *User* dapat memperoleh

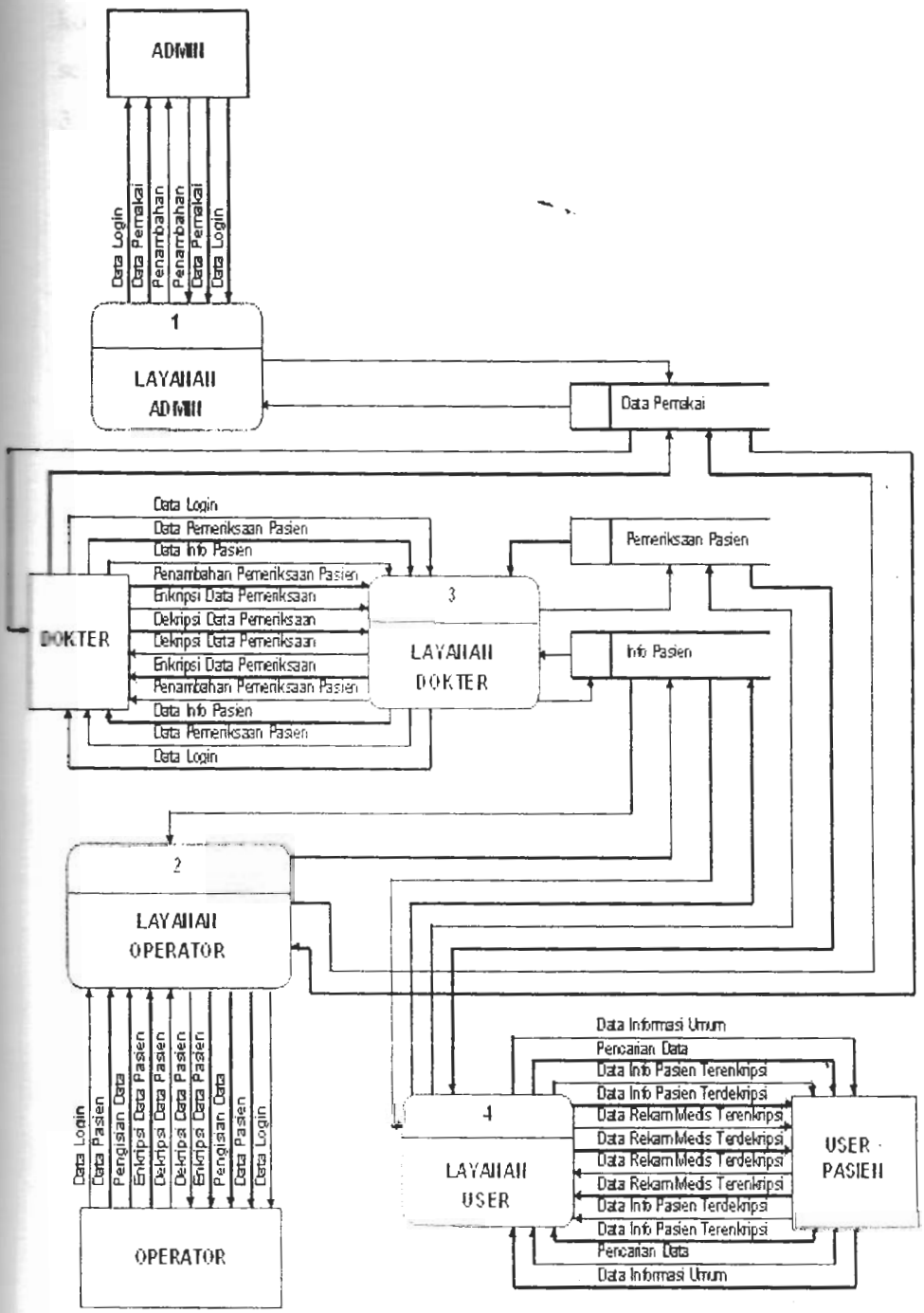
informasi mengenai data informasi pasien dengan memasukkan nomor identitas yang telah diberikan. *User* akan melihat data informasi pasien dan data riwayat rekam medis dirinya. Data informasi pasien memuat data-data pribadi yang telah disimpan oleh operator sebelumnya. Sedangkan data riwayat rekam medis memuat data-data riwayat pemeriksaan dan hasil diagnosa terhadap penyakit seorang pasien. Data informasi pasien dan data riwayat rekam medis memiliki data-data yang telah dienkripsi sebelumnya baik oleh pihak *operator* maupun pihak dokter. Data-data yang dienkripsi tidak dapat dilihat atau dibuka oleh sembarang orang karena memiliki kunci *password* yang telah ditetapkan. Hanya seorang pasien yang telah mendapatkan kunci *password* yang dapat membuka atau mendekripsi data-data tersebut.

Layanan *administrator* merupakan proses perubahan data oleh *administrator*, *administrator* terlebih dahulu harus melakukan *login*, kemudian sistem akan memberikan konfirmasi *login*. Setelah login diterima, *administrator* dapat melakukan perubahan terhadap data *administrator*, dokter dan *operator*. Perubahan ini meliputi perubahan data nama, *password* dan peran seorang pemakai. Data ini bisa diubah-ubah dan dilakukan *update* data oleh *administrator*.

Layanan *operator* merupakan proses perubahan data oleh *operator*, operator terlebih dahulu harus melakukan *login*, kemudian sistem akan memberikan konfirmasi *login*. Setelah login diterima, *operator* dapat melakukan penambahan, penghapusan dan *update* terhadap data-data umum pasien. Setiap data yang dimasukkan oleh *operator* dapat dilakukan proses enkripsi terhadap masing-masing sub data yang dibutuhkan. Kemudian data ini disimpan untuk diumpankan pada proses pengisian dan perubahan data riwayat rekam medis oleh pihak dokter.

Layanan dokter merupakan proses perubahan data oleh dokter. Pada proses ini dokter juga harus melakukan *login* terlebih dahulu untuk dapat melanjutkan proses selanjutnya. Setelah *login* diterima, dokter dapat melanjutkan proses dengan memasukkan nomor identitas pasien yang diperiksa. Kemudian dokter akan melihat data informasi umum pasien yang telah diproses oleh operator. Data informasi umum pasien ini juga memiliki data-data yang berupa

chipertex yang merupakan data yang telah mengalami proses enkripsi pada pihak *operator*. Disini pihak dokter tidak memiliki kewenangan untuk membuka atau mendekripsi data tersebut, karena dokter tidak memiliki kunci *password* sebagai kunci untuk mendekripsinya. Proses dari DFD level 1 dapat ditunjukkan pada Gambar 3.3. sebagai berikut.



Gambar 3.3 Data Flow Diagram level 1

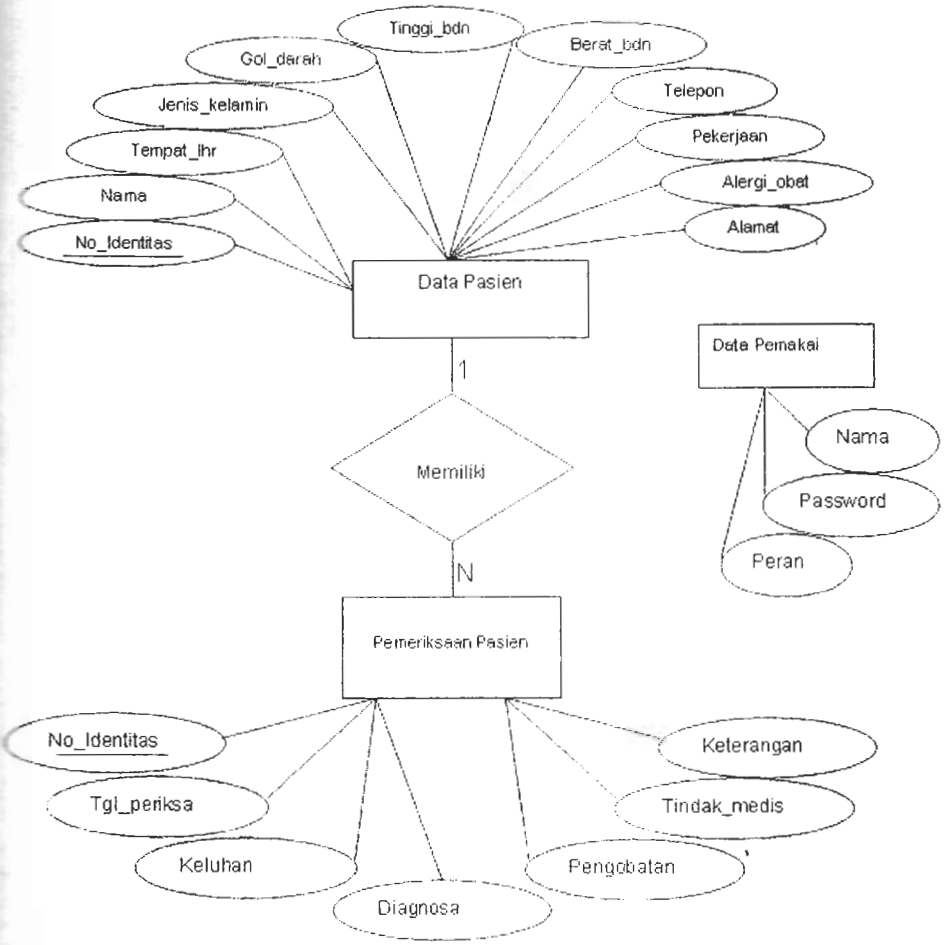
3.6 Perancangan Basisdata

Untuk mengatur sistem penyimpanan dan pengambilan data dalam sistem aplikasi ini dibutuhkan suatu basis data. Basis data merupakan salah satu

komponen yang penting dalam sistem aplikasi ini, karena basis data berfungsi sebagai penyedia informasi bagi para pemakainya.

3.6.1. *Entity Relationship Diagram (ERD)*

Hubungan antar entitas dalam sistem aplikasi ini dapat dilihat pada Gambar 3.4 di bawah ini:



Gambar 3.4 *Entity Relationship Diagram (ERD)*

3.6.2. Basis Data

Desain basis data yang akan dirancang untuk menampung data-data berisi tabel-tabel sebagai berikut:

1. Tabel Data Pasien

Tabel Data Pasien terdiri 11 field yaitu nomoridentitas sebagai *primary key*, nama, tempatlahir, jeniskelamin, golongandarah, tinggi, berat, telepon,

pekerjaan, alergiobat dan alamat. Struktur tabel Data Pasien dapat dilihat pada Tabel 3.1 di bawah ini :

Tabel 3.1 Sruktur tabel Data Pasien

Field	Type	Keterangan
nomoridentitas	String	Nomor identitas
nama	String	Nama
tempatlahir	String	Tempat lahir
jeniskelamin	String	Jenis kelamin
golongandarah	String	Golongan darah
tinggi	String	Tinggi badan
berat	String	Berat badan
telepon	String	Telepon
pekerjaan	String	Pekerjaan
alergiobat	String	Alergi obat
alamat	String	Alamat

2. Tabel Pemeriksaan Pasien

Tabel Pemeriksaan Pasien terdiri 7 field yaitu nomoridentitas sebagai *primary key*, tanggalperiksa, keluhan, diagnosa, pengobatan, tindakanmedis dan keterangan. Struktur tabel Pemeriksaan Pasien dapat dilihat pada Tabel 3.2 di bawah ini :

Tabel 3.2 Tabel Pemeriksaan Pasien

Field	Type	Keterangan
nomoridentitas	String	Nomor identitas
tanggalperiksa	String	Tanggal periksa
keluhan	String	Keluhan
diagnosa	String	Diagnosa
pengobatan	String	Pengobatan
tindakanmedis	String	Tindakan medis
keterangan	String	Keterangan

3. Tabel Data Pemakai

Tabel Data Pemakai terdiri 3 field yaitu nama, *password* dan peran. Struktur tabel Data Pemakai dapat dilihat pada Tabel 3.3 di bawah ini :

Tabel 3.3 Tabel Data Pemakai

Field	Type	Keterangan
nama	String	Nama
password	String	Password
peran	String	Peran

3.6.3. Perancangan menggunakan XML database Xindice

XML database Xindice merupakan sebuah *server* yang didesain untuk menyimpan data-data XML yang disebut dengan *collections*. *Collections* hampir mirip dengan tabel – tabel pada *Relational Database Management Systems* (RDBMS). Dalam xindice data disimpan dalam sebuah basis data dan data tersebut digunakan sebagai sebuah *document collection*. Dalam sebuah *database* terdiri dari beberapa *child collection*. Dalam xindice sebuah database dipanggil dengan nama 'db' dan seluruh *collection* akan dimulai dengan /db. Contoh sebuah *collection* seperti di bawah ini:

```
/db/my-collection/my-child-collection
```

Untuk menambahkan sebuah data dokumen pada *collection* dapat dilihat seperti di bawah ini:

```
/db/my-collection/my-child-collection/my-document
```

Untuk membuat dan mengolah data – data dokumen pada Xindice dapat dilakukan pada *command line program*.

3.6.3.1. Membuat sebuah Collection

Membuat atau menambah sebuah *collection* baru ke *database* dapat dilakukan dengan menuliskan program seperti di bawah ini:

```
xindice add_collection -c (or context) -n (or name)
```

-c adalah perintah untuk menentukan sebuah collection yang baru, -n adalah perintah untuk memberikan nama pada sebuah collection yang akan dibuat. Perintah di bawah ini adalah perintah untuk membuat atau menambahkan sebuah collection yang baru pada sebuah *database* Xindice:

```
xindice add_collection -c /db -n EdyWinarno
```

Untuk membuat *subcollection* atau data di bawah *collection* utama dapat ditulis seperti perintah di bawah ini:

```
xindice add_collection -c /db/EdyWinarno -n Penelitian
```

3.6.3.2. Menghapus sebuah Collection

Untuk menghapus sebuah data *collection* atau *subcollection* yang telah disimpan pada *database* dapat dilakukan dengan menggunakan perintah:

```
xindice delete_collection -c (or context) -n (or name)
```

Di bawah ini adalah sebuah contoh penghapusan data sebuah *collection* yang telah disimpan dalam *database*.

```
xindice delete_collection -c /db -n EdyWinarno
```

Untuk menghapus *subcollection* saja dapat dilakukan dengan menggunakan perintah seperti di bawah ini :

```
xindice delete_collection -c /db/EdyWinarno -n Penelitian
```

3.6.3.3. Memanggil daftar Collection (List Collection)

Untuk memanggil seluruh daftar collection yang telah disimpan dalam database dapat dilakukan dengan menggunakan perintah:

```
xindice list_collections -c (or context)
```

Di bawah ini adalah sebuah perintah untuk memanggil seluruh daftar collection yang telah disimpan pada database:

```
xindice list_collections -c /db
```

Untuk memanggil seluruh daftar *subcollection* pada sebuah *collection* utama dapat dilakukan dengan perintah seperti di bawah ini:

```
xindice list_collections -c /db EdyWinarno
```

3.6.3.4 Membuat sebuah dokumen pada collection

Untuk menambahkan sebuah dokumen edy.xml yang akan ditambahkan pada collection /db/EdyWinarno/Penelitian dapat dituliskan sebagai berikut:

```
xindice add_document -c /db/EdyWinarno/Penelitian -f
edy.xml
```

3.6.3.5. Pemanggilan sebuah dokumen pada sebuah collection

Dokumen yang telah disimpan pada Xindice dapat dipanggil kembali menggunakan perintah seperti di bawah ini:

```
xindice retrieve_document -c /db/EdyWinarno/Penelitian -f
edy.xml
```

3.6.3.6. Menghapus sebuah dokumen pada sebuah collection

Untuk menghapus data edy.xml yang telah disimpan pada Xindice dapat dihapus menggunakan perintah:

```
xindice delete_document -c /db/EdyWinarno/Penelitian -f
edy.xml
```

3.7. Enkripsi dan Dekripsi Data XML

Pada sistem aplikasi ini menggunakan proses enkripsi dan dekripsi dengan metode PBE (*Password Based Encryption*) with MD5 and DES. Enkripsi dan dekripsi dilakukan pada struktur data base XML yang telah dibuat, yaitu dengan cara mengenkripsi dan mendekripsi tag-tag XML yang diperlukan. Metode PBE *with* MD5 *and* DES adalah metode kriptografi simetrik yang telah disediakan pada JCE (*Java Cryptography Extension*) sebagai sebuah metode enkripsi dan dekripsi yang aplikatif terhadap pemrograman java.

Penulisan program menggunakan metode PBE with MD5 and DES dapat dilihat pada Listing 3.1 di bawah ini :

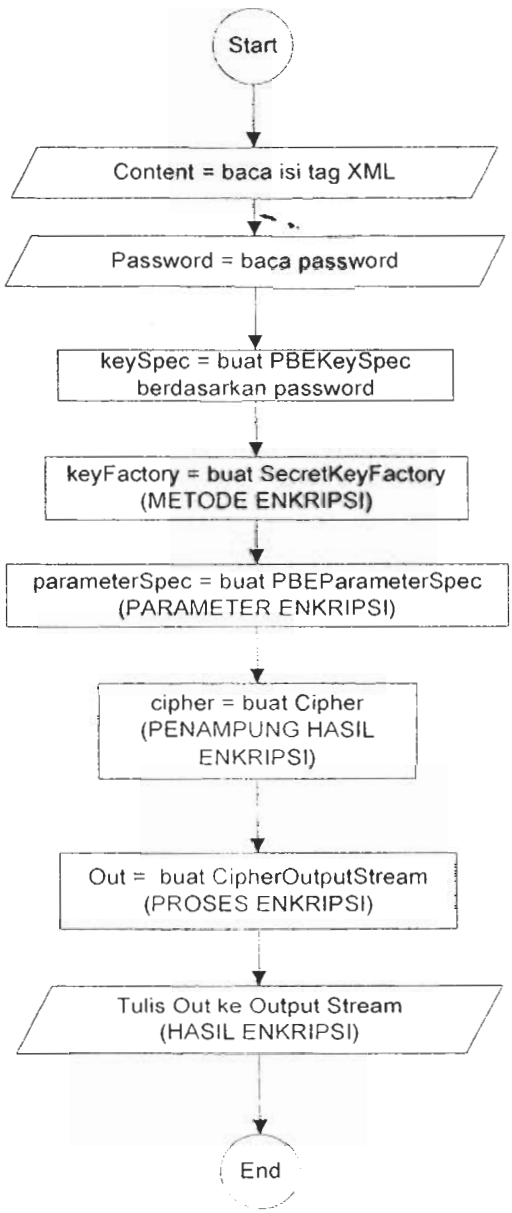
```
private static String METHOD = "PBEWithMD5AndDES";
    private static final byte[] salt = {
    (byte) 0xf5, (byte) 0x33, (byte) 0x01, (byte) 0x2a,
    (byte) 0xb2, (byte) 0xcc, (byte) 0xe4, (byte) 0x7f
    };
    private static int iterationCount = 100;
    private static Cipher cipher;
```

```
        private static byte[] outputArray;
        private static ByteArrayOutputStream cryptedText;
        private static ByteArrayInputStream decryptedText;
        private static String hasilEkripsi = "";
private static String encoding = "ISO-8859-1";//
```

Listing 3.1 Enkripsi dan Dekripsi Metode PBE with MD5 and DES

3.7.1. Enkripsi

Tag XML yang telah dipilih kemudian akan dienkripsi menggunakan metode PBE with MD5 and DES untuk menghasilkan data *chipertex*. Data *chipertex* merupakan data hasil pengolahan enkripsi dari sebuah data *plaintex* yang merupakan tampilan sebuah data dari sebuah tag XML. Pada proses ini sebuah data pada tag XML akan diubah dan dikombinasikan dengan kunci *password* yang dimasukkan sehingga akan menghasilkan tampilan data *chipertex* yang merupakan tampilan karakter acak yang tidak dapat dibaca. Penulisan metode enkripsi pada *flowchart* dan program dapat dilihat pada Gambar 3.5 di bawah ini:

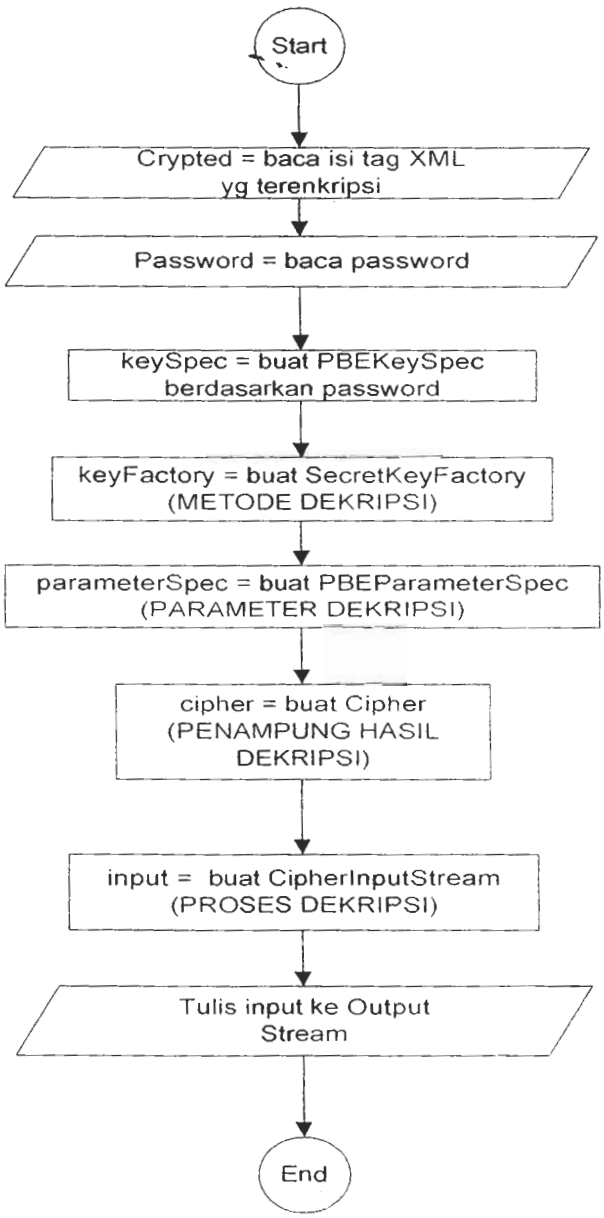


Gambar 3.5 Diagram alir Enkripsi

3.7.2 Dekripsi

Data *chipertex* yang merupakan data hasil pengolahan enkripsi dari sebuah data *plaintex* kemudian akan didekripsi kembali menjadi tampilan data *plaintex*. Pada proses dekripsi ini data *chipertex* akan dikombinasikan dengan kunci *password* sehingga akan menghasilkan data *plaintex* yang sama seperti

aslinya. Kunci *password* yang digunakan juga harus sama dengan kunci *password* pada saat melakukan proses enkripsi. Penulisan metode dekripsi pada *flowchart* dan program dapat dilihat pada Gambar 3.6 di bawah ini:



Gambar 3.6 Diagram alir Dekripsi

BAB IV
PEMBAHASAN DAN IMPLEMENTASI

4.1. Pembahasan Pengolahan data pada XML *Database* Xindice

XML *Database* Xindice digunakan sebagai *server* basis data yang didesain dari awal untuk menyimpan data XML, atau yang saat ini lebih dikenal dengan *native* XML.

Keuntungan utama penggunaan basis data *native* XML adalah memudahkan pengolahan data aplikasi yang berupa data XML. Karena dengan dipergunakannya *server* basis data *native* XML, aplikasi tidak perlu melakukan pemetaan dari data XML ke struktur data tertentu. Aplikasi dapat langsung menyimpan data XML, dan mendapatkan kembali data XML dalam format yang tidak memerlukan perubahan.

Pada saat ini, Xindice menggunakan perintah XPath untuk bahasa *query*-nya dan XML:DB *Update* untuk bahasa *update*-nya. Selain itu, fungsionalitas Xindice juga bisa diakses melalui aplikasi lain, tidak hanya dengan program Java, tapi bisa juga dengan menggunakan API XML-RPC, yang merupakan salah satu mekanisme komunikasi antar aplikasi menggunakan protokol HTTP.

Sebagai *database native* XML, Xindice memiliki kelebihan dalam bekerja menggunakan data XML. Aplikasi dapat bekerja dengan data XML secara fleksibel. Beberapa kelebihan tersebut di antaranya adalah:

1. Dokumen-dokumen XML disimpan di dalam koleksi (*collection*) yang dapat menerima perintah *query* secara keseluruhan. Aplikasi dapat membuat satu koleksi untuk satu jenis dokumen, atau satu koleksi untuk banyak jenis dokumen. Xindice mendukung penuh pilihan manapun yang dipilih.
2. Untuk melakukan *query* atau pencarian pada dokumen, dipergunakan sintaksis XPath sebagaimana yang didefinisikan oleh W3C (*World Wide Web Consortium*). Fitur ini menyediakan mekanisme yang fleksibel untuk melakukan pencarian pada dokumen dengan melakukan navigasi dan membatasi output dari *tree* yang dihasilkan.

3. Untuk mempercepat kinerja pencarian, aplikasi dapat mendefinisikan indeks terhadap elemen atau atribut. Hal ini akan meningkatkan performa pencarian secara cepat dan signifikan.
4. Memiliki fungsi XUpdate yang merupakan mekanisme penyegaran dokumen berbasis XML, yang dilakukan disisi *server*. Modifikasi dari hasil perintah XUpdate dapat diaplikasikan pada satu dokumen, atau juga pada keseluruhan dokumen.
5. Xindice mengimplementasikan dukungan penuh kepada standar pengolahan data XML yang disebut dengan XML:DB *Application Programming Interface*. Seperti JDBC yang merupakan standar pengembangan aplikasi basis data, maka XML:DB merupakan standar pengembangan aplikasi XML.
6. Untuk mendukung administrasi basis data, Xindice menyediakan seperangkat alat bantu berbasis konsol, yang merupakan implementasi dari semua fungsionalitas yang dapat diprogram melalui XML:DB API. Sehingga, admin dari basis data dapat melakukan perawatan terhadap basis data tanpa perlu membuat program Java tersendiri.
7. *Server* Xindice dibangun secara modular, sehingga sangat mudah untuk menambah atau menghapus komponen agar *server* sesuai dengan kebutuhan.

4.1.1. Pemrograman Menggunakan XML *Database* Xindice

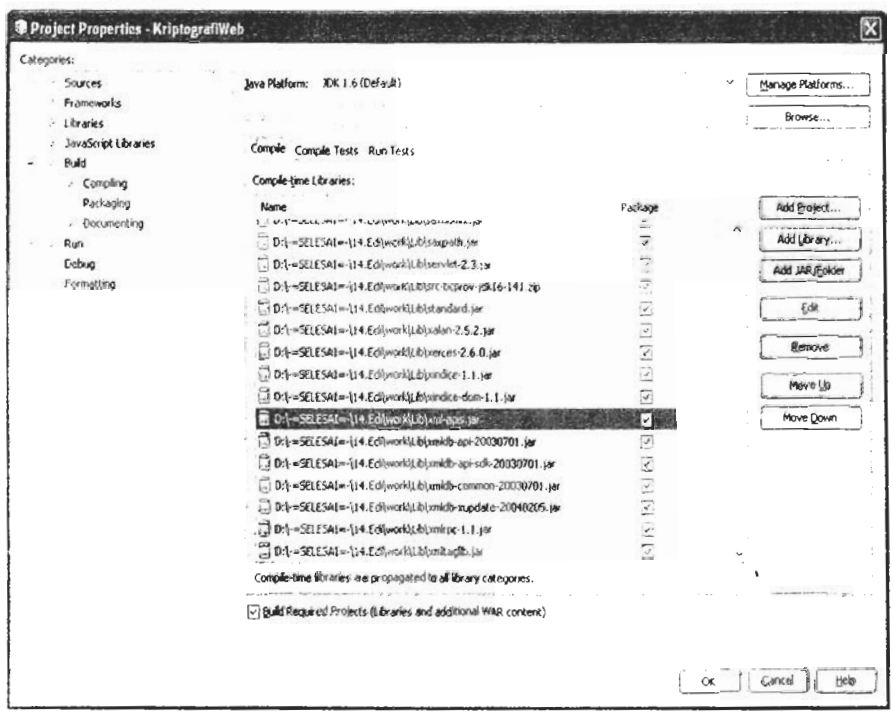
Pada penelitian ini dipergunakan Netbeans 6.0 untuk proses pengembangan sistem, sehingga dalam penjelasan persiapan sistem akan dipergunakan Netbeans 6.0 sebagai acuan.

Setelah melakukan download dari situs Xindice, maka ada beberapa file Jar yang merupakan kumpulan pustaka Java yang harus disertakan pada sembarang aplikasi yang akan mengakses basis data Xindice, yaitu:

- **xindice.jar** – berisi kelas utama Xindice yang dipergunakan oleh aplikasi klien pengakses database Xindice.

- **xmldb-common.jar**, **xmldb-api.jar**, **xmldb-api-sdk.jar**, **xmldb-xupdate.jar** – berisi implementasi XML:DB API dan XUpdate API.
- **xml-apis.jar** – berisi Java XML APIs.
- **xerces.jar** – berisi parser XML Xerces.
- **xalan.jar** – berisi mesin XSLT Xalan.
- **commons-logging.jar** – berisi paket logging Jakarta Commons Logging.

Untuk menambahkan jar tersebut pada aplikasi Netbeans, menggunakan perintah Project Properties, klik node Libraries, klik tombol Add Jar/Folder dan mengarahkan ke file-file jar tersebut. Berikut adalah tampilan Netbeans Project Properties seperti terlihat pada gambar 4.1. di bawah ini :



Gambar 4.1. Project Properties dari Netbeans

Setelah pustaka-pustaka yang diperlukan telah didefinisikan, maka project tersebut telah dapat menggunakan kelas-kelas dalam pustaka Xindice.

Untuk memulai perintah *query* dengan XPath maupun *update* dengan XUpdate, maka terlebih dahulu koneksi ke *server* basis data Xindice bisa

dilakukan. Listing 4.1. adalah implementasi program yang akan melakukan koneksi ke basis data Xindice:

```
package test.xindiced;

import java.util.logging.Level;
import java.util.logging.Logger;
import org.xmldb.api.DatabaseManager;
import org.xmldb.api.base.Database;
import org.xmldb.api.base.XMLDBException;


public class TestKoneksi {
    public static void main(String arg[]){
        try {
            String driver =
"org.apache.xindice.client.xmldb.DatabaseImpl";
            Class c = Class.forName(driver);
            Database database = (Database) c.newInstance();
            DatabaseManager.registerDatabase(database);
        } catch (XMLDBException ex) {

Logger.getLogger(TestKoneksi.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {

Logger.getLogger(TestKoneksi.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

Logger.getLogger(TestKoneksi.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (ClassNotFoundException ex) {

Logger.getLogger(TestKoneksi.class.getName()).log(Level.SEVERE,
null, ex);
        }
```

```
    }
}
```

Listing 4.1. Program Koneksi Basis Data

Kemudian dijalankan di dalam Netbeans dengan perintah SHIFT+F6, maka akan ditampilkan proses koneksi seperti pada gambar 4.2. di bawah ini:

```
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
compile-single:
run-main:
trying to register database
BUILD SUCCESSFUL (total time: 2 seconds)
```

Gambar 4.2. Output Proses Koneksi

Xindice menggunakan arsitektur modular seperti JDBC, sehingga urutan proses koneksi memiliki kesamaan yang erat dengan JDBC. Pada kode di atas, terdapat instansiasi dari kelas `org.apache.xindice.client.xmldb.DatabaseImpl`, yang merupakan *driver* untuk koneksi ke basis data tertentu, yang umumnya ditemui pada aplikasi yang menggunakan JDBC.

Setelah koneksi berhasil dilakukan, maka kelas `DatabaseManager`, yang telah mendaftarkan instan `database`, dapat dipergunakan sebagai *entry point* untuk mengakses basis data Xindice.

4.1.2 Pemrosesan Dokumen XML pada *Database* Xindice

Dokumen yang akan diproses pada *database* xindice adalah dokumen pasien.XML dan pemakai.XML seperti yang terlihat pada listing program 4.2 dan 4.3 di bawah ini.

```
<daftar_pasien>
<pasien>
```

```
  <nomorinduk>1518989</nomorinduk>
  <nama>Edy Winarno</nama>
  <nomoridentitas>003</nomoridentitas>
  <tempatlahir>Salatiga / 15 Nopember 1975</tempatlahir>
  <jeniskelamin>Laki-Laki</jeniskelamin>
  <golongandarah>A</golongandarah>
  <tinggi>165</tinggi>
  <berat>68</berat>
  <telepon>09999999</telepon>
  <pekerjaan>swasta</pekerjaan>
  <alergiobat>Asetaminofen</alergiobat>
  <alamat>Semarang</alamat>
  <riwayatpemeriksaan />
</pasien>
</daftar_pasien>
```

Listing 4.2. Program pasien.xml

```
<daftar_pemakai>
<pemakai>
<idpemakai>1</idpemakai>
<nama>edy</nama>
<sandi>admin</sandi>
<peran>Administrator</peran>
</pemakai>
</daftar_pemakai>
```

Listing 4.3 Program pemakai.xml

Berdasarkan kode koneksi ke basis data *server* Xindice sebelumnya, maka kode untuk pengaksesan data dengan XPath menggunakan XPath *Query Engine* milik Xindice dapat dilakukan seperti ditunjukkan pada listing 4.4. berikut:

```
package test.xindiced;

import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;
import org.xmldb.api.*;

public class TestQuery {
    public static void main(String[] args) throws Exception {
        Collection col = null;
        try {
            String driver =
"org.apache.xindice.client.xmldb.DatabaseImpl";
            Class c = Class.forName(driver);
            Database database = (Database) c.newInstance();
            DatabaseManager.registerDatabase(database);
            col =
DatabaseManager.getCollection("xmldb:xindice:///db/edy");
            String xpath = "//pasien";
            XPathQueryService service = (XPathQueryService)
col.getService("XPathQueryService", "1.0");
            ResourceSet resultSet = service.query(xpath);
            ResourceIterator results = resultSet.getIterator();
            int i = 1;
            while (results.hasMoreResources()) {
                Resource res = results.nextResource();
                System.out.println(i + ":" + (String)
res.getContent());
                i++;
            }
        } catch (XMLDBException e) {
            System.err.println("XML:DB Exception occured " +
e.errorCode + "," + e.getMessage());
        } finally {
            if (col != null) {
                col.close();
            }
        }
```

```
    }
}
```

Listing 4.4. Penggunaan Xpath dalam kode Java

Pada kode tersebut di atas, terlihat bahwa variabel *collection* menampung hasil eksekusi perintah *DatabaseManager.getCollection*, yang mengarahkan perintah-perintah baik XUpdate maupun XPath ke koleksi edy.

```
col = DatabaseManager.getCollection("xmldb:xindice:///db/edy");
```

Selanjutnya, variabel xpath diberikan nilai sintaksis XPath yang diinginkan, yang dalam hal ini adalah menampilkan semua elemen pasien. Jika diinginkan perintah XPath yang lain, maka isi dari variabel xpath ini dapat diubah sesuai dengan kebutuhan.

```
String xpath = "//pasien";
```

Kemudian engine query XPath milik Xindice diinisialisasi seperti berikut:

```
XPathQueryService service = (XPathQueryService)
col.getService("XPathQueryService", "1.0");
```

Eksekusi perintah XPath dilakukan pada baris perintah service.query berikut:

```
ResourceSet resultSet = service.query(xpath);
```

Keberhasilan proses tersebut, akan disimpan ke dalam variabel resultSet yang dapat diakses dengan menggunakan *iterator*, yang merupakan standar Java untuk pengolahan struktur data berupa list . Pengolahan isi dari resultSet:

```
ResourceIterator results = resultSet.getIterator();
int i = 1;
while (results.hasMoreResources()) {
Resource res = results.nextResource();
  System.out.println(i + ":" + (String) res.getContent());
  i++;
```

}

Adapun tampilan dari program tersebut di dalam Netbeans 6.0 adalah seperti ditunjukkan pada gambar 4.3 berikut ini :

```
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
compile-single:
run-main:
trying to register database
1:<pasien src:col="/db/edy" src:key="pasien"
xmlns:src="http://xml.apache.org/xindice/Query"><nomorinduk>151898
9</nomorinduk><nama>Edy</nama><nomoridentitas>2222</nomoridentitas
><tempatlahir>Denpasar</tempatlahir><jeniskelamin>Laki-
Laki</jeniskelamin><golongandarah>B</golongandarah><tinggi>65</tin
ggi><berat>68</berat><telepon>09999999</telepon><pekerjaan>Program
mer</pekerjaan><alergiobat>Zh5AJ9WggJU=</alergiobat><alamat>Semara
ng</alamat><riwayatpemeriksaan /></pasien>
BUILD SUCCESSFUL (total time: 3 seconds)
```

Gambar 4.3. Tampilan Program Eksekusi XPath pada NetBeans 6.0

4.1.3. Modifikasi Dokumen XML pada Basis Data dengan XUpdate

Seperti perintah pada basis data RDBMS (*Relational Data Base Management System*), maka modifikasi dokumen pada Xindice juga terdiri dari *INSERT*, *UPDATE* dan *DELETE*. Masing-masing perintah digunakan untuk memproses dan memodifikasi setiap adanya perubahan data yang diinginkan pada basis data. Untuk setiap tahap perintah akan dipergunakan perintah menggunakan konsolnya.

Perintah XUpdate ini bisa sangat panjang dan memerlukan modifikasi yang sering, maka pada utilitas konsol, perintah tersebut disimpan di dalam file teks

tersendiri, untuk kemudian disertakan nama filenya pada setiap perintah modifikasi dokumen.

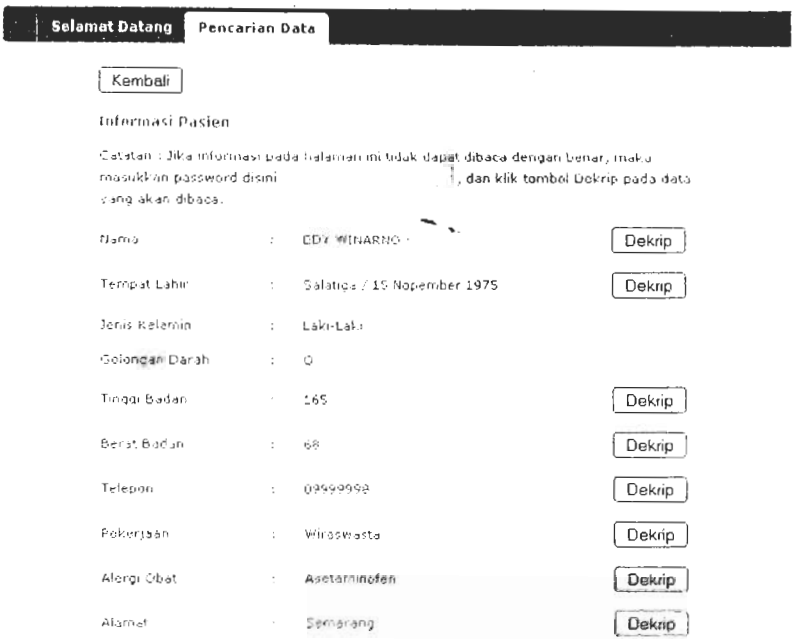
Adapun nilai dari variabel *xupdate* yang akan menentukan efek dari eksekusi servis tersebut, yaitu bisa bertipe INSERT, UPDATE ataupun DELETE.

Untuk menambahkan data baru, yaitu pasien dengan nomorinduk 1518990, maka dipergunakan perintah modifikasi XUpdate yang disimpan ke dalam file insert2.xml seperti ditunjukkan pada listing 4.5. berikut ini:

```
<xupdate:modifications version="1.0"
xmlns:xupdate="http://www.xmldb.org/xupdate">
<xupdate:append select="/daftar_pasien">
<xupdate:element name="pasien">
<nomorinduk>1518990</nomorinduk>
<nama>Yuni Astuti</nama>
<nomoridentitas>KTP-1109099922</nomoridentitas>
<tempatlahir>Semarang</tempatlahir>
<jeniskelamin>Wanita</jeniskelamin>
<golongandarah>O</golongandarah>
<tinggi>160</tinggi>
<berat>45</berat>
<telepon>024-566-xxx</telepon>
<pekerjaan>Swasta</pekerjaan>
<alergiobat>Amphicillyn</alergiobat>
<alamat>Semarang</alamat>
<riwayatpemeriksaan/>
</xupdate:element>
</xupdate:append>
</xupdate:modifications>
```

Listing 4.5. Isi Perintah modifikasi INSERT pada insert2.xml

Untuk memulai perintah ini, berikut akan diperlihatkan hasil eksaminasi isi data pasien sebelum modifikasi dengan perintah *xindice xpath* seperti ditunjukkan pada gambar 4.4 berikut ini:

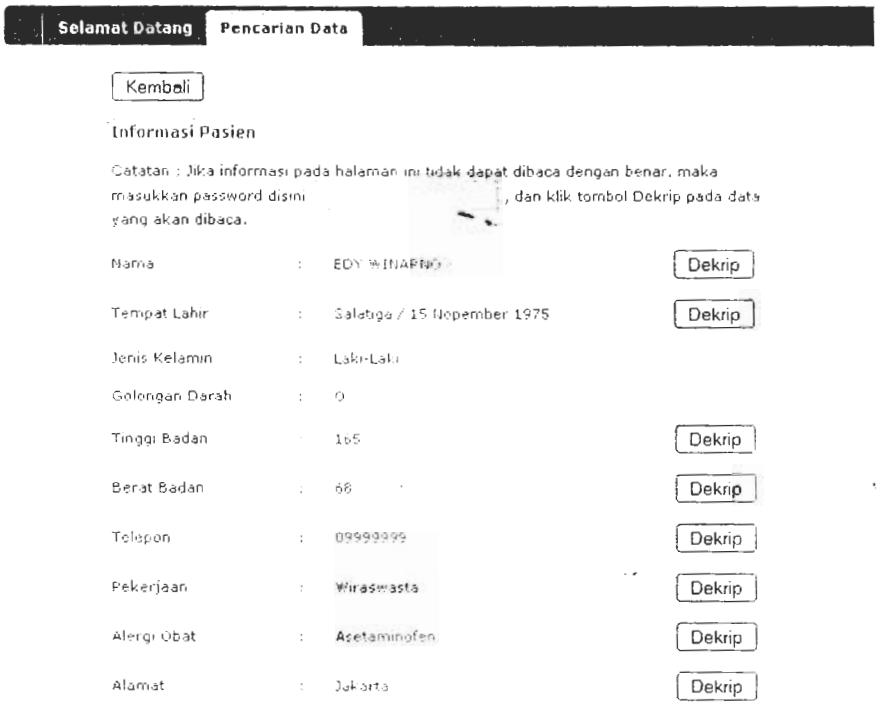


Gambar 4.4. Data Pasien Sebelum mendapat Perintah XUpdate

Tampak bahwa nama pasien dengan nomorinduk 1518989 tersebut memiliki nama EDY WINARNO dengan alamat **Semarang**. Selanjutnya nilai dari tag alamat akan diubah ke alamat yang baru yaitu alamat **Jakarta**, dengan perintah XUpdate yang disimpan ke dalam file teks `update2.xml` seperti pada listing 4.6. berikut ini:

```
<xupdate:modifications version="1.0"
xmlns:xupdate="http://www.xmldb.org/xupdate">
<xupdate:update select="//pasien[nomorinduk=1518989]/alamat">
Jakarta
</xupdate:update>
</xupdate:modifications>
```

Listing 4.6. Isi Perintah Modifikasi Update pada update2.xml



Gambar 4.5. Hasil Eksekusi Perintah XUpdate UPDATE

Tampak bahwa isi tag alamat untuk elemen pasien dengan nomorinduk 1518989 sudah dimodifikasi untuk alamat yang baru yaitu alamat **Jakarta.**

Proses penghapusan elemen pada dokumen Xindice menggunakan format perintah XUpdate yang disimpan pada file `delete2.xml` seperti listing 4.7. berikut ini:

```
<xu:modifications version="1.0"
xmlns:xu="http://www.xmldb.org/xupdate">
<xu:remove select="//pasien[nomorinduk=1157086]"/>
</xu:modifications>
```

Listing 4.7. Isi Perintah Modifikasi DELETE pada delete2.xml

Sedangkan untuk versi Java, baik perintah INSERT, UPDATE atau DELETE tidak mempunyai perbedaan kode, melainkan sintaksis dari perintah

XUpdate yang diinginkan. Listing 4.8. adalah kode progam Java yang diperlukan untuk menjalankan perintah XUpdate tersebut:

```
package test.xindiced;
import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;
import org.xmldb.api.*;
public class XUpdate {
    public static void main(String[] args) throws Exception {
        Collection col = null;
        try {
            String driver =
"org.apache.xindice.client.xmldb.DatabaseImpl";
            Class c = Class.forName(driver);
            Database database = (Database) c.newInstance();
            DatabaseManager.registerDatabase(database);
            col =
DatabaseManager.getCollection("xmldb:xindice:///db/edy");
            String xpath = "//pasien[nomorinduk=1518989]/nama";
            String data = "Edy Winarno";
            String xupdate="<xupdate:modifications version=\"1.0\"
xmlns:xupdate=\"http://www.xmldb.org/xupdate\">" +
                "<xupdate:update select=\"" + xpath + "\">" + data +
"</xupdate:update>" +
                "</xupdate:modifications>";
            XUpdateQueryService service =
                    (XUpdateQueryService)
col.getService("XUpdateQueryService", "1.0");
            service.update(xupdate);
        } catch (XMLDBException e) {
            System.err.println("XML:DB Exception occured " + e.errorCode
+ " " +
                    e.getMessage());
        } finally {
            if (col != null) {
                col.close();
            }
        }
    }
```

Listing 4.8. Kode Perintah Java untuk XUpdate

Perbedaan program ini dengan program Java yang menggunakan query dengan XPath, adalah service yang di-instansiasi. Jika yang menggunakan XPath service adalah bertipe XPathQueryService, maka disini service bertipe XUpdateQueryService, seperti ditunjukkan pada Listing 4.9. di bawah ini :

```
...
XUpdateQueryService service =
(XUpdateQueryService)
col.getService("XUpdateQueryService", "1.0");
   service.update(xupdate);
...
```

Listing 4.9. XUpdateService untuk Eksekusi Perintah-Perintah XUpdate

4.2. Pembahasan Kriptografi

Data-data XML yang telah diolah menggunakan proses penyimpanan dan modifikasi pada *database* Xindice selanjutnya akan diproses menggunakan teknik kriptografi. Teknik kriptografi dilakukan dengan cara mengenkripsi dan mendekripsi tag-tag XML yang diperlukan. Enkripsi dan dekripsi dilakukan menggunakan kriptografi *Password Based Encryption* yang telah disediakan oleh JCE (*Java Crytography Extension*) dengan algoritma yang digunakan adalah metode PBEwithMD5andDES. Sebelum dilakukan proses enkripsi dan dekripsi, urutan pengolahan data yang digunakan mulai dari proses pengambilan data sampai ke menampilkan data yang telah dienkripsi dan didekripsi adalah sebagai berikut:

1. Registrasi *Server Database*

   Registrasi ini dilakukan pada saat awal proses dijalankan, yaitu dengan pengaktifan *server database* yang digunakan sebagai tempat penyimpanan dan pemrosesan data secara keseluruhan.

2. Pengambilan data *Collection* pada *Database*

   Pada proses ini dilakukan pengambilan data yang telah disimpan pada basis data yaitu berupa collection yang didalamnya berisi dokumen-dokumen XML yang akan diproses menggunakan kriptografi.

3. Instantiasi Xpath dan Xupdate

Pada proses ini dilakukan pengaturan dan modifikasi dokumen XML menggunakan XpathQueryService dan XupdateQueryService.

4. Proses Enkripsi pada Tag yang diperlukan

   Enkripsi dilakukan pada tag yang ditentukan sesuai dengan yang dipilih. Enkripsi bisa dilakukan lebih dari satu tag.

5. Proses Dekripsi pada Tag yang diperlukan

   Pada proses dekripsi dilakukan hal yang sama seperti pada proses enkripsi. Dekripsi juga bisa dilakukan lebih dari satu tag.

6. Menampilkan informasi data hasil Enkripsi dan Dekripsi

   Hasil proses enkripsi dan dekripsi kemudian ditampilkan sebagai hasil tampilan akhir pada aplikasi web berupa hasil enkripsi (*chipertext*) dan hasil dekripsi (*plaintext*)

4.2.1. Pembahasan Proses Enkripsi dan Dekripsi

Proses enkripsi dan dekripsi dilakukan dengan menggunakan *Password Based Encryption* yang diperoleh dari JCE *(Java Cryptography Extension)*. Ada 3 class yang digunakan pada penggunaan teknik kriptografi ini yaitu :

1. PBEParameterSpec Class

   Javax.crypto.spec.PBEParameterSpec class merupakan class yang disediakan sebagai pembawa *salt* dan *itetarion count* untuk membantu proses enkripsi dan dekripsinya.

2. The PBEKeySpec Class

   Javax.crypto.spec.PBEKeySpec merupakan class yang digunakan untuk memproses *password* yang digunakan dalam proses enkripsi dan dekripsinya.

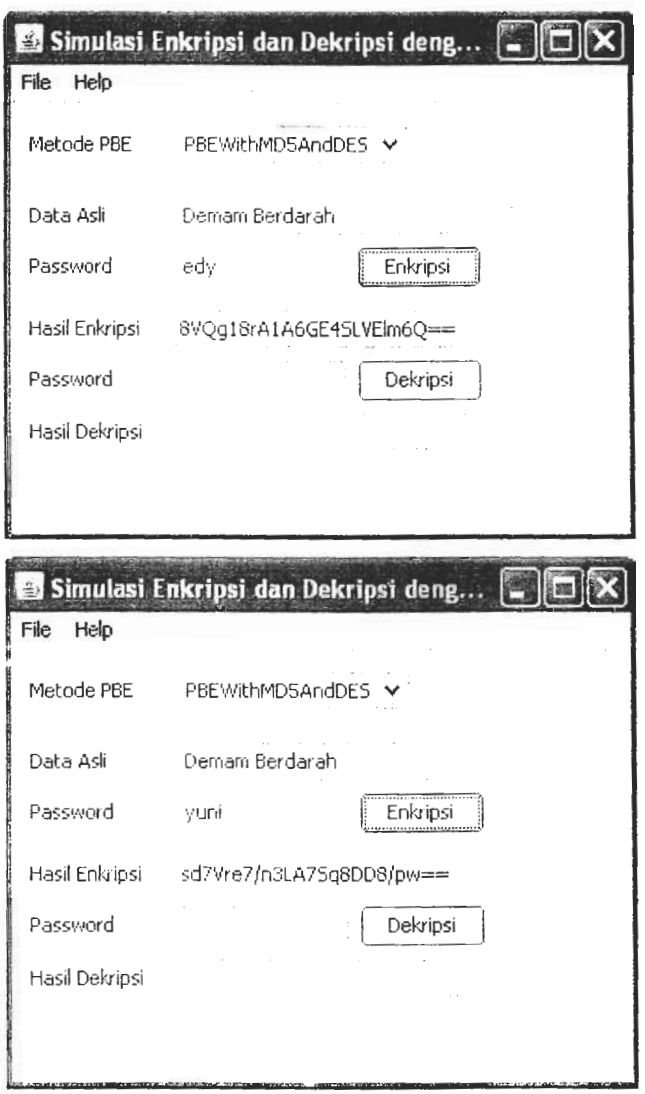
3. The SecretKeyFactory Class

   Javax.crypto.SecretKeyFactory class merupakan class yang digunakan untuk mengkonversikan kunci yang digunakan pada proses enkripsi dan dekripsinya.

Seperti pada class JCE yang lainnya, SecretKeyFactory dibuat menggunakan metode getInstance ( ) method. Penulisan pada kode program pada java dituliskan :

```
private static String METHOD = "PBEWithMD5AndDES";
```

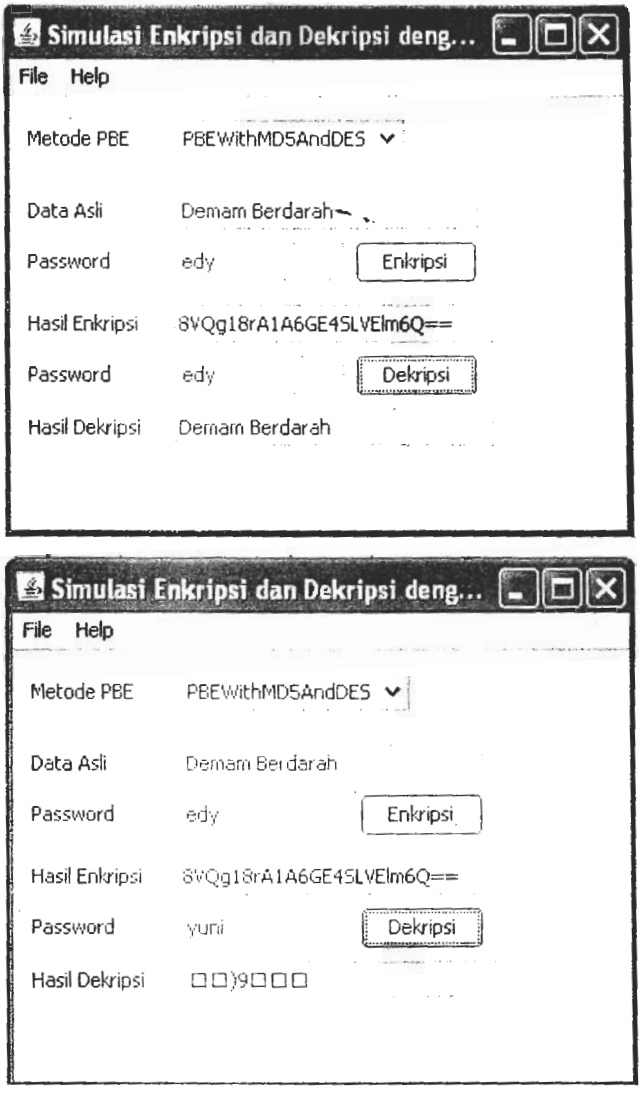
Penggunaan kunci *password* yang berbeda pada proses enkripsi dengan menggunakan data plainteks yang sama akan dihasilkan data chiperteks yang berbeda pula. Gambar 4.6 menunjukkan simulasi hasil data chiperteks yang berbeda dari 2 buah proses enkripsi menggunakan data plainteks yang sama dan dengan kunci *password* yang berbeda.



Gambar 4.6 Hasil enkripsi dengan kunci *password* berbeda

Data plainteks akan diproses menjadi data chiperteks melalui proses pengolahan pada proses enkripsi menggunakan metode *Password Based Encryption*. Data plainteks akan diolah dengan kunci *password*, *salt* dan *iteration count* sehingga akan didapatkan data terenkripsi yang berupa chiperteks. *Salt* digunakan untuk menambah sebuah *string* dari *byte-byte* yang random pada *password*. *Iteration count* digunakan untuk menambah perhitungan waktu yang dibutuhkan untuk mengkonversi sebuah *password* menjadi sebuah kunci. Data plainteks dan kunci tersebut akan diproses pada proses *chiper* untuk menghasilkan *encrypted data* yang disebut dengan chiperteks.

Pada proses dekripsi, data chiperteks yang merupakan hasil pengolahan proses enkripsi dapat dikembalikan lagi menjadi data plainteks atau data asli sebelumnya. Proses ini merupakan kebalikan dari proses enkripsi. *Encrypted data* yang berupa data chiperteks bersama-sama dengan *password* akan diproses untuk menghasilkan data plainteks kembali. *Password, salt, iteration count*, dan chiperteks akan diubah menjadi sebuah kunci dan akan diproses menjadi data plainteks seperti semula. Pada proses dekripsi, *password* akan sangat menentukan pada hasil proses pengembalian data chiperteks menjadi data plainteks. *Password* yang sama dengan *password* saat proses enkripsi akan dapat mengembalikan data chiperteks menjadi data plainteks aslinya. Gambar 4.7 memperlihatkan simulasi perbedaan data hasil dekripsi antara *password* yang benar dan yang salah pada proses aplikasi.



Gambar 4.7 Hasil dekripsi menggunakan *password* berbeda

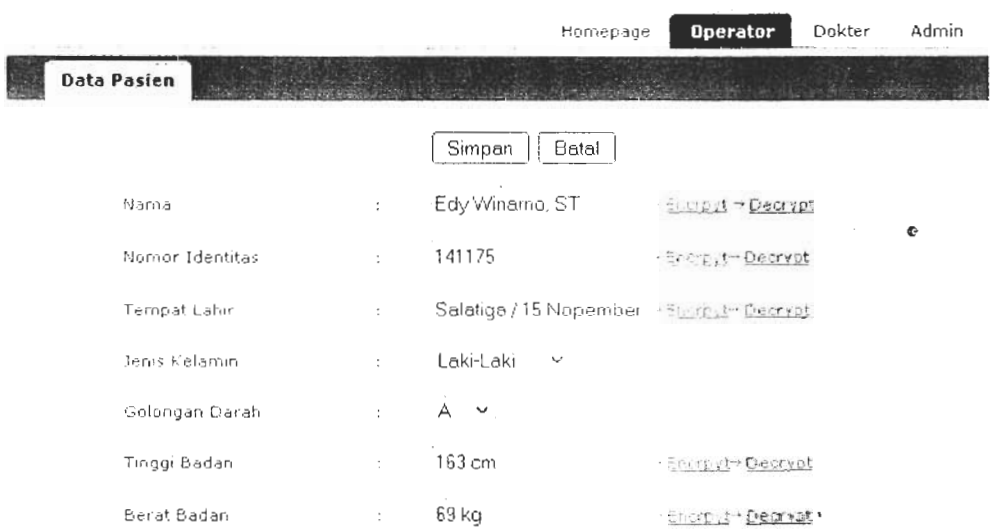
4.2.2. Pembahasan Implementasi Enkripsi dan Dekripsi pada Web

Aplikasi penerapan kriptografi *Password Based Encryption* untuk keamanan distribusi data menggunakan data base XML yang dibuat dalam penelitian ini adalah sebuah implementasi program pada pertukaran data rekam medis pasien di sebuah web rumah sakit dengan dilakukan proses enkripsi dan dekripsi pada bagian *user*, *operator* dan dokter. Operator dan dokter dapat melakukan proses enkripsi dan dekripsi serta menyimpannya untuk digunakan pada bagian yang lain. Sedangkan user atau pasien hanya dapat melakukan proses dekripsi dari data yang telah dienkripsi oleh *operator* dan dokter.

Pada bagian operator, enkripsi dan dekripsi dilakukan untuk menyembunyikan data informasi pasien yang dapat dipilih sesuai dengan tag XML yang dipilih dan juga untuk mengembalikan data informasi pasien yang telah dienkripsi. Sedangkan pada bagian dokter, enkripsi dilakukan untuk menyembunyikan data pemeriksaan pasien dan mengembalikan data pemeriksaan pasien yang telah dienkrip.

4.2.2.1. Enkripsi dan Dekripsi pada bagian operator

Pada bagian *operator* enkripsi dan dekripsi dapat dilakukan sesuai dengan data informasi pasien yang dipilih. Ketika *operator* memasukkan data-data seorang pasien maka tampilan yang akan muncul adalah seperti contoh Gambar 4.8 di bawah ini.



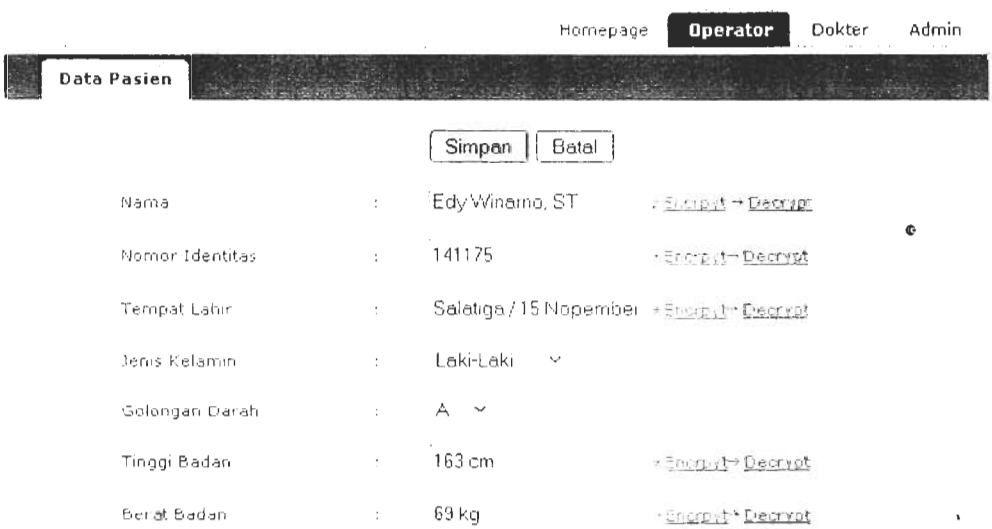
Gambar 4.8 Isian data pada bagian operator sebelum dienkripsi

Dari tampilan diatas terlihat bahwa terdapat menu *encrypt* dan *decrypt* yang digunakan untuk melakukan proses enkripsi dan dekripsi dari data informasi yang diinginkan. Enkripsi pada bagian *operator* ini dapat dilakukan pada sebuah data informasi pasien ataupun sekaligus beberapa data informasi yang dikehendaki. Enkripsi dilakukan pada tag-tag XML yang dipilih yang telah disimpan pada *database* Xindice XML. Tag-tag inilah yang ditampilkan menjadi

Pada bagian operator, enkripsi dan dekripsi dilakukan untuk menyembunyikan data informasi pasien yang dapat dipilih sesuai dengan tag XML yang dipilih dan juga untuk mengembalikan data informasi pasien yang telah dienkripsi. Sedangkan pada bagian dokter, enkripsi dilakukan untuk menyembunyikan data pemeriksaan pasien dan mengembalikan data pemeriksaan pasien yang telah dienkrip.

4.2.2.1. Enkripsi dan Dekripsi pada bagian operator

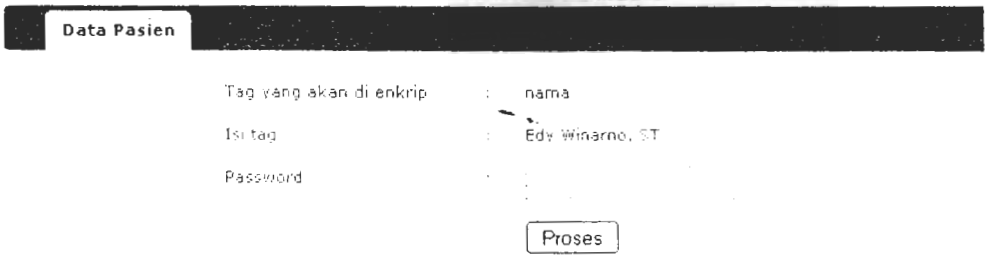
Pada bagian *operator* enkripsi dan dekripsi dapat dilakukan sesuai dengan data informasi pasien yang dipilih. Ketika *operator* memasukkan data-data seorang pasien maka tampilan yang akan muncul adalah seperti contoh Gambar 4.8 di bawah ini.



Gambar 4.8 Isian data pada bagian operator sebelum dienkripsi

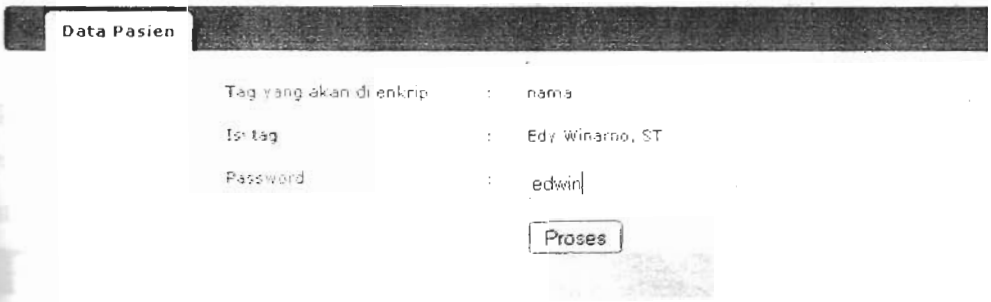
Dari tampilan diatas terlihat bahwa terdapat menu *encrypt* dan *decrypt* yang digunakan untuk melakukan proses enkripsi dan dekripsi dari data informasi yang diinginkan. Enkripsi pada bagian *operator* ini dapat dilakukan pada sebuah data informasi pasien ataupun sekaligus beberapa data informasi yang dikehendaki. Enkripsi dilakukan pada tag-tag XML yang dipilih yang telah disimpan pada *database* Xindice XML. Tag-tag inilah yang ditampilkan menjadi

tampilan masing-masing data informasi yang dapat dienkripsi. Gambar 4.9 menunjukkan sebuah data informasi yang akan dilakukan proses enkripsi.



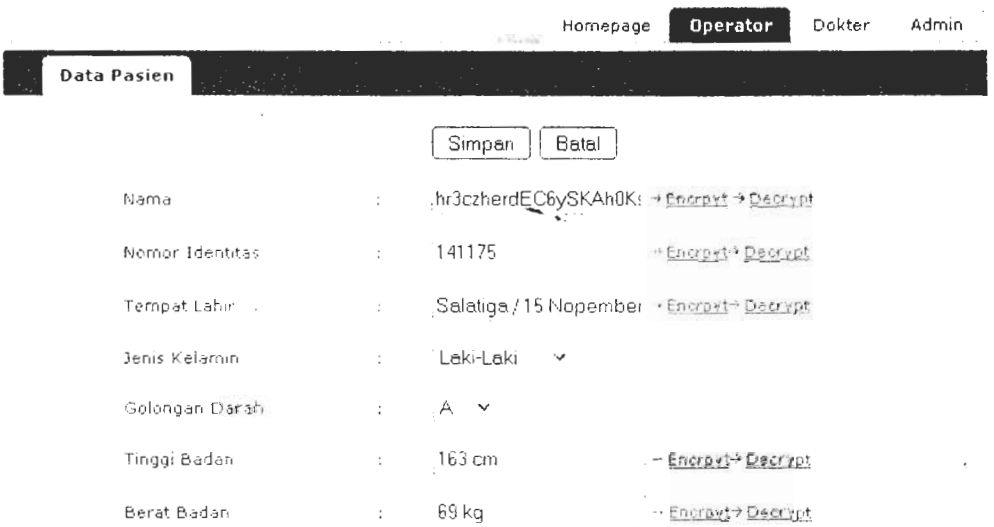
Gambar 4.9 Sebuah data informasi yang akan dienkripsi

Gambar diatas merupakan sebuah contoh *tag* yang telah dipilih untuk dilakukan proses enkripsi. Gambar 4.10 menunjukkan sebuah contoh pengisian menggunakan kunci *password* untuk dilakukan proses enkripsi.



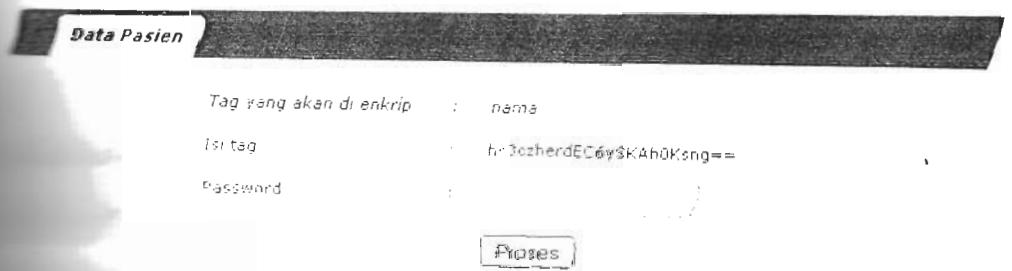
Gambar 4.10 Pengisian *password* pada tag yang akan dienkripsi

Setelah kunci *password* dimasukkan maka data informasi pada tag yang telah dipilih tadi akan diproses menggunakan kriptografi PBE with MD5 and DES menjadi data *chipertex* yang tidak akan dapat dibaca oleh siapapun. Hasil *chipertex* ini merupakan hasil dari proses enkripsi dimana setiap output *chipertex* hasil enkripsi akan memiliki hasil yang berbeda-beda tergantung dari kunci *password* yang dimasukkan. Pada proses pengisian *password* pada bagian *operator* ini menggunakan kunci *password* yang sama antara masing-masing tag. Hal ini bertujuan untuk memudahkan user atau pasien ketika melakukan proses dekripsi. Gambar 4.11 merupakan tampilan hasil *chipertex* dari sebuah proses enkripsi pada sebuah tag yang dipilih.



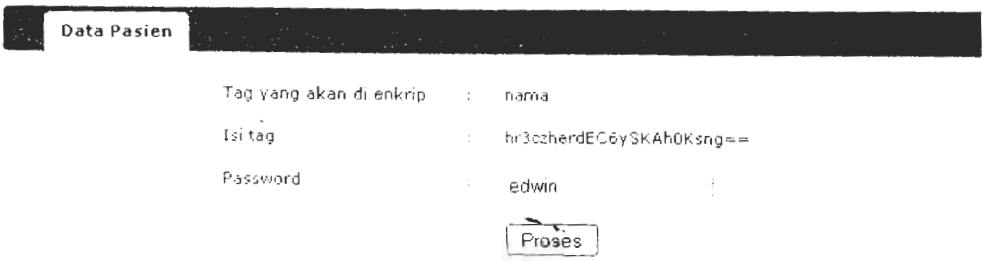
Gambar 4.11 Hasil chipertex dari sebuah tag yang dienkripsi.

Proses dekripsi pada bagian *operator* hampir sama dengan proses enkripsinya. Jika akan mendekripsi sebuah data informasi pada sebuah *tag*, maka diperlukan kunci *password* yang sama dengan kunci password saat melakukan proses enkripsi. Gambar 4.12 adalah tampilan data informasi yang telah dipilih untuk dilakukan proses dekripsi.



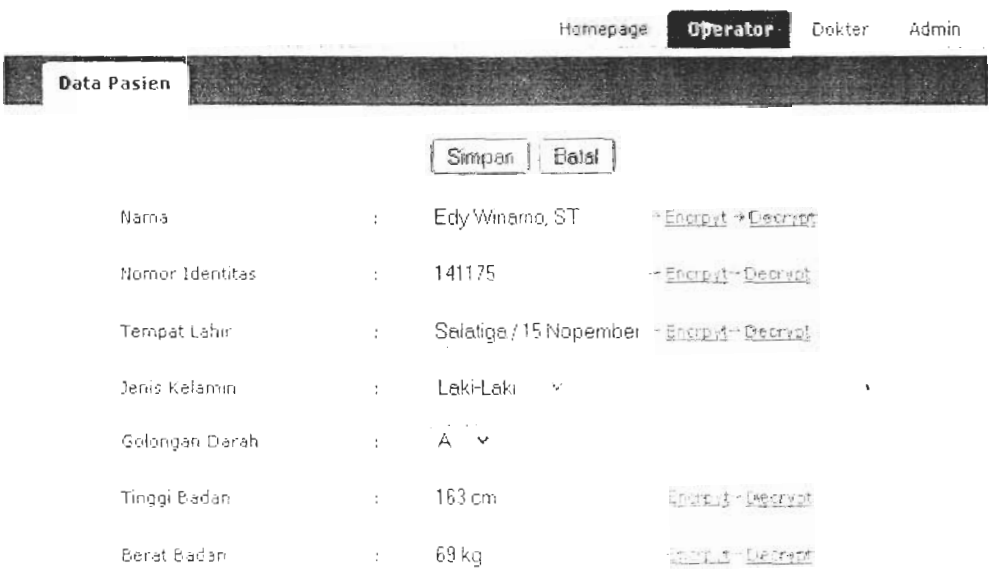
Gambar 4.12 Data informasi chiperteks yang akan didekripsi

Pada gambar 4.13 menunjukkan pengisian kunci password yang sama



Gambar 4.13 Pengisian kunci password pada tag yang akan didekripsi

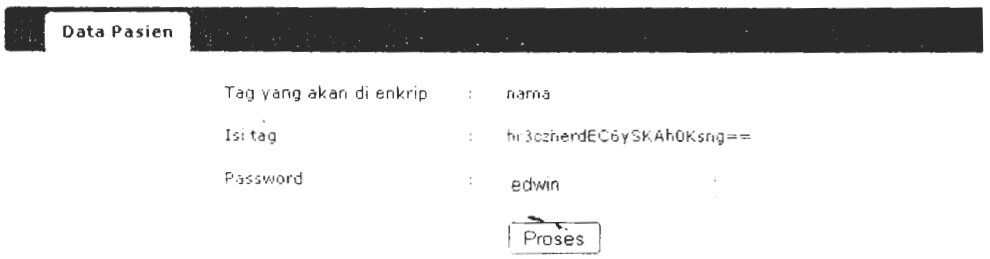
Setelah kunci *password* dimasukkan maka tampilan data informasi yang berupa data *chipertex* tadi akan dikembalikan pada kondisi semula, sehingga akan ditampilkan data informasi yang sama dengan data informasi saat sebelum dilakukan proses enkripsi. Gambar 4.14 merupakan tampilan hasil dari proses dekripsi pada data informasi dimana data yang sebelumnya berupa data *chipertex* telah berubah kembali menjadi data seperti semula.



Gambar 4.14 Hasil data dari sebuah proses dekripsi

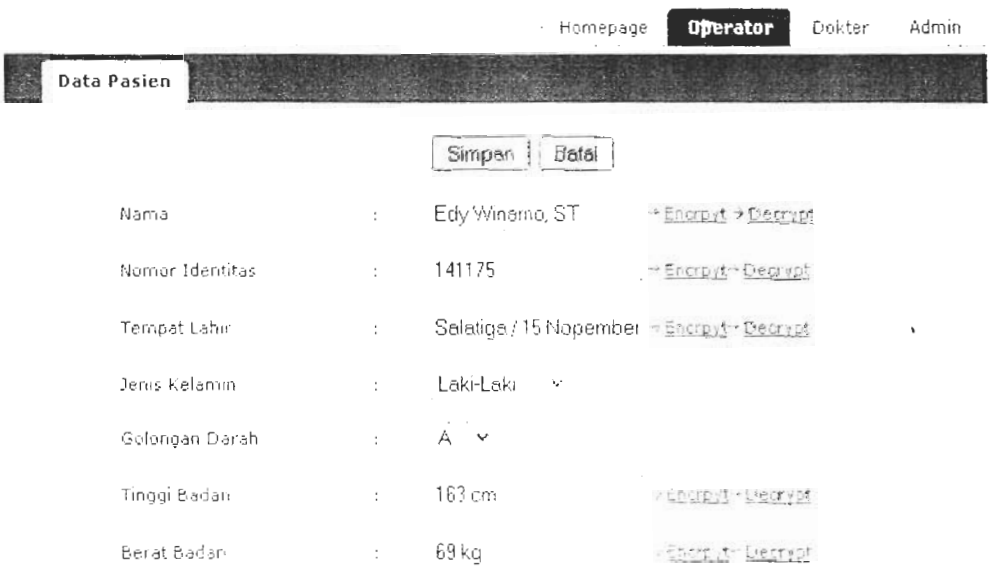
4.2.2.2. Enkripsi dan Dekripsi pada bagian dokter

Data yang telah diolah menggunakan proses enkripsi pada bagian *operator* kemudian akan dipanggil dan ditampilkan pada bagian dokter. Data ini kemudian



Gambar 4.13 Pengisian kunci password pada tag yang akan didekripsi

Setelah kunci *password* dimasukkan maka tampilan data informasi yang berupa data *chipertex* tadi akan dikembalikan pada kondisi semula, sehingga akan ditampilkan data informasi yang sama dengan data informasi saat sebelum dilakukan proses enkripsi. Gambar 4.14 merupakan tampilan hasil dari proses dekripsi pada data informasi dimana data yang sebelumnya berupa data *chipertex* telah berubah kembali menjadi data seperti semula.

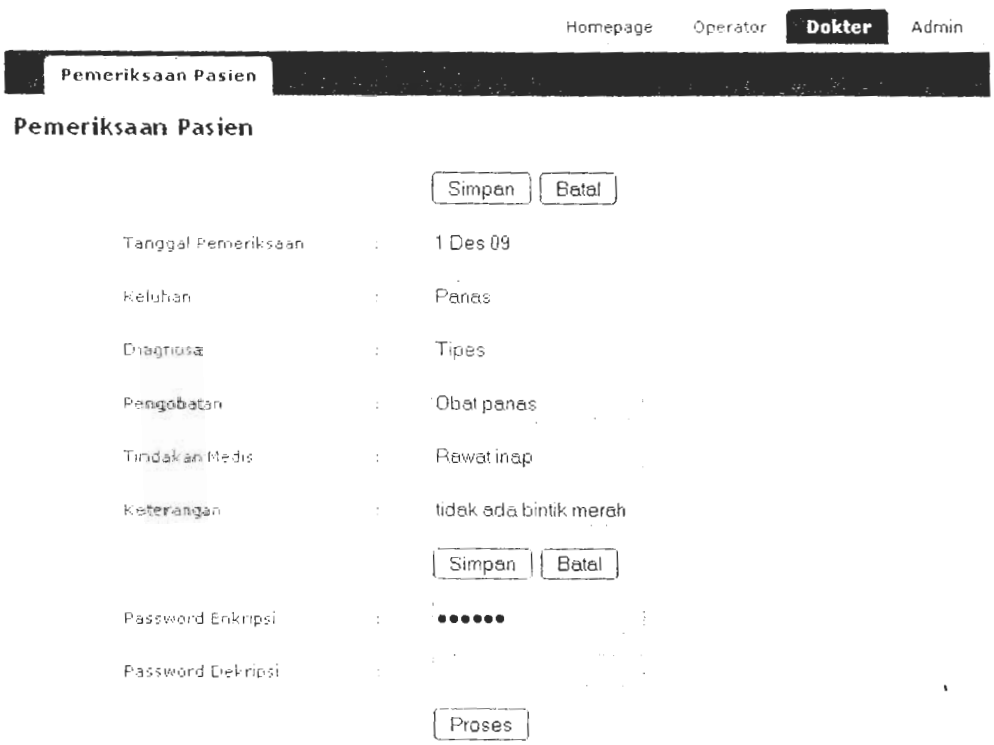


Gambar 4.14 Hasil data dari sebuah proses dekripsi

4.2.2.2. Enkripsi dan Dekripsi pada bagian dokter

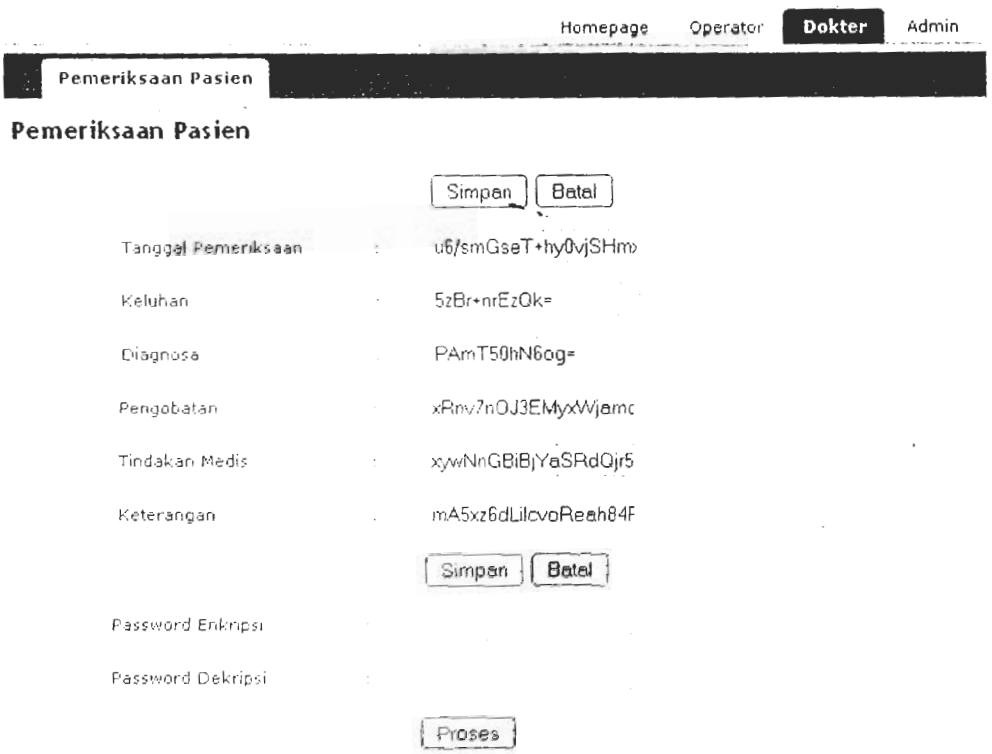
Data yang telah diolah menggunakan proses enkripsi pada bagian *operator* kemudian akan dipanggil dan ditampilkan pada bagian dokter. Data ini kemudian

ditambah dengan masukan data tambahan dari dokter. Data tambahan merupakan data informasi riwayat pemeriksaan pasien yang bisa diisi, dihapus, di-*update* dienkripsi dan didekripsi oleh dokter. Pada bagian dokter proses enkripsi dan dekripsi dapat dilakukan sesuai dengan data informasi pasien yang dipilih. Dokter dapat melihat data informasi pasien dengan memasukkan kunci *password* berupa nomor identitas pasien. Kemudian dokter akan menambahkan data riwayat pemeriksaan pasien pada menu pemeriksaan pasien. Gambar 4.15 adalah contoh tampilan data riwayat pemeriksaan pasien yang telah diisikan oleh dokter.



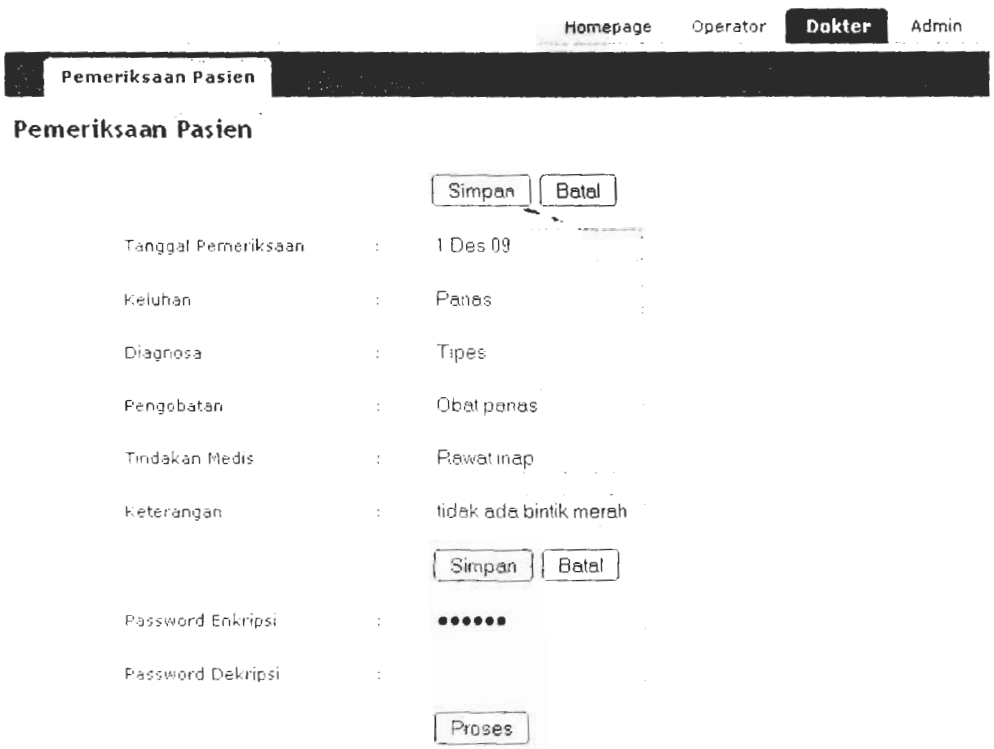
Gambar 4.15. Data isian pemeriksaan pasien pada bagian dokter

Untuk melakukan proses enkripsi pada bagian dokter diperlukan sebuah kunci *pasword* untuk mengubah data informasi riwayat pemeriksaan pasien menjadi tampilan data *chipertex* yang tidak dapat dibaca. Pada bagian dokter seluruh data pemeriksaan akan dienkripsi menjadi tampilan *chipertex* seperti yang terlihat pada Gambar 4.16 di bawah ini.



Gambar 4.16 Hasil chipertex dari data riwayat pemeriksaan yang dienkripsi

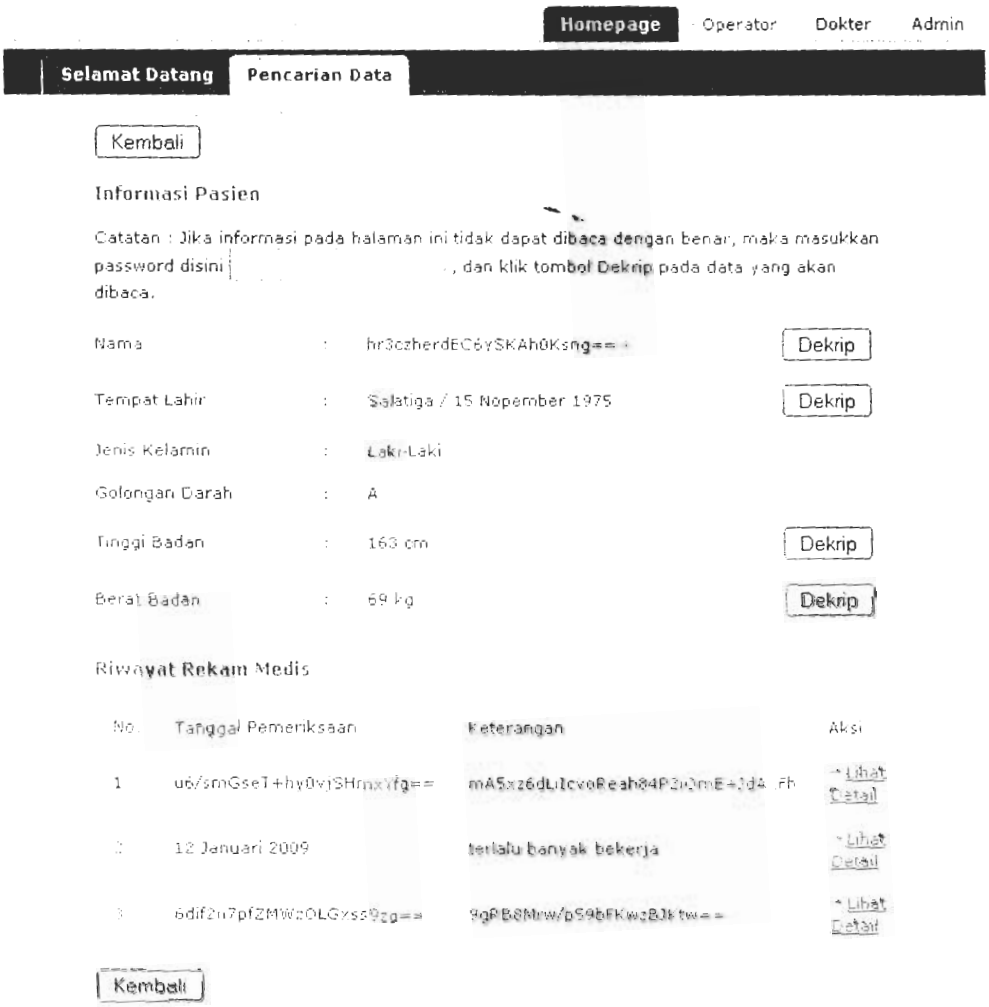
Data riwayat pemeriksaan pasien yang telah dienkripsi ini nantinya akan disimpan dan akan ditampilkan bersama-sama dengan data informasi pasien yang akan ditampilkan pada bagian *homepage* untuk dapat dilakukan proses dekripsi oleh *user* atau pasien. Dokter juga dapat melakukan proses dekripsi untuk melihat data riwayat pemeriksaan yang telah dienkripsi tersebut. Untuk melakukan proses dekripsi diperlukan kunci *password* yang sama dengan kunci *password* pada proses enkripsi. Contoh hasil tampilan dari sebuah data riwayat pemeriksaan pasien yang telah dilakukan proses dekripsi dapat dilihat pada Gambar 4.17 di bawah ini.



Gambar 4.17 Hasil proses dekripsi pada bagian dokter

4.2.2.3. Dekripsi pada bagian user

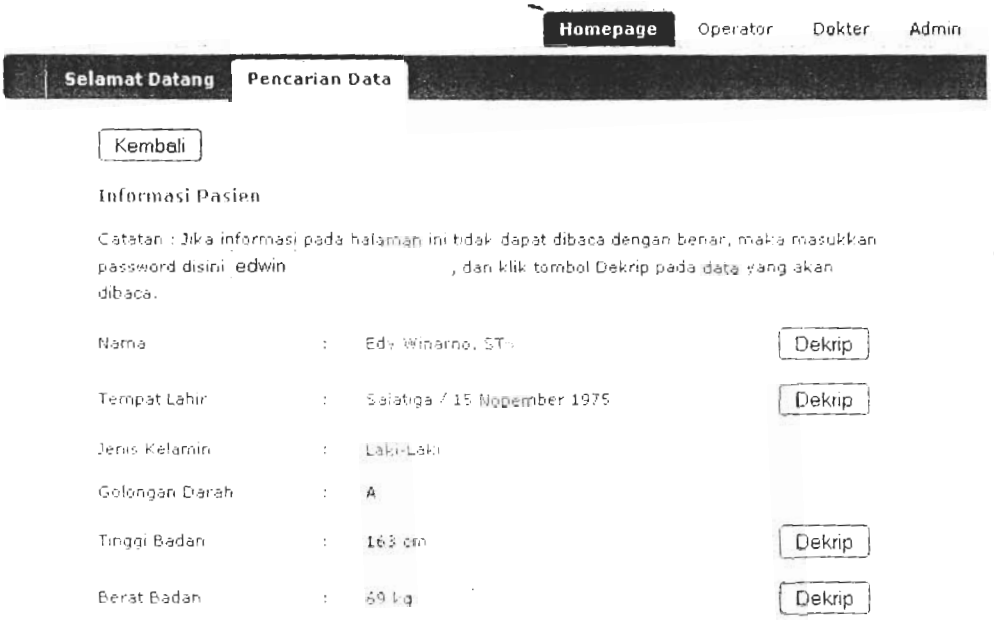
Bagian *user* adalah bagian terakhir yang menerima seluruh data informasi yang telah dilakukan proses enkripsi baik dari bagian *operator* maupun bagian dokter. Data yang akan ditampilkan pada bagian *user* adalah data informasi pasien yang telah dienkripsi sesuai dengan *tag* yang dipilih ditambah dengan data riwayat pemeriksaan pasien yang telah dienkripsi oleh dokter. Contoh tampilan data informasi pasien dan riwayat pemeriksaan yang dienkripsi dapat dilihat pada Gambar 4.18 di bawah ini.



Gambar 4.18 Data informasi pasien dan riwayat pemeriksaan pasien yang telah dienkripsi pada bagian user

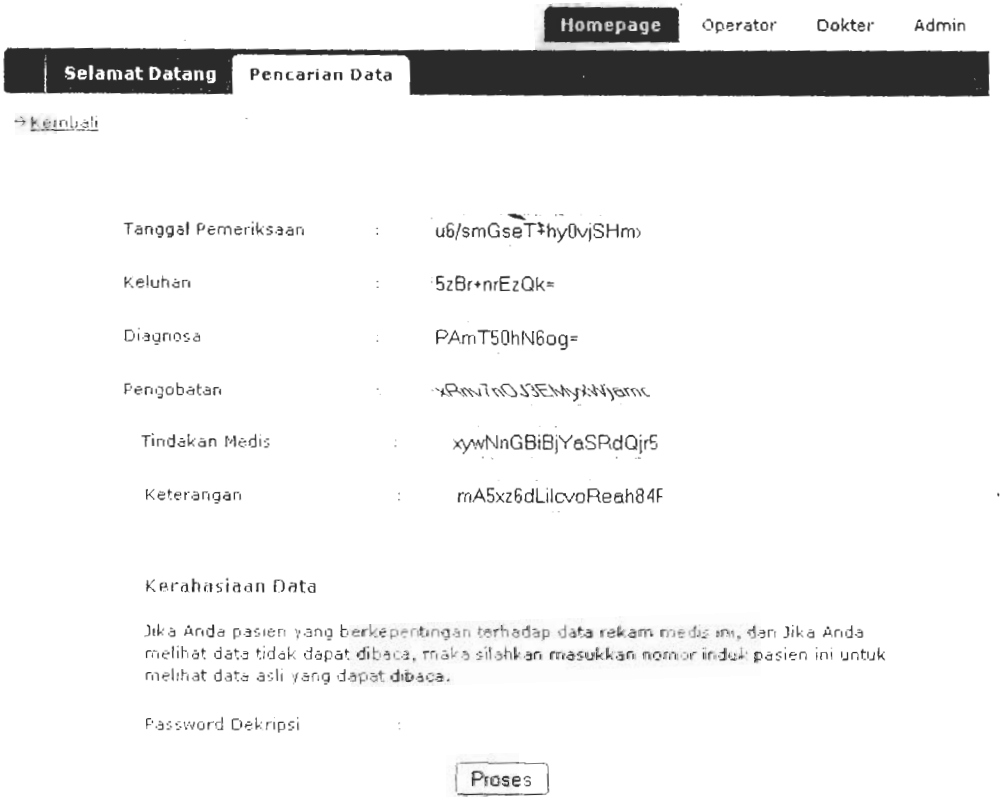
Pada gambar di atas dapat dilihat ada 2 buah data yang ditampilkan yaitu informasi pasien yang terenkripsi dan riwayat rekam medis yang terenkripsi. Pada data informasi pasien terdapat data informasi pasien yang telah dienkripsi dan ditampilkan sesuai dengan tag yang telah ditentukan sebelumnya pada proses enkripsi di bagian *operator*. Sedangkan pada bagian riwayat rekam medis ditampilkan data yang telah dienkripsi sebelumnya pada bagian dokter.

Untuk membuka atau mendekripsi data informasi pada bagian informasi pasien diperlukan kunci *password* yang sama dengan kunci *password* yang telah ditentukan oleh *operator*. Gambar 4.19 adalah tampilan dari data informasi pasien yang telah dilakukan proses dekripsi menggunakan kunci *password* yang sama.



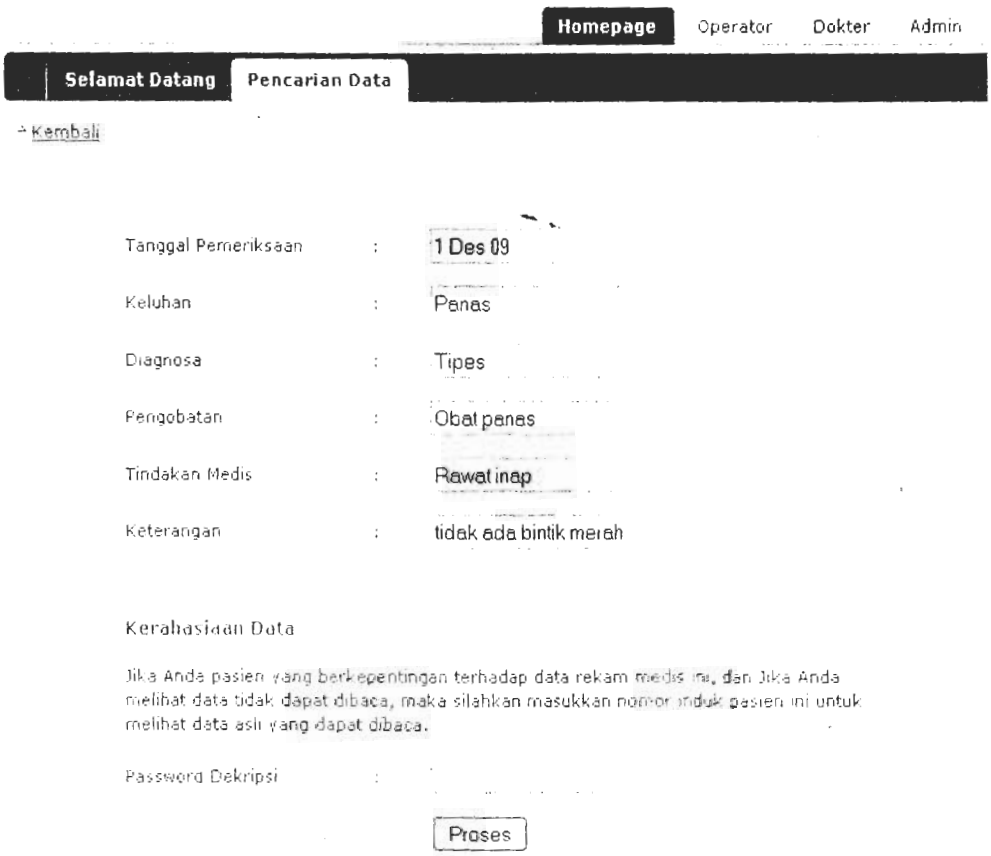
Gambar 4.19 Proses dekripsi informasi pasien pada bagian user

Pada riwayat rekam medis *user* dapat melihat data enkripsi secara lengkap dengan melihat detail pada menu yang digunakan. Sedangkan untuk mendekripsi riwayat rekam medis diperlukan kunci *password* yang sama dengan kunci *password* yang telah diberikan oleh dokter. Tampilan sebuah riwayat rekam medis yang dienkripsi dapat dilihat pada Gambar 4.20 di bawah ini.



Gambar 4.20 Data riwayat rekam medis yang telah dienkripsi pada bagian user

Untuk mendekripsi atau membuka data riwayat rekam medis yang telah dienkripsi dilakukan dengan memasukkan kunci *password* yang sama dengan kunci *password* yang telah diberikan oleh dokter. Setelah kunci *password* dimasukkan maka akan dapat dilihat data riwayat rekam medis seperti terlihat pada Gambar 4.21 di bawah ini.



Gambar 4.21 Hasil dekripsi riwayat rekam medis pada bagian user

4.3. Hasil Implementasi

Hasil implementasi penelitian ini adalah berupa *prototype* program web sebuah rumah sakit yang dilengkapi dengan proses enkripsi dan dekripsi pada tiap-tiap bagian. Proses enkripsi dan dekripsi menggunakan metode kriptografi dengan algoritma PBE with MD5 and DES.

Pada tahap ini akan dibahas mengenai hasil tampilan program secara keseluruhan dan beberapa file potongan program yang berhubungan dengan proses aplikasi program.

4.3.1. Implementasi Halaman User

Implementasi halaman user adalah tampilan *homepage* yang merupakan tampilan halaman web yang digunakan oleh *user*. Pada halaman ini memiliki 2 menu utama yaitu: menu Selamat Datang yang berisi informasi umum tentang rumah sakit dan menu Pencarian Data yang menampilkan form untuk dipergunakan oleh user untuk memasukkan nomor identitas untuk mengetahui informasi umum pasien dan informasi riwayat rekam medis dirinya.

4.3.1.1. Implementasi Halaman Main Page

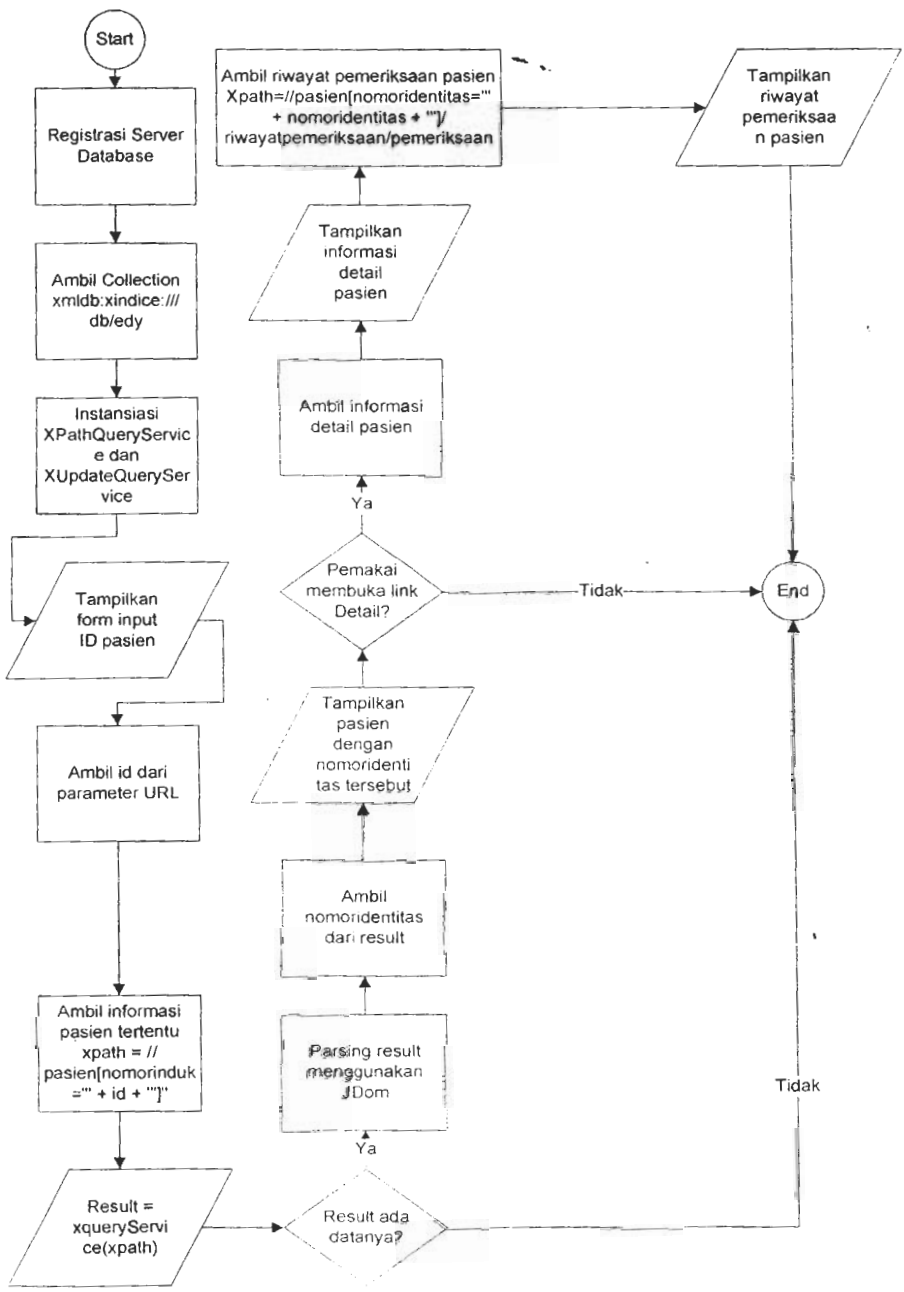
Halaman *main page* merupakan halaman yang pertama kali muncul pada tampilan web untuk *user*. Pada halaman ini berisi tentang informasi umum yang berupa informasi perkembangan teknologi kedokteran yang dimiliki oleh rumah sakit. Tampilan halaman main page dapat dilihat seperti Gambar 4.22 dibawah ini:

Gambar 4.22 Implementasi Halaman *Main Page*

4.3.1.2. Implementasi Halaman Pencarian Data

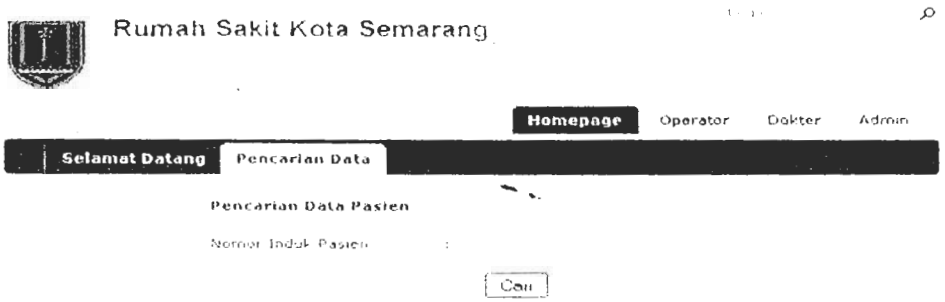
Halaman Pencarian Data digunakan oleh user atau pasien untuk memasukkan nomor identitas pasien yang dimiliki pada form sehingga *user* dapat

mengetahui informasi data pasien dan data rekam medisnya. Untuk mengetahui urutan program pada implementasi pencarian data dapat dilihat pada flowchart gambar 4.23 :



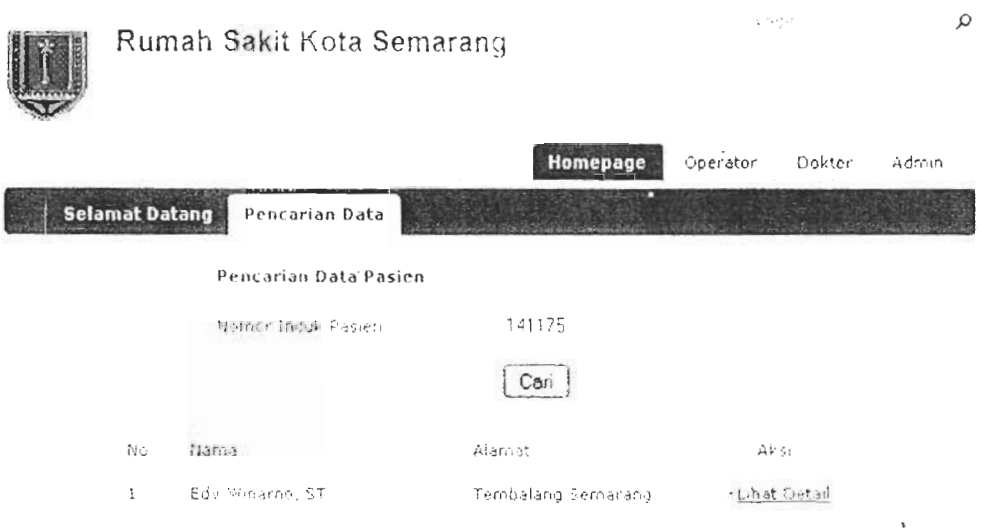
Gambar 4.23 Diagram alir program pencarian data

Form pengisian nomor identitas pasien dapat dilihat seperti Gambar 4.24 di bawah ini :



Gambar 4.24 Implementasi Halaman Pencarian Data

Setelah *user* memasukkan nomor identitas pasien maka akan muncul tampilan seperti pada Gambar 4.25. di bawah ini :



Gambar 4.25 Implementasi Pencarian Data setelah user memasukkan Nomor Identitas

Setelah *user* memasukkan nomor identitas dan terlihat tampilan Gambar 4.25 diatas, kemudian user akan dapat melihat tampilan data Informasi Pasien dan

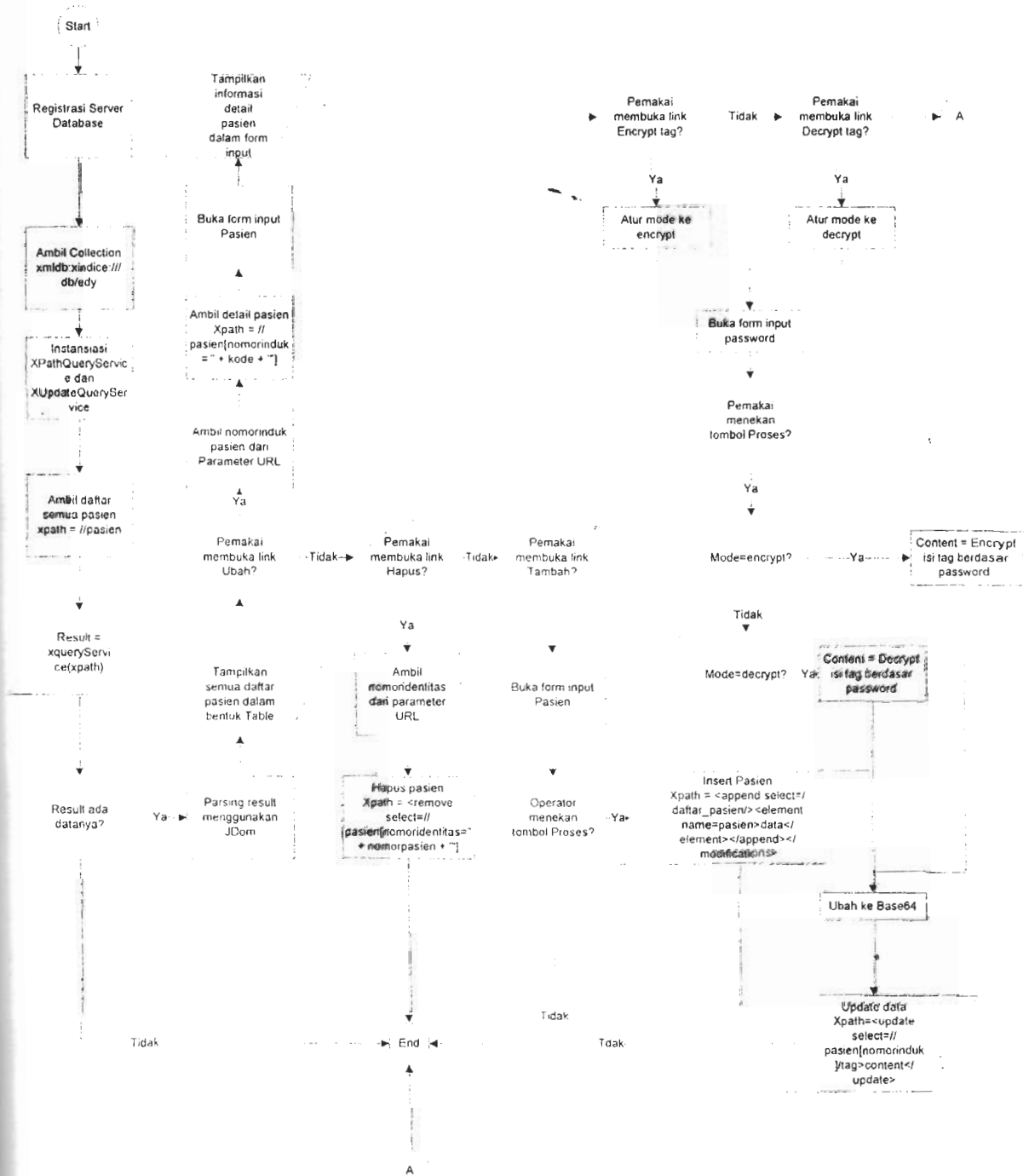
Riwayat Rekam Medis secara lengkap seperti terlihat pada Gambar 4.26 di bawah ini :

Gambar 4.26 Implementasi Pencarian Data untuk melihat detail Informasi Pasien dan Riwayat Rekam Medis

4.3.2. Implementasi Halaman *Operator*

Implementasi halaman *operator* adalah tampilan halaman web yang digunakan oleh operator untuk melakukan proses input data pasien untuk melakukan penambahan data, penghapusan data, *update* data dan melakukan proses enkripsi dan dekripsi dari data pasien yang diperlukan. *Operator* baru bisa melakukan semua proses itu setelah *operator* melakukan proses *login* dengan memasukkan *user name* dan *password*-nya.

Untuk mengetahui urutan langkah program implementasi pada halaman operator dapat dilihat pada *flowchart* di Gambar 4.27 berikut ini:

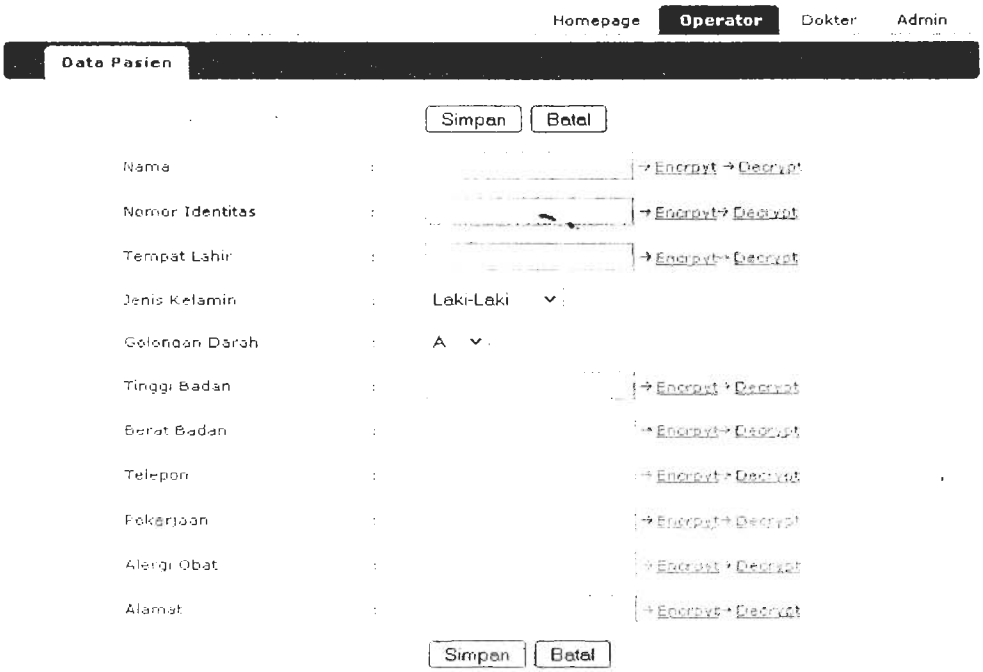


Gambar 4.27 Diagram alir Program Implementasi Halaman *Operator*

Gambar tampilan implementasinya seperti terlihat pada Gambar 4.28 di bawah ini :

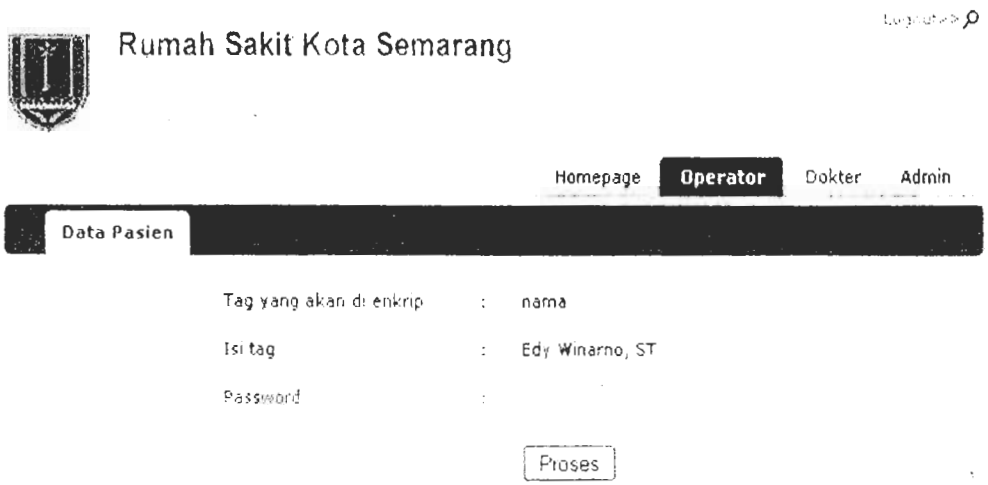
Gambar 4.28 Implementasi Halaman Operator

Pada halaman implementasi ini, *operator* dapat memasukkan data informasi pasien, menyimpan data informasi pasien, serta dapat juga melakukan proses enkripsi dan dekripsi dari data pasien yang diperlukan. Tampilan *form* implementasinya dapat dilihat pada Gambar 4.29 di bawah ini:



Gambar 4.29 Form Isian untuk pengisian Data Informasi Pasien

Pada implementasi halaman *operator* ini juga menampilkan *form* untuk proses enkripsi dan dekripsi data sesuai dengan *tag* yang dipilih. Proses enkripsi dan dekripsi ini menggunakan kunci password yang ditentukan oleh *operator*. Gambar implementasi tampilan program dapat dilihat pada Gambar 4.30 di bawah ini :

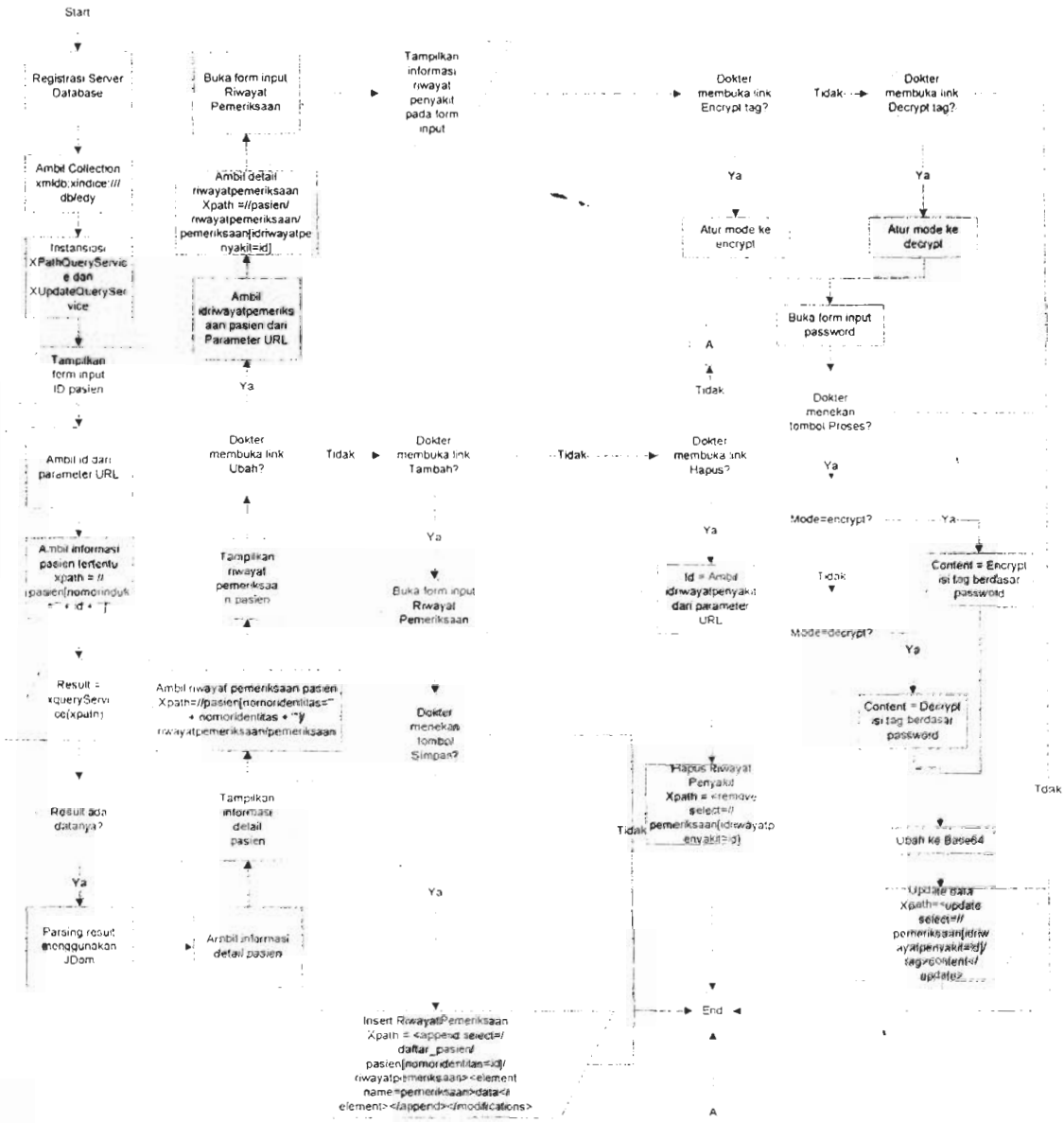


Gambar 4.30 Halaman *operator* untuk menampilkan *form* pada proses enkripsi dan dekripsi data sesuai dengan *tag* yang dipilih

4.3.3. Implementasi Halaman Dokter

Implementasi halaman dokter adalah tampilan halaman web yang digunakan oleh dokter untuk melakukan proses input data pemeriksaan pasien untuk melakukan penambahan data pemeriksaan, penghapusan data pemeriksaan, *update* data pemeriksaan dan melakukan proses enkripsi dan dekripsi dari data pemeriksaan pasien yang diperlukan. Dokter baru bisa melakukan semua proses itu setelah dokter melakukan proses *login* dengan memasukkan *user name* dan *password*-nya.

Untuk mengetahui urutan langkah program implementasi pada halaman dokter dapat dilihat pada *flowchart* di Gambar 4.31 berikut ini:



Gambar 4.31 Diagram alir program implementasi halaman dokter

Gambar tampilan implementasinya seperti terlihat pada Gambar 4.32 di bawah ini :

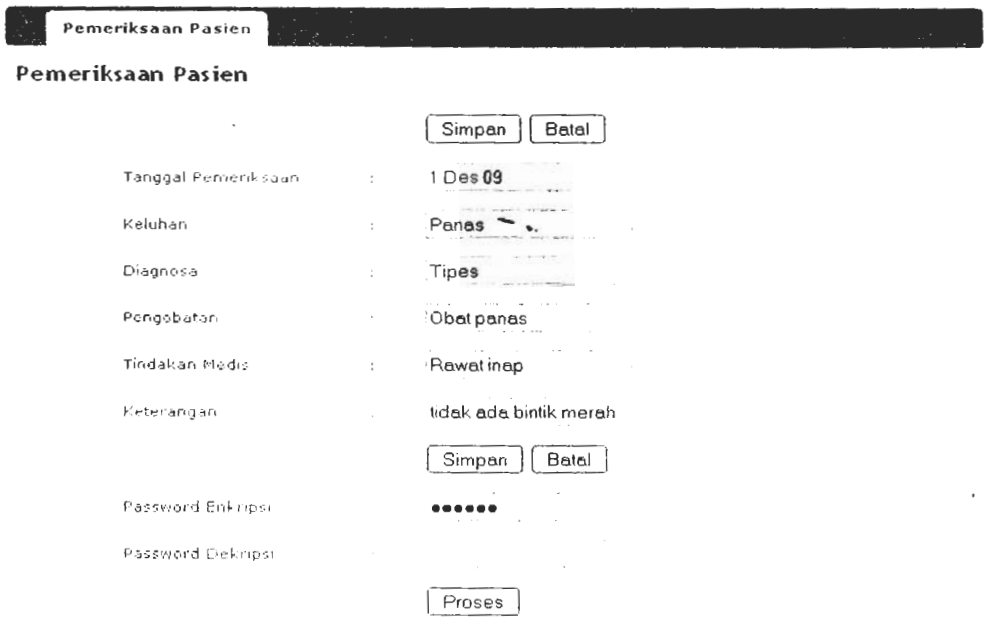
Gambar 4.32 Implementasi Halaman Dokter

Dokter harus memasukkan nomor identitas pasien untuk bisa masuk ke data pemeriksaan pasien. Setelah itu dokter dapat mengakses data informasi pasien seperti yang terlihat pada Gambar 4.33 di bawah ini :

Pemeriksaan Pasien

Gambar 4.33 Pemeriksaan pasien pada halaman dokter

Dokter tidak dapat mengubah data informasi pasien yang telah dibuat oleh *operator*. Dokter hanya bisa mengubah data pemeriksaan pasien, menambah dan menghapus, serta melakukan proses enkripsi pada data pemeriksaan. Contoh pengisian pada *form* tentang pengisian data pemeriksaan pasien dapat dilihat pada Gambar 4.34 di bawah ini:

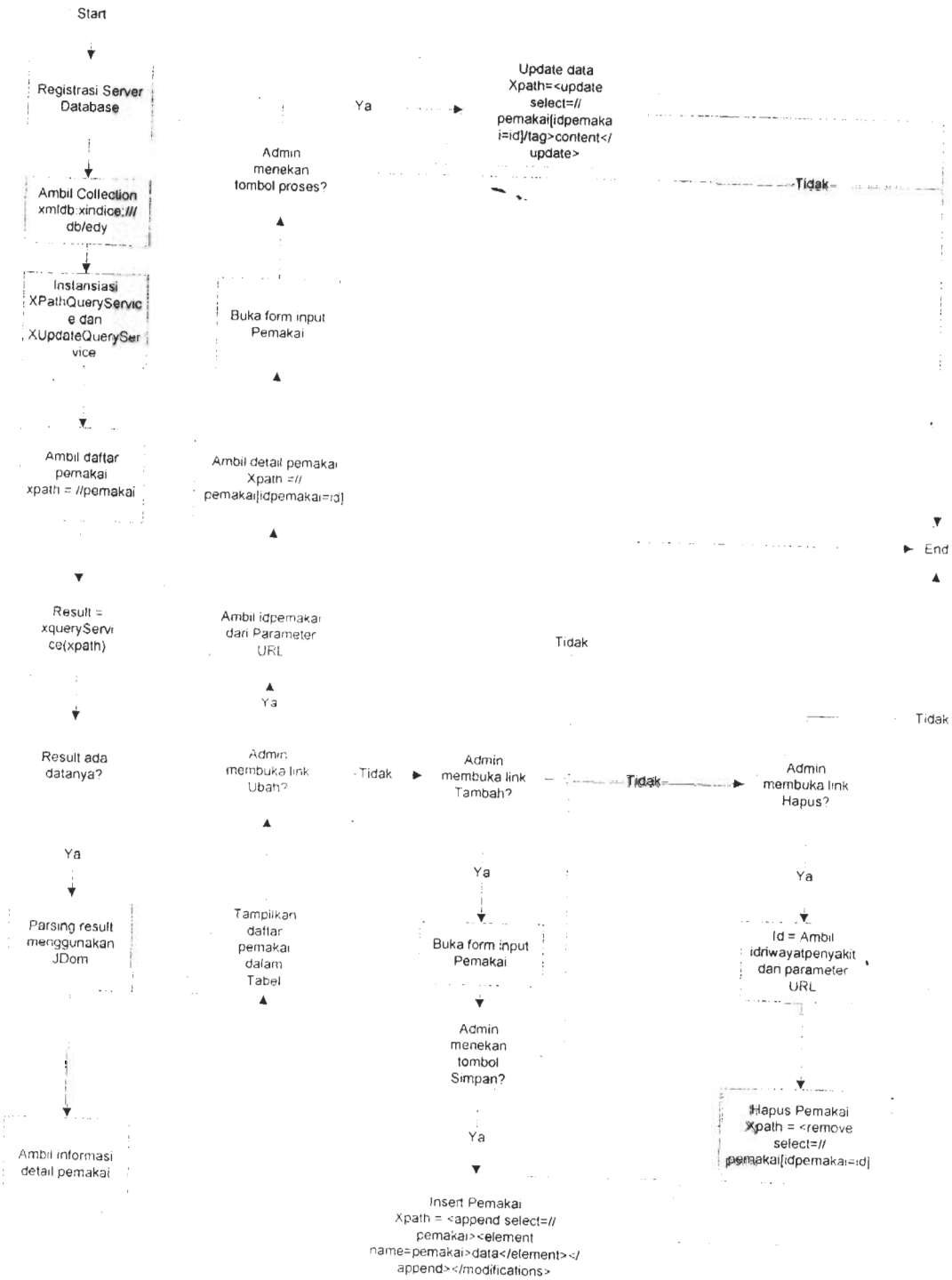


Gambar 4.34 *Form* pengisian data pemeriksaan pasien pada halaman dokter

4.3.4. Implementasi Halaman *Administrator*

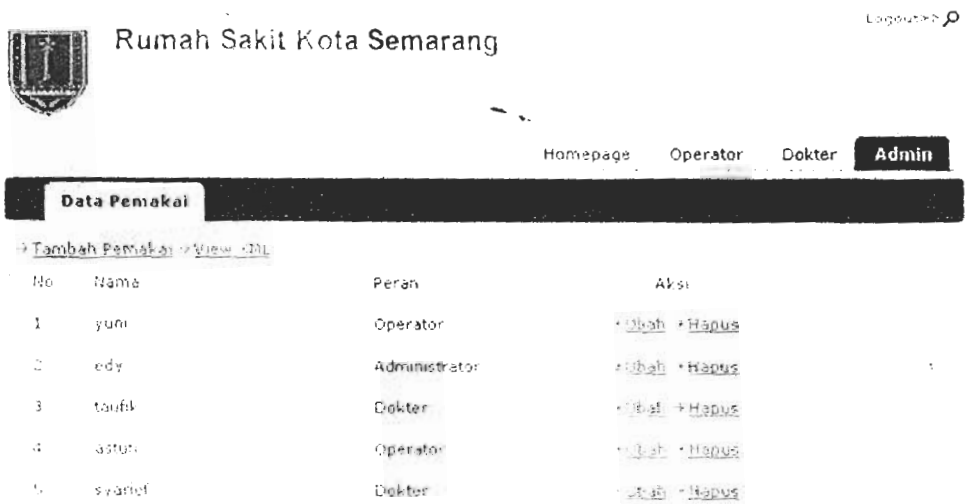
Implementasi halaman *Administrator* adalah tampilan halaman web yang digunakan oleh *Administrator* untuk melakukan proses input data pemakai untuk mengatur siapa saja yang berhak mengakses web ini. Ada 3 jenis pemakai yang digunakan yaitu *Administrator*, *Operator* dan Dokter. Ketiganya memiliki *user name*, *password* dan peran masing-masing untuk bisa mengakses tiap-tiap bagiannya. Pada bagian ini hanya *Administrator* yang behak mengubah, menghapus dan meng-*update* data pemakai. *Administrator* baru bisa melakukan semua proses itu setelah *Administrator* melakukan proses *login* dengan memasukkan *user name* dan *password*-nya.

Untuk mengetahui urutan langkah program implementasi pada halaman *administrator* dapat dilihat pada *flowchart* di Gambar 4.35 berikut ini:



Gambar 4.35 Diagram alir program implementasi halaman administrator

Gambar tampilan implementasinya seperti terlihat pada Gambar 4.36 di bawah ini :



Gambar 4.36 Implementasi Halaman *Administrator*

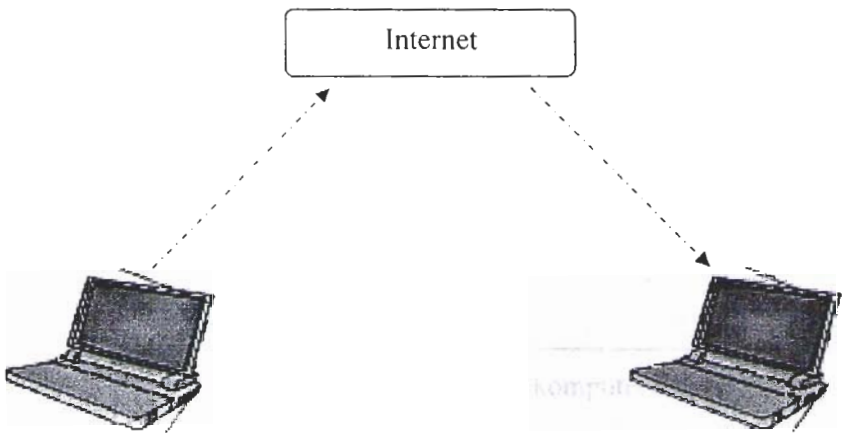
Untuk mengubah data pemakai pada implementasi halaman administrator ini dapat dilakukan oleh *administrator* dengan memasukkan data informasi pemakai yaitu berupa nama, *password* dan peran masing-masing pemakai. Implementasi form pengisian pada halaman ini dapat dilihat pada Gambar 4.37 di bawah ini:

Gambar 4.37 Form pengisian data pemakai pada implementasi halaman *administrator*

4.4. Pengujian dan Analisa Hasil Implementasi

4.4.1. Konfigurasi Pengujian

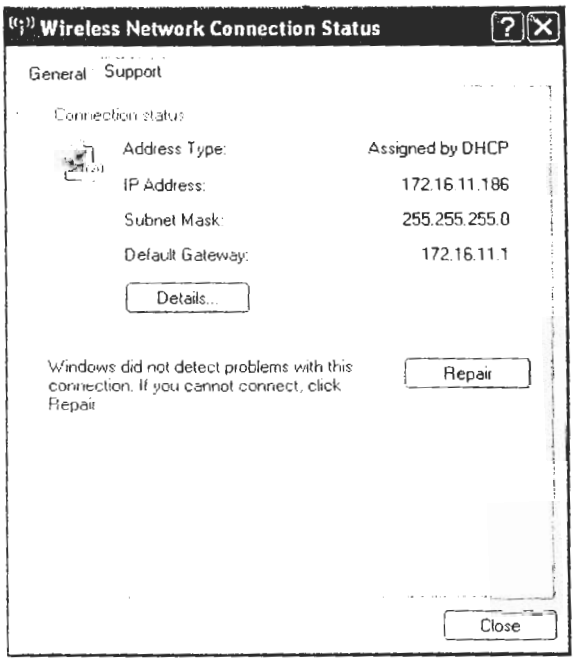
Pada penelitian ini dilakukan pengujian menggunakan metode *client* dan *server* pada sebuah jaringan komputer untuk mengetahui apakah implementasi yang dibuat dapat diterapkan pada kondisi yang sebenarnya. Pengujian dilakukan menggunakan 2 buah komputer yang terhubung dengan *internet* dengan model jaringan *wireless*, komputer pertama digunakan sebagai *server* dan komputer yang kedua digunakan sebagai *client*. Gambar 4.38 menunjukkan hubungan antara 2 buah komputer yang digunakan pada proses pengujian ini.



Gambar 4.38 Hubungan dua buah komputer *client* dan *server* pada pada proses pengujian implementasi

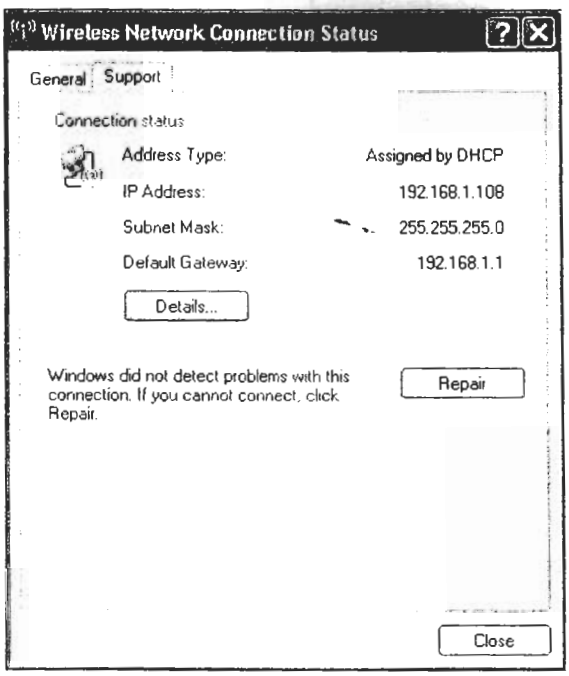
Komputer server yang digunakan memiliki IP 172.16.11.186, komputer ini digunakan sebagai *server* untuk menampilkan seluruh implementasi program. Sedangkan komputer client dengan IP 192.168.1.108 digunakan sebagai *client* yang berfungsi sebagai *browser* untuk melihat dan menguji hasil implementasi program. Kedua komputer ini terhubung dalam jaringan internet dengan komunikasi menggunakan metode *wireless*.

Pada proses ini dilakukan pengujian terhadap aplikasi penerapan kriptografi menggunakan *Password Based Encryption* untuk keamanan distribusi data menggunakan XML *Database* Xindice pada sebuah rumah sakit. Gambar 4.39 menunjukkan alamat IP pada komputer *server*.



Gambar 4.39 Alamat IP pada komputer *server*

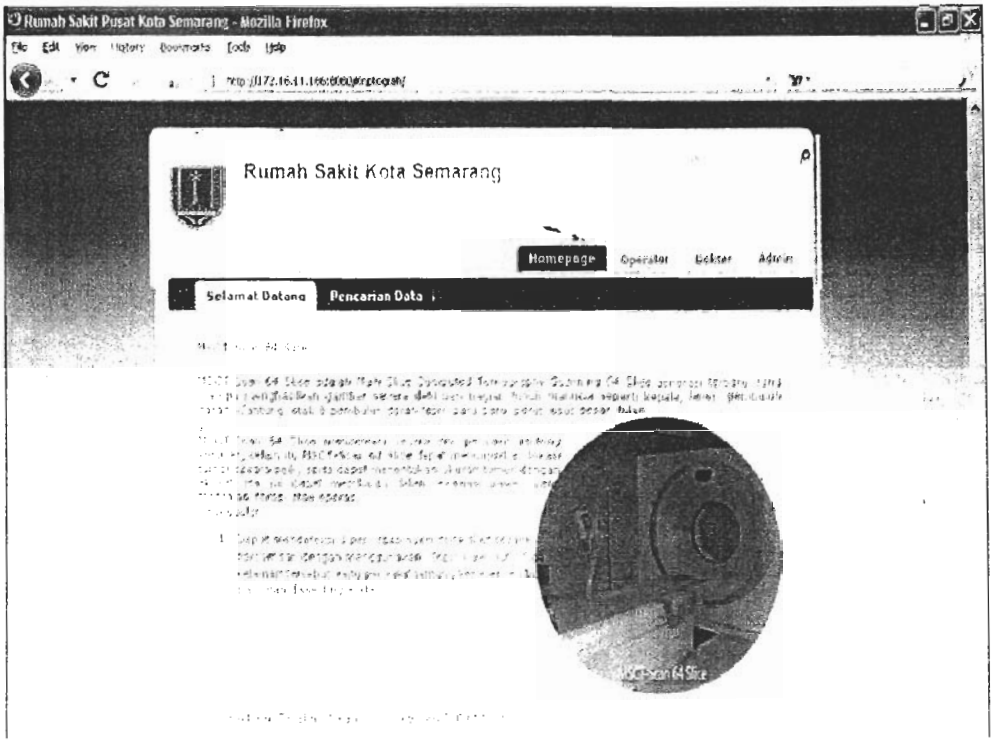
Gambar 4.40 menunjukkan alamat IP pada komputer *client*.



Gambar 4.40 Alamat IP pada komputer *client*

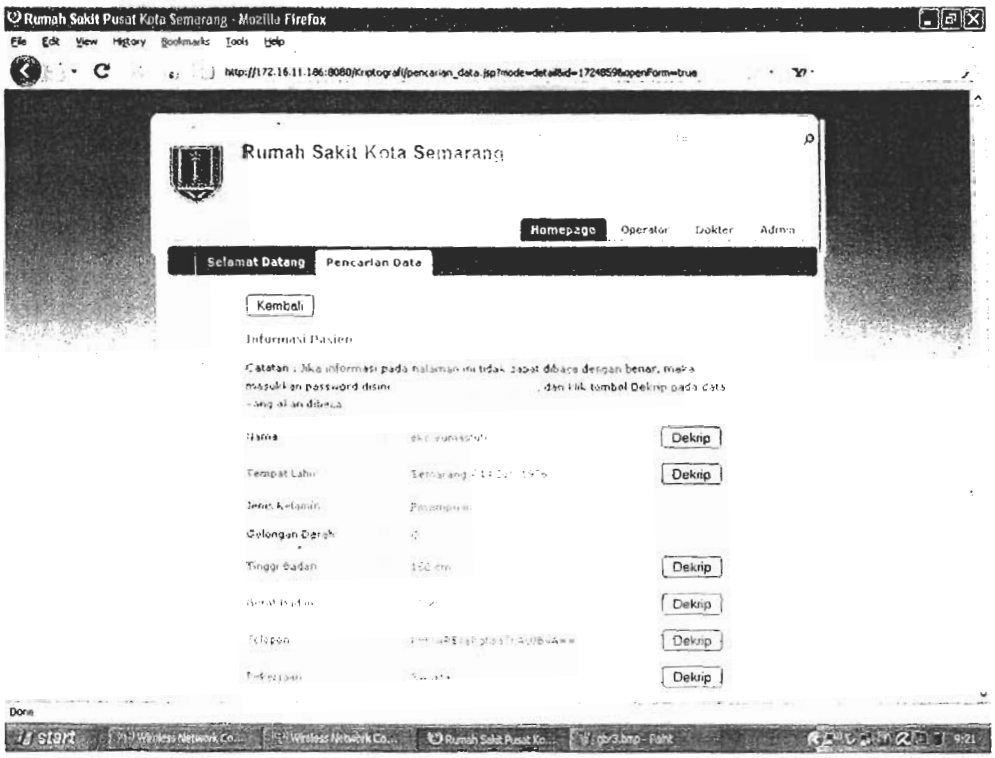
4.4.2. Hasil Pengujian

Hasil pengujian dapat dilakukan pada tiap-tiap halaman yang ada pada tampilan *web*, yaitu pada halaman *user*, halaman *operator*, halaman dokter dan pada halaman *administrator*. Pada hasil pengujian ini diperlihatkan contoh pengujian pada halaman *user* yang dilakukan pada sebuah komputer *client*, menggunakan *Mozilla Firefox* yang digunakan sebagai *browser*. Gambar 4.41 menunjukkan tampilan *web* yang diakses melalui *browser* pada komputer *client*.



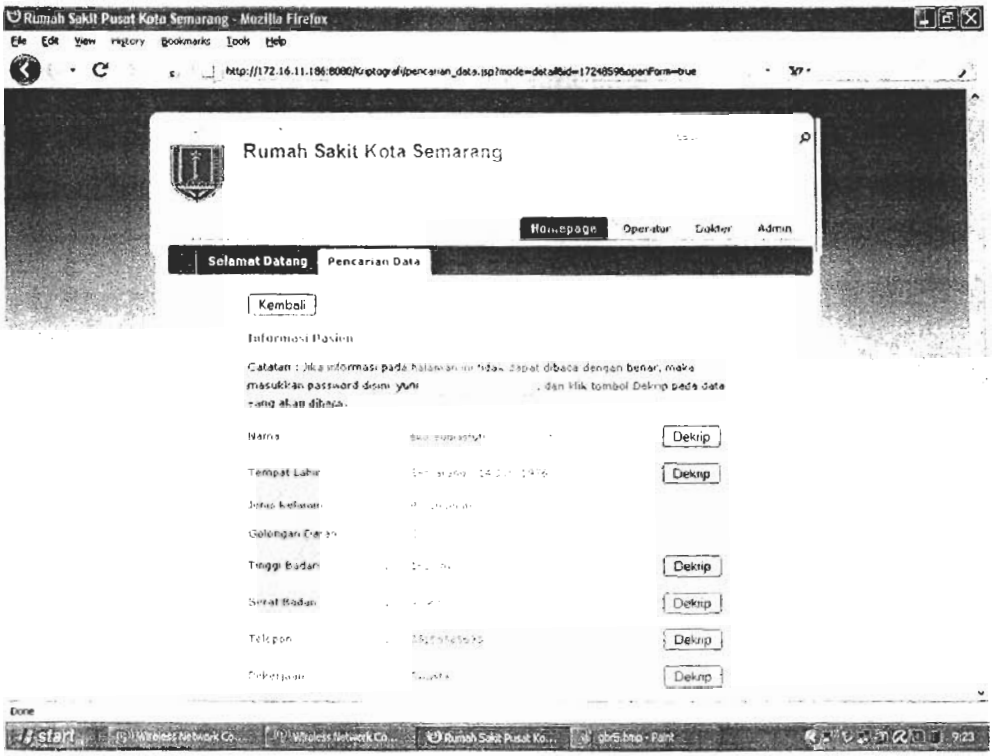
Gambar 4.41 Tampilan *web* yang diakses melalui *browser* pada komputer *client*.

Pada komputer *client* ini seorang *user* dapat mengakses seluruh program pada tampilan dan dapat melakukan proses dekripsi untuk mengetahui data riwayat rekam medis dari user yang telah dienkripsi. Pada gambar 4.42 merupakan sebuah tampilan pada *browser* yang menampilkan sebuah data *user* yang telah dienkripsi. Dengan menggunakan kunci *password* yang telah diberikan, *user* dapat melakukan proses dekripsi pada data sesuai dengan data yang akan didekripsi. Contoh pada gambar 4.42 menunjukkan sebuah data yang telah dienkripsi yaitu pada *tag* Telepon.



Gambar 4.42 Tampilan data pada *browser* di komputer client sebelum proses dekripsi

Pada gambar 4.43 menunjukkan hasil dari proses dekripsi yang telah dilakukan, yaitu dengan memasukkan kunci *password* 'yuni' sehingga pada *tag* telepon dapat diperlihatkan data sebenarnya setelah proses dekripsi dilakukan.



Gambar 4.43 Tampilan data pada *browser* di komputer client setelah proses dekripsi.

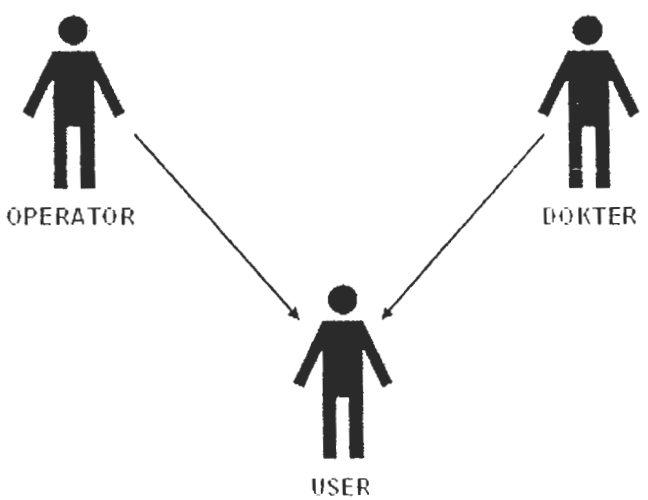
4.4.3 Analisa Sistem Keamanan Pada Implementasi Web

Implementasi yang dibuat merupakan sebuah web rumah sakit yang merupakan sebuah sistem informasi pelayanan rumah sakit terhadap kerahasiaan data rekam medis pasien. Implementasi ini dilengkapi dengan menu enkripsi dan dekripsi untuk menyembunyikan atau merahasiakan data. Data rekam medis seorang pasien menjadi sangat penting untuk dijaga kerahasiaannya yaitu untuk melindungi data rekam medis pasien dari pihak yang tidak berwenang untuk mengaksesnya. Kerahasiaan data dapat dibaca dengan memasukkan *password* yang telah disepakati sebelumnya. Data chiperteks hasil enkripsi dapat didekripsi menjadi sebuah data plainteks kembali dengan menggunakan kunci *password* tersebut.

Seorang *operator* dapat memasukkan data rekam medis pasien dan dapat melakukan enkripsi data sesuai yang diperlukan. Oleh dokter, data tersebut

kemudian diproses untuk dilakukan penambahan data riwayat rekam medis pasien sesuai hasil pemeriksaan. Dokter tidak dapat membaca seluruh data yang telah dienkripsi oleh *operator*, karena dokter tidak memiliki kunci *password* untuk mendekripsi data dari *operator*. Dokter memiliki hak untuk melakukan proses enkripsi pada data riwayat rekam medis pasien yang diperiksanya. Kunci *password* yang digunakan dokter untuk melakukan proses enkripsi ini berbeda dengan kunci *password* yang diberikan oleh *operator* saat melakukan proses enkripsi saat memasukkan data pasien. Demikian juga sebaliknya, *operator* tidak memilik hak untuk mengakses data pasien pada bagian yang diisikan oleh dokter. *Operator* tidak mempunyai kunci *password* yang dibuat oleh dokter. Satu-satunya yang dapat mengakses untuk mendekripsi data yang diisikan oleh pihak *operator* dan dokter adalah pasien atau *user*.

Pasien atau *user* mendapatkan 2 buah kunci *password* yang masing – masing diberikan oleh pihak *operator* dan pihak dokter. Kunci *password* pertama diberikan oleh *operator* untuk mendekripsi data yang telah dienkripsi oleh pihak *operator*, dan kunci *password* yang kedua diberikan oleh dokter untuk mendekripsi data yang telah dienkripsi oleh pihak dokter. Hubungan distribusi pembagian kunci dapat dilihat pada gambar 4.44 di bawah ini.



Gambar 4.44 Hubungan distribusi pembagian kunci *password*

Dari sistem distribusi kunci *password* diatas dapat dilihat bahwa hanya *user* yang memiliki 2 buah kunci *password* untuk digunakan pada proses dekripsi

data yang telah dienkripsi oleh pihak *operator* ataupun pihak dokter. Pihak *operator* tidak mempunyai kunci *password* dari pihak dokter, sehingga *operator* tidak bisa melakukan proses dekripsi pada data yang telah dienkripsi oleh dokter. Demikian pula pihak dokter tidak bisa melakukan proses dekripsi pada data yang telah dienkripsi oleh pihak *operator*, sehingga pihak dokter tidak dapat melakukan proses dekripsi pada data yang telah dienkripsi oleh pihak *operator*.

Dari segi keamanan distribusi data, sistem ini bisa diaplikasikan untuk menyimpan dan menyembunyikan data-data rahasia dari seorang pasien pada sebuah rumah sakit agar tidak dapat diakses oleh pihak yang tidak berkepentingan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian dan pembahasan yang telah dilakukan dapat diambil kesimpulan sebagai berikut.

1. XML *database* Xindice dapat digunakan sebagai sebuah sistem basis data khusus untuk menyimpan, mengolah dan mengakomodasi data-data XML yang memiliki struktur data tertentu.
2. Pemakaian XML *Database* Xindice akan lebih memudahkan dalam pengolahan data aplikasi yang berupa data XML dibandingkan dengan menggunakan *relational database* (RDBMS)
3. Dengan digunakannya XML *Database* Xindice maka aplikasi tidak perlu melakukan pemetaan dari data XML ke struktur data tertentu. Aplikasi dapat langsung menyimpan data XML, dan mendapatkan kembali data XML dalam format yang tidak memerlukan perubahan.
4. Aplikasi penerapan kriptografi dapat dilakukan pada dokumen yang memiliki format data XML yaitu dengan cara melakukan proses enkripsi dan dekripsi pada tag / bagian tertentu sesuai yang diinginkan.

5.2 Saran

Dari hasil penelitian yang telah dilakukan maka penulis dapat memberikan beberapa saran untuk penelitian berikutnya, yaitu:

1. Membuat perbandingan antara XML *database* Xindice dengan sistem basis data yang lain dan mengimplementasikan pada kasus yang lebih luas.
2. Membuat implementasi program yang lebih kompleks dan variatif untuk mencakup sistem distribusi data yang lebih besar, dan tidak hanya digunakan pada distribusi data rekam medis di sebuah rumah sakit tapi

juga bisa dikembangkan pada instansi-instansi dan lembaga yang lain yang membutuhkan.

3. Sistem aplikasi dapat dikembangkan menjadi sebuah sistem yang memiliki kehandalan dalam sistem keamanan jaringan, sehingga distribusi data akan lebih aman dari serangan *hacker*.

DAFTAR PUSTAKA

Abdul Kadir, 2004, Dasar Pemrograman Web Dinamis Dengan *Java Server Page* (JSP). Penerbit Andi, Yogyakarta.

Bruce S., 1996, *Applied Cryptography,Protocols,Algorithms,and Source Code in C*. John Wilwey & Sons, Inc

David Hook, 2005, *Beginning Cryptography With Java*. Wiley Publishing,Inc, Indianapolis, USA

Hartono, 2003, Pemakaian Kriptografi Kunci Publik Denagn Algoritma RSA Untuk Keamanan Data XML. Universitas Gadjah Mada Yogyakarta.

John E.S., alih bahasa: Dwi Prabantini, 2002, *Just XML*. Penerbit Andi, Yogyakarta.

Noprianto, 2004, Mengenal XML. Info Linux 12/2004

Stallings W.,1999, *Cryptography and Network Security Principles and Practice second edition*.Prentice Hall, New Jersey, USA

Theodore W. Leung, 2004, *Professional XML Development with Apache Tools: Xerces, Xalan, FOP, Cocoon, Axis, Xindice*, Wrox Press, Wiley Publishing,Inc, Indianapolis, USA

Umniati, Mukodim, 2002, Perbandingan Algoritma dalam Enkripsi antara *Conventional Cryptosystems* dan *Public Key Cryptosystems. Proceeding*, Komputer dan Sistem Intelijen (KOMMIT 2002)

Wrox Books,2002,"*Profesional XML Databases*", *http"//tutorials.freeskills.com/read/category/81/id/137*

CURRICULUM VITAE

1. Ketua Peneliti:

Nama	: Edy Winarno, S.T.,M.Eng.
NIP/NIK	: YU.2.04.10.071
Tempat dan Tanggal lahir	: Salatiga, 15 Nopember 1975
Jenis Kelamin	: Laki-laki
Status Perkawinan	: Kawin
Agama	: Islam
Golongan/Pangkat	: III-A / Penata Muda
Jabatan Akademik	: Asisten Ahli
Perguruan Tinggi	: Universitas Stikubank (UNISBANK) Semarang
Alamat Kantor	: Kampus Mugas: JL. Tri Lomba Juang No.1 Semarang
	Kampus Kendeng : JL. Kendeng V Bendan Ngisor Semarang
Telp	: 024-8311668 ;024-8414970
Fax	: 024-8443240 ; 024-8441738
Alamat Rumah	: Perum Klipang Pesona Asri III E-46 Tembalang Semarang
Telp	: 085865206752
Alamat e-mail	: winarnoedy@gmail.com

a. Pendidikan:

2009 S-2 Master of Engineering (M.Eng.) dari S2 Sistem Komputer dan Informatika Teknik Elektro Universitas Gadjah Mada Yogyakarta.

2001 S-1 Teknik Elektro Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang

b. Pengalaman Penelitian

2008 Analisis Model Akuisisi Data Terhadap Piranti Analog to Digital (ADC) 0804 UNISBANK.

2007 Teknik Instrumentasi Pemrograman Kendali Terapan Mikro Menggunakan Visual Basic 6.0 UNISBANK

2007 Karakteristik Umum antar muka Sistem Kendali berbasis MCS51 sebagai Terminal Masukan Rangkaian Transduser UNISBANK

2005 Aplikasi PLC Pada Sistem Kontrol Lampu Lalu Lintas Digital Menggunakan SYSWIN 3.4 UNISBANK

c. Pengalaman Publikasi:

1. Pemrograman Basis Data Menggunakan XML Database Xindice, Jurnal Electrika, ISSN 2085- Oktober 2009
2. Analog to Digital Converter sebagai Antarmuka Sistem – DINAMIKA TEKNIK – Jurnal Pengembangan Ilmu-ilmu Teknik Volume II no.1. ISSN:1412-3355. Januari 2008
3. Aplikasi PLC (Programmable Logic Controller) pada Sistem Lampu lalu-lintas Digital menggunakan program SYSWIN 3.4-DINAMIKA TEKNIK- Jurnal Pengembangan Ilmu-ilmu Teknik Vol 1 no.1 Januari 2007.

Saya menyatakan bahwa semua keterangan dalam daftar riwayat hidup ini adalah benar dan apabila terdapat kesalahan, saya bersedia mempertanggungjawabkannya.

Semarang, 12 Oktober 2010

Yang menyatakan,

Edy Winarno, S.T., M.Eng.

2. Anggota Peneliti:

Nama	: Veronica Lusiana, ST, M.Kom
NIP/NIK	: YU.2.02.10.050
Tempat dan Tanggal lahir	: Semarang, 3 April 1976
Jenis Kelamin	: Perempuan
Status Perkawinan	: Kawin
Agama	: Katholik
Golongan/Pangkat	: III-b / Penata Muda
Jabatan Akademik	: Asisten Ahli
Perguruan Tinggi	: Universitas Stikubank (UNISBANK) Semarang
Alamat Kantor	: Kampus Mugas: JL. Tri Lomba Juang No.1 Semarang
	Kampus Kendeng : JL. Kendeng V Bendan Ngisor Semarang
Telp	: 024-8311668 ;024-8414970
Fax	: 024-8443240 ; 024-8441738
Alamat Rumah	: Jl.Tejokusumo II/ 26 Tlogosari Semarang
Telp	: 08882424605
Alamat e-mail	: verolusiana@yahoo.com

a. Pendidikan:

2009 S-2 Magister Sistem Informasi Universitas Diponegoro Semarang

2000 S-1 Teknik Elektro Universitas Semarang

b. Pengalaman Penelitian

2002 Pendeteksi dan Penghitung Detak Jantung UNISBANK

2003 Simulasi dan Desain Sistem Pengaturan Kecepatan Motor DC dengan Metode FeedForward Command dan Integral Feedback Control

2005 Kendali Sistem Terpadu Dengan Metode Octal Bus Transceiver With Non-Inverting 3 State Output

2008 Sistem Informasi Promosi Batik Tradisional Semarang berbasis WEB

c. Pengalaman Publikasi:

1. Sistem Keamanan Internet Banking – DINAMIK TEKNIK- Jurnal Pengembangan Ilmu-Ilmu Teknik Volume III Januari 2009 ISSN:1412-3355.
2. Pemetaan Karnaugh untuk Penyederhanaan Rangkaian Logika dan Pemrogramannya Menggunakan Matlab- DINAMIK TEKNIK-Jurnal Pengembangan Ilmu-Ilmu Teknik Volume 1 no.1 Januari 2007 ISSN:1412-3355
3. Menentukan Posisi Di Permukaan Bumi Dengan GPS (Global Positioning System)- DINAMIKA TEKNIK – Jurnal Pengembangan Ilmu-Ilmu Teknik Volume II No.2 Mei 2003

Saya menyatakan bahwa semua keterangan dalam daftar riwayat hidup ini adalah benar dan apabila terdapat kesalahan, saya bersedia mempertanggungjawabkannya.

Semarang, 17 Maret 2010

Yang menyatakan,

Veronica Lusiana, ST,M.Kom

3. Anggota Peneliti:

Nama	: Wiwien Hadikurniawati, ST
NIP/NIK	: YU.2.02.10.051
Tempat dan Tanggal lahir	: Semarang, 16 Maret 1976
Jenis Kelamin	: Perempuan
Status Perkawinan	: Kawin
Agama	: Islam
Golongan/Pangkat	: III-b / Penata Muda
Jabatan Akademik	: Asisten Ahli
Perguruan Tinggi	: Universitas Stikubank (UNISBANK) Semarang
Alamat Kantor	: Kampus Mugas: JL. Tri Lomba Juang No.1 Semarang
	Kampus Kendeng : JL. Kendeng V Bendan Ngisor Semarang
Telp	: 024-8311668 ;024-8414970
Fax	: 024-8443240 ; 024-8441738
Alamat Rumah	: Jl. Banjarsari Raya 10 C Tembalang Semarang
Telp	: 08156514321
Alamat e-mail	: wien_hadikurniawati@yahoo.co.id

a. Pendidikan:

2009 S-2 Magister Sistem Informasi Universitas Diponegoro Semarang,

1999 S-1 Teknik Elektro Universitas Semarang

b. Pengalaman Penelitian

2002 Pendeteksi dan Penghitung Detak Jantung UNISBANK

2003 Perancangan alat Stimulasi Otot dan Syaraf (Electical Stimulator)

2004 Aplikasi PLC Pada Sistem Kontrol Lampu Lalu Lintas Digital Menggunakan SYSWIN 3.4

2005 Sistem Kendali Robot Mobil dengan Menggunakan IC Mikrokontroler AT89S52

2008 Sistem Pendukung keputusan Manajemen Sumber Daya Manusia pada AJB Bumi Putera Kendal.

c. Pengalaman Publikasi:

1. Teknik Kriptografi pada Transponder Radio Frequency Identification – Jurnal Elektrika, ISSN 2085-0565 Volume I Nomor 1 – Maret 2009.
2. Sistem Keamanan Internet Banking – DINAMIKA TEKNIK – Jurnal Pengembangan Ilmu-Ilmu Teknik Volume III Januari 2009 ISSN:1412-3355.
3. Keunggulan Power Mosfet dan IGBT sebagai Piranti Elektronika di Bidang Elektronika Daya – DINAMIKA TEKNIK – Jurnal Pengembangan Ilmu-Ilmu Teknik Volume II No.1 Januari 2008 ISSN 1412-3355.
4. Pengaturan Timer dan Waktu Tunda pada Sistem Timer dan Counter dalam MCS51 – DINAMIKA TEKNIK – Jurnal Pengembangan Ilmu-Ilmu Teknik Volume I No.1 Januari 2007 ISSN 1412-3355.
5. Perbandingan Prinsip Kerja Osiloskop Analog dengan Osiloskop Digital – DINAMIKA TEKNIK – Jurnal Pengembangan Ilmu-Ilmu Teknik Volume II No.2 Mei 2003 ISSN 1412-3339.

Saya menyatakan bahwa semua keterangan dalam daftar riwayat hidup ini adalah benar dan apabila terdapat kesalahan, saya bersedia mempertanggungjawabkannya.

Semarang, 17 Maret 2010

Yang menyatakan,

Wiwien Hadikurniawati, ST., M.Kom.