BAB 2
TINJAUAN PUSTAKA

2.1 Profil Perusahaan

Badan Geologi mempunyai unit Pusat Sumber Daya Air Tanah dan Geologi Lingkungan mempunyai tugas dan fungsi melaksanakan penelitian, penyelidikan, perekayasaan, pemodelan, serta pelayanan di bidang air tanah, geologi teknik dan geologi lingkungan. Pusat Sumber Daya Air Tanah dan Geologi Lingkungan menyelenggarakan fungsi [6]:

1. Penyiapan penyusunan kebijakan teknis, norma, standar, prosedur dan kriteria serta rencana dan program di bidang air tanah, geologi teknik, dan geologi lingkungan.
2. Pelaksanaan pemetaan, penelitian, penyelidikan, perekayasaan, pemodelan, dan bimbingan teknis, serta pemberian rekomendasi teknis di bidang air tanah, geologi teknik, dan geologi lingkungan.
3. Pemantauan, evaluasi dan pelaporan pelaksanaan pemetaan, penelitian, penyelidikan, perekayasaan, pemodelan di bidang air tanah, geologi teknik, dan geologi lingkungan.
4. Pelaksanaan administrasi Pusat Sumber Daya Air Tanah dan Geologi Lingkungan.

Pusat Sumber Daya Air Tanah dan Geologi Lingkungan terdiri atas:

a. Bagian Tata Usaha
b. Bidang Air Tanah
c. Bidang Geologi Lingkungan
d. Bidang Geologi Teknik
e. Kelompok Jabatan Fungsional.

2.1.1 Sejarah Instansi

Pusat Sumber Daya Air Tanah dan Geologi Lingkungan (PAG) adalah salah satu unit kerja di bawah Badan Geologi, Kementrian Energi dan Sumber Daya Mineral. Pusat Sumber Daya Air Tanah dan Geologi Lingkungan (PAG) mempunyai latar belakang sejarah sejak tahun 1978 dengan nama Direktorat

Geologi Tata Lingkungan. Tahun 2001 berubah nama menjadi Direktorat Tata Lingkungan Geologi dan Kawasan Pertambangan. Pada tahun 2005 berubah nama menjadi Pusat Lingkungan Geologi dengan perubahan tugas yang bertitik berat kearah penelitian dan pelayanan. Dan terakhir pada tahun 2010 berubah nama lagi menjadi Pusat Sumber Daya Air Tanah dan Geologi Lingkungan. [7]

2.1.2 Visi dan Misi Pusat Sumber Daya Air Tanah dan Geologi Lingkungan

Visi dan Misi dari Pusat Sumber Daya Air Tanah dan Geologi Lingkungan adalah sebagai berikut [8]:

2.1.2.1 Visi

Mewujudkan informasi geologi lingkungan, geologi teknik dan air tanah yang handal serta rujukan untuk rekomendasi penyusunan kebijakan dan pengaturan, pedoman dan prosedur penelitian implementasi pengelolaan tata ruang dan lingkungan. Mencapai rekomendasi geologi lingkungan, geologi teknik dan air tanah yang dipergunakan dalam setiap penyusunan tata ruang, pengelolaan lingkungan dan air tanah, rekonstruksi dan rehabilitasi lingkungan pasca bencana; Mencapai pelayanan informasi bidang geologi lingkungan, geologi teknik dan air tanah yang efektif, memuaskan pelanggan dan menjangkau masyarakat luas. Menjadikan sumber daya manusia dan organisasi sebagai rujukan terpercaya dan mitra terdepan dalam bidang penelitian dan pelayanan informasi geologi lingkungan, geologi teknik dan air tanah.

2.1.2.2 Misi

Menjadi institusi rujukan dalam bidang penelitian, penyelidikan, dan pemberian rekomendasi serta penyediaan, pengelolaan dan pelayanan yang terpercaya dalam mewujudkan pengelolaan air tanah, pengelolaan pengawasan pertambangan, penataan ruang, dan pengembangan wilayah berbasis geologi lingkungan, geologi lingkungan dan air tanah.

2.1.3 Logo Instansi

Logo Instansi dimana tempat penelitian dilaksanakan yaitu logo Kementrian Energi dan Sumber Daya Mineral dapat dilihat seperti pada gambar 2.1.



Gambar 2.1 Logo Kementrian Energi dan Sumber Daya Mineral (ESDM) [6]

Adapun arti/makna dari bentuk Logo Departemen Energi dan Sumber Daya Mineral yaitu :

a. Rangka segi lima menggambarkan falsafah Bangsa Indonesia, Pancasila.
b. Bulatan warna kuning menggambarkan dunia, di dalamnya terdapat 3 (tiga) garis melintang di bagian tengah dan berwarna hitam menggambarkan letak negara Indonesia secara geografis berada di tengah garis khatulistiwa yang melintang dari Barat ke Timur.
c. Tiga garis tebal warna hitam bergelombang yang terletak di bagian bawah bulatan dunia, menggambarkan lapisan bumi Indonesia yang mengandung sumber daya alam, mineral dan energi yang sangat potensial dan dikelola oleh Departemen Energi dan Sumber Daya Mineral untuk meningkatkan kesejahteraan Bangsa Indonesia.
d. Gambar palu dan belencong berwarna hitam yang melintang di depan menara merupakan lambang peralatan dasar eksplorasi mineral (bahan tambang).
e. Dua gambar kilatan warna kuning menggambarkan kilatan arus listrik yang merupakan energi sekunder.
f. Tulisan “ENERGI DAN SUMBER DAYA MINERAL” yang berwarna kuning di atas dasar hitam yang terletak di bawah bulatan dunia, yang ditulis pada garis khatulistiwa di ujung kanan menunjukkan nama Departemen yang memilki lambang tersebut.

Adapun arti/makna dari warna Logo Departemen Energi dan Sumber Daya Mineral yaitu :

a. Kuning merupakan lambang kejayaan, yang menggambarkan tugas yang diemban oleh Departemen Energi dan Sumber Daya Mineral dalam mengelola kekayaan sumber daya alam dan mineral yang sangat potensial di bumi Indonesia ini.
b. Hitam merupakan lambang keteguhan. Teguh dalam mengembangkan dan memanfaatkan sumber daya alam dan mineral untuk meningkatkan kesejahteraan bangsa dan negara.

2.1.4 Badan Hukum Instansi

Pusat Sumber Daya Air Tanah dan Geologi Lingkungan merupakan bagian dari Kementrian ESDM (Energi dan Sumber Daya Mineral) sehingga badan hukumnya menyerupai badan hukum Kementrian ESDM (Energi dan Sumber Daya Mineral). Landasan hukum kementrian adalah Bab V Pasal 17 UUD 1945 yang menyebutkan bahwa:

1. Presiden dibantu oleh menteri-menteri negara.
2. Menteri-menteri itu diangkat dan diperhentikan oleh Presiden.
3. Setiap menteri membidangi urusan tertentu dalam pemerintahan.
4. Pembentukan, pengubahan, dan pembubaran kementrian negara diatur dalam undang-undang.

Lebih lanjut, kementrian diatur dalam Undang-Undang Nomor 39 Tahun 2008 tentang Kementrian Negara dan Peraturan Presiden Nomor 47 Tahun 2009 tentang Pembentukan dan Organisasi Kementrian Negara. [9]

2.1.5 Struktur Organisasi dan *Job Description*

Berikut adalah struktur organisasi dan *job description* yang ada di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan.

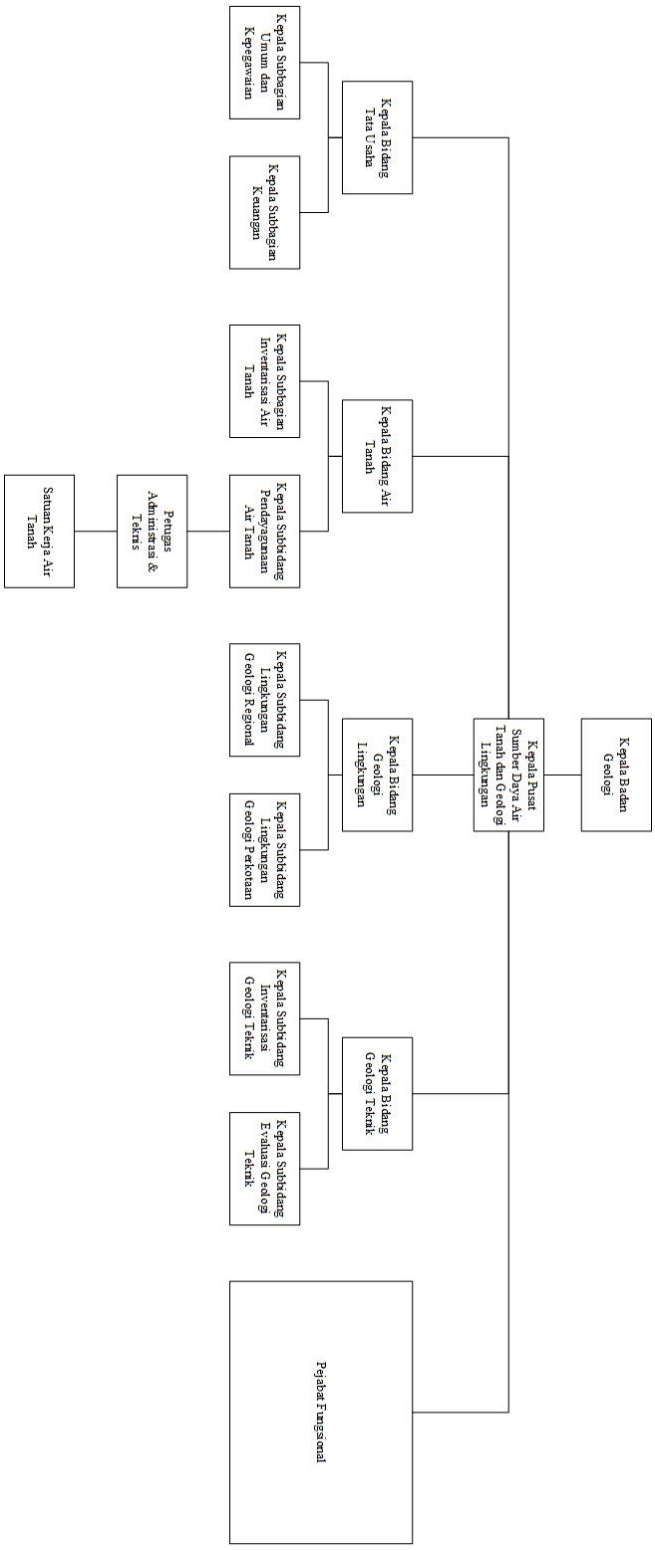
2.1.5.1 Struktur Organisasi

Struktur organisasi perusahaan merupakan bagian yang sangat penting dalam penyelenggaraan kegiatan, karena struktur organisasi perusahaan terdapat pendelegasian wewenang terhadap setiap lini/tingkatan dalam perusahaan.

Berdasarkan Peraturan Menteri Energi dan Sumber Daya Mineral No. 0030/2005 tanggal 20 Juli 2005 tentang Organisasi dan Tata Kerja Departemen Energi dan Sumber Daya Mineral, Pusat Sumber Daya Air Tanah dan Sumber Daya Mineral merupakan salah satu unit yang berada di bawah Badan Penelitian dan Pengembangan Energi dan Sumber Daya Mineral. Struktur organisasi Pusat Sumber Daya Air Tanah dan Sumber Daya Mineral terdiri dari beberapa bidang yaitu:

1. **Bidang Tata Usaha**, mempunyai tugas melaksanakan urusan kepegawaian, keuangan, rumah tangga dan ketatausahaan Pusat.
2. **Bidang Geologi Air Tanah**, mempunyai tugas melaksanakan pengelolaan sarana dan prasarana penelitian dan pengembangan geologi bidang air tanah.
3. **Bidang Geologi Lingkungan**, mempunyai tugas melaksanakan pengelolaan sarana dan prasarana penelitian dan pengembangan geologi bidang lingkungan.
4. **Bidang Geologi Teknik**, pengelolaan sarana dan prasarana penelitian dan pengembangan geologi bidang teknik.
5. **Pejabat Fungsional**, mempunyai tugas melaksanakan dan memberikan pelayanan jasa penelitian dan pengembangan di bidang energi dan sumber daya mineral, serta melaksanakan tugas lainnya yang didasarkan pada keahlian dan atau keterampilan tertentu sesuai ketentuan peraturan perundang-undangan.

Struktur Organisasi pada Pusat Sumber Daya Air Tanah dan Geologi Lingkungan dapat dilihat seperti pada gambar 2.2.



Gambar 2.2 Struktur Organisasi Pusat Sumber Daya Air Tanah dan Geologi Lingkungan

2.1.5.2 *Job Description*

Job description dari bagian yang terlibat dalam proses perizinan pengusahaan air tanah di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan adalah sebagai berikut:

1. Kepala Badan Geologi

   Kepala Badan Geologi merupakan pejabat yang menaungi seluruh kegiatan yang ada di badan geologi nasional. Memiliki tanggung jawab untuk melakukan pengesahan setiap perizinan yang masuk ke badan geologi.

2. Kepala Pusat Sumber Daya Air Tanah dan Geologi Lingkungan

   Kepala Pusat Sumber Daya Air Tanah dan Geologi Lingkungan merupakan pejabat yang menaungi seluruh pegawai dan segala aktivitas yang terkait di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan. Tujuan jabatan Kepala Pusat Sumber Daya Air Tanah dan Geologi Lingkungan ini adalah untuk mengkoordinir tugas dan fungsi setiap unit dan bidang di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan.

   Tanggung jawab utama seorang Kepala Pusat Sumber Daya Air Tanah dan Geologi Lingkungan adalah sebagai berikut:

   a. Melakukan disposisi terhadap surat yang masuk.
   b. Melakukan penetapan terhadap konsep surat.
   c. Melakukan penandatanganan terhadap surat keluar.
   d. Mengawasi setiap surat yang masuk dan surat yang keluar.

3. Kepala BidangAir Tanah

   Kepala Bidang Air Tanah merupakan pihak yang menaungi setiap bagian-bagian bidang yang dipimpinnya. Tujuan jabatan Kepala Bidang ini adalah untuk mengkoordinir tugas dan fungsi setiap unit dan bidang di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan.

   Tanggung jawab utama seorang Kepala Bidang Air Tanah adalah sebagai berikut:

   a. Memeriksa setiap surat yang akan dikeluarkan oleh bidang yang dipimpinnya.

b. Menerima surat masuk yang telah melalui proses disposisi.

4. Kepala Sub Bidang Pendayagunaan Air Tanah

   Kasubbid merupakan pihak yang menjadi perpanjangan tangan dari Kepala Bidang Air Tanah di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan. Tanggung jawab utama dari seorang Kasubbid Air Tanah adalah sebagai berikut :

   a. Meneruskan surat yang akan dikeluarkan oleh kepala bidang air tanah yang telah dikeluarkan.
   b. Menerima surat masuk yang telah melalui proses disposisi.

5. Petugas Administrasi & Teknis Air Tanah

   Petugas Administrasi & Teknis Air Tanah merupakan pegawai atau karyawan di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan. Tujuan karyawan adalah untuk melakukan setiap aktivitas yang terkait di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan.

   Tanggung jawab utama Petugas Administrasi & Teknis Air Tanah adalah sebagai berikut :

   a. Membuat konsep surat yang akan dikeluarkan oleh setiap bidang tempat karyawan bertugas.
   b. Melakukan perbaikan ketika konsep surat salah.
   c. Melakukan pengiriman surat keluar yang dikeluarkan oleh setiap bidang tempat karyawan bertugas.
   d. Melakukan pengecekan surat permohonan perizinan pengusahaan air tanah yang masuk.

6. Satuan Kerja Air Tanah

   Satuan Kerja Air Tanah merupakan pihak yang bertanggung jawab untuk menerima permohonan perizinan pengusahaan air tanah dari para pemohon serta bertugas untuk melakukan pengecekan lapangan terhadap permohonan yang dikirimkan.

2.2 Landasan Teori

Landasan teori yang digunakan dalam penyusunan skripsi dan Penerapan Keamanan Sistem Informasi dan *Digital signature* pada Sistem Informasi Perizinan Pengusahaan Air Tanah di Pusat Sumber Daya Air Tanah dan Geologi Lingkungan adalah sebagai berikut:

2.2.1 Sistem Informasi [10]

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

2.2.2 Basis Data [11]

Basis data adalah sekumpulan data yang terkait atau saling berhubungan, terorganisir dan bisa melibatkan lebih dari 1 organisasi. Sebuah basis data mencatat berbagai data yang diperlukan oleh suatu organisasi. Rekaman-rekaman data tersebut pada suatu saat akan diambil dan melalui suatu pemrosesan akan diperoleh informasi yang dikehendaki oleh pengguna. Basis data berbeda dengan sistem pemrosesan berbasis berkas. Sistem pemrosesan berbasis berkas adalah suatu model penyimpanan data yang medasarkan pada penyimpanan data dalam bentuk berkas, yang memiliki banyak kelemahan dibanding basis data. Sistem seperti ini banyak dipakai di masa lalu. Salah satu perangkat lunak yang biasa dipakai untuk mengimplementasikannya adalah COBOL.

2.2.3 *Object Oriented Programming* [12]

Pemrograman berorientasi objek atau *Object-Oriented Programming* disingkat OOP merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang

lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

2.2.4 *Framework* [13]

Framework adalah sebuah abstraksi di mana sebuah perangkat lunak yang melakukan fungsional umum mampu di modifikasi oleh sintaks yang digunakan pengguna untuk menghasilkan sebuah perangkat lunak yang spesifik. *Framework* bersifat universal dan mampu digunakan kembali. Berikut ini adalah perbedaan *framework* dengan *library* perangkat lunak pada umumnya:

1. *Framework* memiliki aturan atau *flow control* tersendiri.
2. *Framework* memiliki perilaku tersendiri.
3. *Framework* harus bisa di *extend* untuk digunakan oleh pengguna.

Sintaks *framework* tidak bisa dimodifikasi, oleh karena itu pengguna harus meng-*extend framework* untuk menggunakannya.

2.2.5 MVC (*Model View Controller*) [14]

MVC (*Model-View-Controller*) adalah sebuah konsep perangkat lunak yang mengelompokkan beberapa elemen dari sebuah arsitektur perangkat lunak. Sebuah perangkat lunak dikelompokkan menjadi 3 bagian yaitu *Model, View* dan *Controller*. Setiap kelompok memiliki tanggung jawab yang berbeda, sehingga tidak mengganggu satu sama lain. Inilah yang menjadikan MVC menjadi salah satu konsep perangkat lunak yang cukup sering di implementasikan.

- *Model*, merepresentasikan struktur data. Biasanya, kelas model berisi fungsi yang membantu proses CRUD (*Create Retrieve Update Delete*) pada *database*.
- *View*, berisi tampilan atau antarmuka aplikasi.
- *Controller*, mengatur proses pertukaran *resources* antara *Model*, *View* atau *resource* yang di butuhkan untuk melayani permintaan HTTP

2.2.6 *Quick Response* (QR) *Code* [15]

QR *Code* atau Kode QR adalah suatu jenis kode matriks atau kode batang dua dimensi yang dikembangkan oleh Denso Wave, sebuah divisi Denso Corporation yang merupakan sebuah perusahaan Jepang dan dipublikasikan pada tahun 1994 dengan fungsionalitas utama yaitu dapat dengan mudah dibaca oleh pemindai. QR merupakan singkatan dari *Quick Response* atau respons cepat, yang sesuai dengan tujuannya adalah untuk menyampaikan informasi dengan cepat dan mendapatkan respons yang cepat pula. Berbeda dengan kode batang, yang hanya menyimpan informasi secara horizontal, kode QR mampu menyimpan informasi secara horizontal dan vertikal, oleh karena itu secara otomatis Kode QR dapat menampung informasi yang lebih banyak daripada kode batang.

2.2.7 Digital Signature

Digital signature atau Tanda Tangan Elektronik adalah tanda tangan yang terdiri atas Informasi Elektronik yang dilekatkan, terasosiasi atau terkait dengan Informasi Elektronik lainnya yang digunakan sebagai alat verifikasi dan autentikasi [16]. Penanda Tangan adalah subjek hukum yang terasosiasikan atau terkait dengan Tanda Tangan Elektronik [16].

Tanda Tangan Elektronik memiliki kekuatan hukum dan akibat hukum yang sah selama memenuhi persyaratan sebagai berikut:

a. data pembuatan Tanda Tangan Elektronik terkait hanya kepada Penanda Tangan;
b. data pembuatan Tanda Tangan Elektronik pada saat proses penandatanganan elektronik hanya berada dalam kuasa Penanda Tangan;
c. segala perubahan terhadap Tanda Tangan Elektronik yang terjadi setelah waktu penandatanganan dapat diketahui;
d. segala perubahan terhadap Informasi Elektronik yang terkait dengan Tanda Tangan Elektronik tersebut setelah waktu penandatanganan dapat diketahui;
e. terdapat cara tertentu yang dipakai untuk mengidentifikasi siapa Penandatangannya; dan

f. terdapat cara tertentu untuk menunjukkan bahwa Penanda Tangan telah memberikan persetujuan terhadap Informasi Elektronik yang terkait.

Tanda Tangan Elektronik terbagi menjadi 2 jenis, yaitu Tanda Tangan Elektronik tersertifikasi dan Tanda Tangan Elektronik tidak tersertifikasi [5]. Tanda Tangan Elektronik tersertifikasi harus memenuhi persyaratan:

a. Dibuat dengan menggunakan jasa penyelenggara sertifikasi elektronik; dan
b. Dibuktikan dengan sertifikat elektronik [5].

Tanda Tangan Elektronik tidak tersertifikasi adalah Tanda Tangan Elektronik yang dibuat tanpa menggunakan jasa penyelenggara sertifikasi elektronik [5].

Dalam skripsi ini, Pusat Sumber Daya Air Tanah dan Geologi Lingkungan adalah penyelenggara Tanda Tangan Elektronik. Penyelenggara Tanda Tangan Elektronik adalah badan hukum yang berfungsi sebagai pihak terpercaya yang memfasilitasi pembuatan Tanda Tangan Elektronik [5]. Tanda Tangan Elektronik dibuat dengan menggunakan sejumlah data. Data Pembuatan Tanda Tangan Elektronik adalah kode pribadi, kode biometrik, kode kriptografi, dan/atau kode yang dihasilkan dari pengubahan tanda tangan manual menjadi Tanda Tangan Elektronik, termasuk kode lain yang dihasilkan dari perkembangan Teknologi Informasi [5]. Tanda Tangan Elektronik akan dilekatkan pada dokumen elektronik berupa dokumen perizinan pengusahaan air tanah.

Dokumen Elektronik adalah setiap Informasi Elektronik yang dibuat, diteruskan, dikirimkan, diterima, atau disimpan dalam bentuk analog, digital, elektromagnetik, optikal, atau sejenisnya, yang dapat dilihat, ditampilkan, dan/atau didengar melalui komputer atau Sistem Elektronik, termasuk tetapi tidak terbatas pada tulisan, suara, gambar, peta, rancangan, foto atau sejenisnya, huruf, tanda, angka, kode akses, simbol atau perforasi yang memiliki makna atau arti atau dapat dipahami oleh orang yang mampu memahaminya [5].

Sedangkan Informasi Elektronik adalah satu atau sekumpulan data elektronik, termasuk tetapi tidak terbatas pada tulisan, suara, gambar, peta, rancangan, foto, electronic data interchange (EDI), surat elektronik (electronic mail), telegram, teleks, telecopy atau sejenisnya, huruf, tanda, angka, kode akses,

simbol, atau perforasi yang telah diolah yang memiliki arti atau dapat dipahami oleh orang yang mampu memahaminya [5].

2.2.8 Kriptografi [17]

Kriptografi adalah ilmu mengenai teknik enkripsi di mana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi. Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. Proses enkripsi dilakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma dianggap sesuatu yang tidak baik. Rahasia terletak di beberapa parameter yang digunakan, jadi kunci ditentukan oleh parameter. Parameter yang menentukan kunci dekripsi itulah yang harus dirahasiakan (parameter menjadi ekuivalen dengan kunci).

2.2.9 SSL/TLS [18]

SSL adalah singkatan dari *Secure Socket Layer*, suatu *defacto standard* yang dibuat oleh Netscape, perusahaan pembuat *web browser* terpopuler tahun 1995. Ada dua versi SSL yang digunakan secara umum, yaitu SSL2 dan SSL3. Tahun 1999, standard SSL diambil alih oleh Internet Engineering Task Force (IETF) dan namanya diubah menjadi TLS, yang merupakan singkatan dari *Transport Layer Security* (SSL3 menjadi TLS versi 1). Perubahan nama ini mungkin agar nama menjadi lebih netral karena *socket* adalah istilah Unix. Versi terbaru dari TLS adalah versi 1.2. Karena nama SSL sudah sangat melekat, meskipun nama sudah berganti, dalam skripsi ini menyebutnya sebagai SSL/TLS.

SSL/TLS dimaksudkan untuk mengamankan sesi komunikasi antar proses. Dalam suatu *operating system* yang *multi-tasking* seperti Unix, Microsoft Windows, Linux atau MacOS, berbagai proses berjalan secara bersamaan. (Untuk melihat berbagai proses yang aktif, jika menggunakan Microsoft Windows, kita dapat menggunakan Windows Task Manager. Jika menggunakan Unix, Linux atau MacOS 10, kita dapat menggunakan *command line program* ***ps*** untuk melihat berbagai proses yang aktif.) Pengamanan komunikasi tidak terbatas pada

pengamanan komunikasi antar proses dalam satu komputer, akan tetapi juga pengamanan komunikasi antar proses di dua komputer yang berbeda. Bahkan pengamanan komunikasi antar proses di dua komputer yang berbeda jauh lebih penting karena terdapat lebih banyak ancaman. Sebagai contoh, penggunaan terbesar dari SSL/TLS adalah untuk "sesi aman" antara proses *web browser* dengan proses *web server* yang biasanya berada pada dua komputer yang berbeda.

2.2.9.1 Standar SSL/TLS

Dalam sesi menggunakan SSL/TLS, proses yang disebut *client* berkomunikasi dengan proses yang disebut *server*. Secara garis besar, sesi antara *client* dan *server* diamankan dengan, pertama melakukan *handshake*, lalu mengenkripsi komunikasi antara *client* dan *server* selama sesi berlangsung. Tujuan dari *handshake* adalah:

1. *Server authentication (optional)*.
2. Menentukan parameter enkripsi.
3. *Client authentication (optional)*.

Bagian *handshake* mungkin merupakan yang terpenting dalam sesi SSL/TLS. Yang jelas *handshake* merupakan bagian paling rumit dari segi protokol. Secara garis besar, protokol *handshake* adalah sebagai berikut:

1. *Client* mengirim ke *server* nomor versi SSL/TLS yang digunakan *client*, parameter enkripsi, data yang dibuat secara acak, dan informasi lain yang dibutuhkan oleh *server* untuk berkomunikasi dengan *client*. Jika dibutuhkan, *client* juga meminta *server certificate*.
2. *Server* mengirim ke *client* nomor versi SSL/TLS yang digunakan *server*, parameter enkripsi, data yang dibuat secara acak, dan informasi lain yang dibutuhkan oleh *client* untuk berkomunikasi dengan *server*. Jika diminta dalam langkah 1, *server* juga mengirim *server certificate*. Jika dibutuhkan, *server* juga meminta *client* untuk mengirim *client certificate*.
3. Jika *client* meminta *server certificate* dalam langkah 1, *client* melakukan *server authentication* (akan dijelaskan secara lebih rinci) menggunakan *server certificate* dan informasi lain yang didapat. Jika *authentication*

sukses, *server certificate* tidak diminta, atau pengguna mengizinkan, *client* meneruskan ke langkah 4. Jika tidak, sesi dihentikan.

4. Menggunakan data yang telah didapat, *client* membuat suatu premaster secret untuk sesi. Tergantung jenis enkripsi yang digunakan, ini dapat dilakukan dengan partisipasi *server*. Premaster secret dienkripsi menggunakan kunci publik *server* (diambil dari *server certificate*), lalu dikirim ke *server*.
5. Jika *server* meminta *client certificate* pada langkah 2, *client* menandatangan secara digital data yang unik untuk sesi yang diketahui oleh *client* dan *server*. Data berikut *digital signature* dan *client certificate* dikirim oleh *client* ke *server*.
6. Jika *server* meminta *client certificate* pada langkah 2, *server* melakukan *client authentication*. Jika *authentication* diminta dan gagal, maka sesi dihentikan.
7. *Client* dan *server* membuat master secret menggunakan premaster secret. Master secret digunakan oleh *client* dan *server* untuk membuat kunci sesi yang merupakan kunci enkripsi simetris. Enkripsi simetris adalah teknik kriptografi yang dalam proses enkripsi maupun dekripsi menggunakan kunci yang sama.
8. *Client* memberi tahu *server* bahwa kunci sesi akan digunakan untuk mengenkripsi komunikasi lebih lanjut. *Client* kemudian mengirim pesan yang dienkripsi ke *server* yang mengatakan bahwa ia selesai dengan *handshake*.
9. *Server* memberi tahu *client* bahwa kunci sesi akan digunakan untuk mengenkripsi komunikasi lebih lanjut. *Server* kemudian mengirim pesan yang dienkripsi ke *client* yang mengatakan bahwa ia selesai dengan *handshake*.
10. *Handshake* selesai.

Server authentication dilakukan dengan memeriksa *server certificate*. Dalam *server certificate* terdapat informasi antara lain:

a. kunci publik *server*,
b. masa berlaku *certificate*,
c. domain name untuk *server*, dan
d. domain name untuk pembuat *certificate* (biasanya *certificate* dibuat oleh suatu *certificate authority*).

Server certificate ditanda-tangan secara digital oleh pembuatnya. Pemeriksaan *certificate* dilakukan dengan menjawab berbagai pertanyaan sebagai berikut:

a. Apakah tanggal pemeriksaan berada dalam masa berlaku *certificate*?
b. Apakah pembuat *certificate* (teridentifikasi dengan domain name pembuat) dapat kita percayai? Pembuat bisa merupakan suatu *certificate authority* atau sumber lain.
c. Apakah *certificate* ditanda-tangan secara digital dengan benar oleh pembuatnya?
d. Apakah domain name untuk *server* yang tertera dalam *certificate* sesuai dengan domain name untuk *server* yang sebenarnya?

Jika semua pertanyaan terjawab secara memuaskan, maka *authentication* sukses. Jika ada pertanyaan yang jawabannya tidak memuaskan, maka *authentication* gagal. *Client authentication* juga dilakukan serupa. Setelah *handshake* selesai, komunikasi antara *client* dan *server* dienkripsi menggunakan kunci sesi. Standard SSL/TLS tentunya juga menentukan format yang harus digunakan dalam komunikasi antara *client* dan *server*. Standard juga menentukan *cipher suite* yang dapat digunakan. *Cipher* suite adalah kombinasi metode enkripsi yang terdiri dari:

1. Metode enkripsi untuk *key* exchange atau *key agreement*.
2. Metode enkripsi untuk *certificate*.
3. Metode enkripsi menggunakan kunci sesi.
4. Metode enkripsi untuk message *authentication* code (MAC).

Untuk *key* exchange atau *key agreement*, metode enkripsi yang dapat digunakan antara lain RSA, dan Diffie-Hellman (DH). Untuk *certificate*, metode enkripsi yang dapat digunakan antara lain RSA, dan DSA/DSS. Untuk enkripsi menggunakan

kunci sesi, metode enkripsi yang dapat digunakan antara lain RC4, 3DES, AES 128 bit, dan AES 256 bit.

Jika menggunakan *block cipher* seperti 3DES atau AES, maka metode enkripsi harus menggunakan mode *cipher* block chaining (CBC). Untuk message *authentication* code, metode enkripsi yang dapat digunakan antara lain:

1. HMAC-MD5,
2. HMAC-SHA-1,
3. HMAC-SHA256.

Standard TLS versi 1.2 mengharuskan implementasi sedikitnya bisa mendukung *cipher suite* yang terdiri dari RSA untuk *key* exchange dan *certificate*, AES 128 bit mode CBC untuk enkripsi menggunakan kunci sesi, dan HMACSHA-1 untuk *message authentication code*.

2.2.9.2 Penggunaan SSL/TLS

Seperti dikatakan sebelumnya, penggunaan terbesar SSL/TLS adalah untuk *secure web browsing*. Semua *web browser* yang populer mendukung *secure web browsing* menggunakan SSL/TLS. Biasanya pengguna *web browser* tidak perlu mengetahui SSL/TLS. Saat *web browser* akan menampilkan web page dengan prefix https (jadi bukan http), maka *web browser* secara otomatis akan memulai sesi SSL/TLS. Dalam melakukan *handshake* SSL/TLS, *web browser* akan melakukan *server authentication* dengan memeriksa *certificate* untuk web *server*. *Web browser* biasanya sudah memiliki daftar *certificate authority* yang dapat dipercaya, dan ada *web browser* yang memperbolehkan pengguna untuk menambah pembuat *certificate* yang dipercaya ke dalam daftar. Jika ada masalah dalam *authentication* maka *web browser* biasanya memberi tahu pengguna dan menanyakan pengguna apakah sesi diteruskan atau tidak. Biasanya ada juga opsi untuk menambah *certificate* yang bermasalah ke daftar pengecualian dimana *certificate* yang ada dalam daftar tidak perlu diperiksa. Jika *handshake* SSL/TLS berhasil maka *web page* dapat ditampilkan setelah terlebih dahulu di unduh dengan proteksi sesi SSL/TLS.

Selain *web browser*, perangkat lunak untuk web *server* seperti Apache juga sudah mendukung SSL/TLS. Jadi kita dapat membuat web *server* yang dapat melayani pengguna internet dengan *secure web page*. *Apache* memberi opsi untuk menggunakan *server certificate* yang dibuat sendiri atau dibuat oleh suatu *certificate authority*. Tentunya ada kemungkinan *web browser* pengguna akan berpendapat bahwa *certificate* bermasalah, akan tetapi ini dapat diatasi seperti dibahas di paragraf sebelum ini.

Karena *web browser* dan web *server* mendukung SSL/TLS, maka semua sistem informasi *client-server* yang berbasis web dapat menggunakan fasilitas SSL/TLS dengan mudah. Penggunaan sistem informasi akan sangat fleksibel karena pengguna dapat berada dimana saja asalkan ada koneksi TCP/IP dengan *server*, contohnya:

1. di komputer yang sama dengan *server*,
2. di komputer lain yang terhubung dengan *server* melalui *local area network*, atau
3. di lokasi lain, bahkan di negara yang berbeda waktu 12 jam dengan *server*, asalkan ada koneksi internet ke *server*.

Jika sistem informasi dibuat menggunakan sesuatu yang *platform independent* seperti Apache-MySQL-PHP, maka sistem informasi akan lebih fleksibel lagi karena *server* dapat ditempatkan di komputer dengan operating system apa saja, baik Microsoft Windows, Linux, MacOS 10, FreeBSD, OpenBSD, atau varian Unix lainnya, dan dapat dengan mudah dipindahkan. Tentunya penggunaan SSL/TLS tidak harus melalui web. Pada prinsipnya apa saja yang memerlukan pengamanan komunikasi antar proses dapat
menggunakan SSL/TLS.

Ada beberapa alat yang dapat membantu pembuatan sistem yang menggunakan SSL/TLS. Contohnya adalah *cryptographic* library seperti RSA BSafe atau Cryptlib. Contoh lain adalah OpenSSL yang juga memberikan fasilitas *cryptographic* library selain memberikan fasilitas SSL/TLS. Cryptlib dan OpenSSH keduanya menggunakan OpenSSL. OpenSSL adalah implementasi open source dari SSL/TLS yang didasarkan pada SSLeay, implementasi SSL oleh Eric

Young. Meskipun *open source*, OpenSSL menggunakan licensing yang tidak kompatibel dengan GPL (GNU Public License). Untuk pembaca yang menginginkan implementasi SSL yang lebih kompatibel dengan GPL, GnuTLS adalah implementasi serupa yang menggunakan LGPL (Lesser GNU Public License).

2.2.10 Aplikasi *Public Key Infrastructure* [19]

Hampir semua aplikasi kriptografi di bidang teknologi informasi menggunakan kriptografi *public key*. Aplikasi dasar kriptografi *public key* memang untuk *key* management dan *digital signature*. Namun bermula dari aplikasi dasar, kebutuhan berkembang dan *certificate* management kini menjadi bagian penting dari kriptografi *public key*. Suatu *public key infrastructure* (PKI) adalah suatu infrastruktur yang mendukung

1. *key* management termasuk *key* generation, *key* exchange, dan *key agreement*,
2. digital signing dan *digital signature* checking, dan
3. *certificate* management termasuk *certificate* generation, *certificate* publishing, *certificate* checking, dan *certificate revocation*.

Key generation dalam hal ini adalah proses pembuatan pasangan kunci publik dan privat, baik untuk keperluan *digital signature* maupun keperluan *key* management. Pembuatan kunci simetris tidak masuk disini karena tidak melibatkan unsur kriptografi publik, kecuali jika menggunakan *key agreement*. *Key* exchange adalah proses pengiriman kunci simetris secara aman menggunakan kriptogafi *public key*. *Key agreement* adalah proses pembuatan kunci simetris secara bersama menggunakan kriptografi *public key*, contohnya menggunakan Diffie-Hellman. Kunci simetris hasil *key agreement* bisa berupa seed yang digunakan untuk membuat kunci simetris lainnya.

Digital signing adalah proses pembuatan *digital signature* contohnya menggunakan RSA atau DSA, sedangkan *digital signature* checking adalah proses pengecekan *digital signature* apakah sesuai dengan naskah yang disign dan kunci publik yang diklaim.

Certificate generation adalah proses pembuatan *certificate* untuk kunci publik, baik untuk keperluan *digital signature* maupun untuk keperluan *key* management. Secara garis besar, ada dua standard untuk format *certificate* yaitu format X.509 dan format PGP. *Certificate* publishing bisa dilakukan menggunakan fasilitas seperti LDAP (lightweight directory access protocol) atau cara yang lebih sederhana. *Certificate* checking adalah proses pengecekan *certificate*, baik dari segi format, maupun dari segi isi. *Certificate revocation* adalah proses pembatalan suatu *certificate* yang telah dibuat. Semua aspek *certificate* management dapat melibatkan *certificate authority* yaitu seorang atau suatu badan yang fungsi utamanya adalah mengesahkan *certificate*.

Solusi PKI biasanya berbasis pada X.509 atau pada PGP. Pada awalnya, solusi berbasis X.509 adalah solusi komersial, sedangkan solusi PGP populer di kalangan open source saat RSA masih dilindungi hak paten dan pemerintah Amerika Serikat melakukan kontrol yang ketat1 terhadap penggunaan kriptografi. Dengan habisnya masa berlaku paten untuk RSA dan mengendurnya kontrol pemerintah Amerika Serikat terhadap penggunaan kriptografi, tren saat ini adalah standardisasi kearah X.509.

2.2.11 Aplikasi *Cryptographic Library* [20]

Cryptographic library sangat membantu untuk membuat kriptografi terintegrasi dengan aplikasi. Cara integrasi bisa embedded (di-*compile* bersama aplikasi) atau melalui module yang dapat dimasukan oleh aplikasi, contohnya

a. dynamically linked library (DLL) untuk Microsoft Windows, atau
b. jar untuk Java.

Penggunaan *cryptographic* library oleh aplikasi biasanya dilakukan melalui application program interface (API). Yang diberikan oleh *cryptographic* library antara lain:

a. berbagai algoritma kriptografi, contohnya AES, RSA, dan DSA,
b. kemampuan *public key infrastructure* (PKI), dan
c. secure sessions, contohnya SSL/TLS.

OpenSSL merupakan produk open source, RSA BSafe merupakan produk komersial (tetapi ada versi yang disumbangkan oleh RSA sebagai open source), dan Cryptlib adalah open source dengan dua macam lisensi (dual license) yaitu GPL dan komersial.

2.2.12 SHA [21]

Algoritma *secure hashing* SHA dirancang oleh National Security Agency (NSA) dan dijadikan standard FIPS. Ada 4 varian SHA dalam standard FIPS-180-2 dengan parameter yang berbeda. 4 varian SHA dapat dilihat pada tabel 2.1. Keamanan algoritma didasarkan pada fakta bahwa *birthday attack* pada *digest* sebesar *n* bit menghasilkan *collision* dengan faktor kerja sekitar $2^{n/2}$. *Birthday Attack* adalah sejenis serangan *cryptographic* yang mengeksploitasi matematika pada *birthday problem* dalam teori probabilitas dengan cara menyederhanakan kompleksitas ketika membongkar fungsi *hashing*. [22]

Tabel 2.1 4 Varian SHA

Algoritma	Naskah (bit)	Block (Bit)	Word (Bit)	*Digest* (bit)	Keamanan (bit)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

SHA-1 menggunakan fungsi f_t, dimana $0 \leq t \leq 79$, dengan masukan 3 *word* masing-masing sebesar 32 bit dan menghasilkan keluaran 1 *word*:

$$f(x) = \begin{cases} Ch(x,y,z) = (x \wedge a) \oplus (\neg x \wedge z), 0 \leq t \leq 19 \\ Parity(x,y,z) = x \oplus y \oplus z, 20 \leq t \leq 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), 40 \leq t \leq 59 \\ Parity(x,y,z) = x \oplus y \oplus z, 60 \leq t \leq 79 \end{cases}$$

SHA-1 menggunakan konstan k_t sebagai berikut (dengan menggunakan *hexadecimal*):

$$f(x) = \begin{cases} 5a827999\ 0 \leq t \leq 19 \\ 6ed9eba1\ 20 \leq t \leq 39 \\ 8f1bbcdc\ 40 \leq t \leq 59 \\ ca62c1d6\ 60 \leq t \leq 79 \end{cases}$$

Seperti halnya dengan MD5, SHA-1 terdiri dari dua tahap yaitu *preprocessing* dan *hashing*. *Preprocessing* dimulai dengan *padding* yang prosesnya persis sama dengan MD5, yaitu setelah akhir naskah, 1 bit dengan nilai 1 ditambahkan, disusul oleh bit dengan nilai 0 sebanyak 0 sampai dengan 511 tergantung panjang naskah, dan diahiri dengan 64 bit yang merepresentasikan besar naskah asli. Setelah *padding*, naskah terdiri dari *n word M*[0 . . . n − 1] dimana n adalah kelipatan 16. Langkah berikutnya dalam preprocessing adalah menyiapkan SHA-1 *buffer* sebesar 5 word:

$$\left(H_0^{(0)}, H_1^{(0)}, H_2^{(0)}, H_3^{(0)}, H_4^{(0)}\right).$$

Buffer diberi nilai awal sebagai berikut (nilai dalam hexadecimal):

$$H_0^{(0)} \leftarrow 67452301$$

$$H_1^{(0)} \leftarrow efcdab89$$

$$H_2^{(0)} \leftarrow 98badcfe$$

$$H_3^{(0)} \leftarrow 10325476$$

$$H_4^{(0)} \leftarrow c3d2e1f0$$

Setelah *buffer* diberi nilai awal, tahap kedua yaitu *hashing* dilakukan terhadap setiap blok ($M^{(1)}$, $M^{(2)}$, ..., $M^{(n)}$) sebagai berikut (i = 1, 2, ..., n):

1. Siapkan *message schedule* $\{W_t\}$:

$$W_t \leftarrow \begin{cases} M_t^{(i)}, 0 \le t \le 15 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, 16 \le t \le 79 \end{cases}$$

2. Berikan nilai awal untuk variabel *a*, *b*, *c*, *d*, dan *e*

$$a \leftarrow H_0^{(i-1)}$$

$$b \leftarrow H_1^{(i-1)}$$

$$c \leftarrow H_2^{(i-1)}$$

$$d \leftarrow H_3^{(i-1)}$$

$$e \leftarrow H_4^{(i-1)}$$

3. Untuk t = 0, 1, 2, ..., 79:

$$T \leftarrow (a <<< 5) + f_t(b, c, d) + e + K_t + W_t (mod\ 2^{23})$$

$$e \leftarrow d$$
$$d \leftarrow c$$
$$c \leftarrow b <<< 30$$
$$b \leftarrow a$$
$$a \leftarrow T$$

4. Lakukan kalkulasi *hash value* tahap *i*:

$$H_0^i \leftarrow a + H_0^i$$
$$H_1^i \leftarrow b + H_1^i$$
$$H_2^i \leftarrow c + H_2^i$$
$$H_3^i \leftarrow d + H_3^i$$
$$H_4^i \leftarrow e + H_4^i$$

Setelah *hashing* dilakukan pada semua blok ($M^{(1)}$, $M^{(2)}$,....., $M^{(n)}$), kita dapatkan *digest* sebagai berikut

$$\left(H_0^{(n)}, H_1^{(n)}, H_2^{(n)}, H_3^{(n)}, H_4^{(n)}\right).$$

Beberapa ahli kriptografi telah mencoba mencari kelemahan SHA-1. Xiaoyun Wang dan koleganya berhasil memperkecil ruang pencarian untuk *collision* SHA-1 dari 2^{80} operasi SHA-1 (yang merupakan "kekuatan" teoritis SHA-1 jika SHA-1 tidak memiliki kelemahan, berdasarkan birthday attack) menjadi 2^{69} operasi. Walaupun demikian, penggunaan SHA-1 masih dianggap cukup aman, dan jika ingin lebih aman lagi, maka SHA-256, SHA-384 atau SHA-512 dapat digunakan.

SHA-256, SHA-384 dan SHA-512, bersama dengan SHA-224 secara kolektif masuk dalam standard SHA-2. Saat bab ini ditulis, NIST sedang mengadakan sayembara pembuatan standard SHA-3 yang diharapkan akan lebih tangguh dari SHA-2.

2.2.13 RSA [23]

Tahun 1978, Len Adleman, Ron Rivest dan Adi Shamir mempublikasikan sistem RSA. Semula sistem ini dipatenkan di Amerika Serikat dan seharusnya masa paten habis tahun 2003, akan tetapi RSA Security melepaskan hak paten setelah 20

September 2000. Sebetulnya sistem serupa telah dilaporkan oleh Clifford Cocks tahun 1973 meskipun informasi mengenai ini baru dipublikasi tahun 1997 karena merupakan hasil riset yang diklasifikasikan sangat rahasia oleh pemerintah Britania Raya (Clifford Cocks bekerja untuk GCHQ, suatu badan di Britania Raya yang fungsinya serupa dengan fungsi NSA di Amerika Serikat), jadi validitas paten patut dipertanyakan karena adanya *prior art*. Berikut ini penjelasan secara garis besar bagaimana cara kerja RSA. Setiap pengguna memilih, menggunakan *random number generator*, dua bilangan prima yang sangat besar *p* dan *q* (masing-masing lebih dari 200 digit). Untuk produk $n = pq$, jika *p* dan *q* diketahui, fungsi Euler dapat dengan mudah di hitung yaitu

$$\emptyset(n) = (p-1)(q-1)$$

Kemudian pengguna memilih, menggunakan *random number generator*, suatu bilangan *e* diantara 1 dan $\emptyset(n)$ yang koprima dengan $\emptyset(n)$. Berikutnya pengguna mengkomputasi *inverse* dari e modulo $\emptyset(n)$.

Ini dapat dilakukan menggunakan *extended Euclidean algorithm*. Pengguna kemudian mempublikasi kunci publiknya

$$K_E = (n, e),$$

dan merahasiakan kunci privatnya

$$K_D = (n, d),$$

Rumus untuk mengenkripsi atau mendekripsi menggunakan kunci *public* adalah

$$M^e \bmod n$$

dimana *M* adalah representasi naskah asli (menggunakan bilangan bulat) jika mengenkripsi, atau representasi naskah acak jika mendekripsi. Rumus untuk mengenkripsi atau mendekripsi menggunakan kunci privat adalah

$$M^d \bmod n$$

dimana *M* adalah representasi naskah asli jika mengenkripsi atau representasi naskah acak jika mendekripsi. Tidak terlalu sulit untuk melihat bahwa naskah yang dienkripsi menggunakan kuncil publik dapat didekripsi menggunakan kunci privat:

$$(M^e)^d \equiv M^{ed} \equiv M \ (mod\ n).$$

Untuk gcd(*M*, *n*) > 1 dimana *M* bukan kelipatan *n*, ini hanya bisa terjadi jika *M* merupakan kelipatan *p* atau kelipatan *q*, tetapi bukan kelipatan keduanya. Jika *M* merupakan kelipatan *p*, maka

$$M^{ed} \equiv 0 \ (mod\ n)$$

$$M^{ed} \equiv M^{\emptyset(p)\emptyset(q)+1} \equiv M^{\emptyset(p)\emptyset(q)}M \equiv \ M \ (mod\ q)$$

Menggunakan *Chinese Remainder Theorem*, dengan *n* = *pq*, kita dapatkan

$$M^{ed} \equiv M \ (mod\ n)$$

Untuk *M* kelipatan *q*, hal serupa dapat ditunjukan. Untuk *M* kelipatan *n*, kita dapatkan

$$(M^e)^d \equiv 0 \equiv M \ (mod\ n)$$

Jadi

$$(M^e)^d \equiv M \ (mod\ n)$$

untuk sembarang *M*.

Naskah yang dienkripsi menggunakan kunci privat dapat didekripsi menggunakan kunci publik:

$$(M^e)^d \equiv M^{ed} \equiv M \ (mod\ n)$$

Secara umum, jika $f \not\equiv d \ (mod\ \emptyset(n))$ maka

$$(M^e)^f \not\equiv M \ (mod\ n)$$

yang berarti sesuatu yang dienkripsi menggunakan kunci publik tidak dapat didekripsi selain menggunakan kunci privat. Juga, jika $f \not\equiv d \ (mod\ \emptyset(n))$ maka secara umum

$$(M^e)^f \not\equiv M \ (mod\ n)$$

yang berarti sesuatu yang dienkripsi menggunakan kunci privat tidak dapat didekripsi selain menggunakan kunci publik. Keamanan dari RSA bersandar pada fakta bahwa mengetahui *n* dan *d* secara umum tidak membantu untuk mencari *e* yaitu *inverse* modulo $\emptyset(n)$ dari *d*. Hal ini karena mengetahui *n* tidak membantu mencari $\emptyset(n)$ jika *n* tidak bisa diuraikan menjadi *n* = *pq*.

Untuk menjaga keamanan tersebut, ada beberapa hal yang perlu diperhatikan dalam memilih *p* dan *q*:

a. Nilai *p* harus cukup jauh dari nilai *q*. Sebaiknya panjang dari *p* harus berbeda beberapa digit dari *q*. Jika nilai *p* terlalu dekat dengan nilai *q*, maka *Fermat factorization* dapat digunakan untuk menguraikan $n = pq$.
b. Sebaiknya gcd($p - 1$, $q - 1$) tidak terlalu besar.
c. Sebaiknya $p - 1$ dan $q - 1$ mempunyai faktor prima yang besar.

Karena perkembangan yang pesat dalam teknik penguraian, kunci RSA sebaiknya minimal 2048 bit. RSA dapat digunakan, baik untuk *key* distribution (termasuk *key exchange*), maupun untuk *digital signature*. Karena merupakan sistem pertama yang dapat digunakan untuk *key distribution* dan *digital signature*, RSA menjadi sistem kriptografi *public key* yang terpopuler. Boleh dikatakan semua standard protokol kriptografi memperbolehkan penggunaan RSA, termasuk SSL/TLS (untuk pengamanan http) dan SSH (*secure shell*).

Pembahasan RSA di atas tidak menjelaskan standar implementasi secara rinci, termasuk format data dan kunci. Untuk yang ingin mengetahui standard RSA dengan lebih rinci dipersilahkan membaca berbagai publikasi RSA Laboratories (bagian dari RSA Security) dalam seri yang berjudul *Public-Key Cryptography Standards* (PKCS).

2.2.13.1 Penggunaan Kriptografi Kunci Publik

Hampir semua sistem kriptografi *public key* yang telah dibahas dirancang untuk keperluan khusus. Diffie-Hellman dirancang untuk keperluan *key agreement*, DSA untuk *digital signature*, dan sistem yang berbasis pada zero-knowledge protocol untuk identifikasi. RSA dan ElGamal adalah dua sistem kriptografi *public key* yang dapat digunakan untuk enkripsi. Namun jika dibandingkan dengan sistem enkripsi klasik seperti AES, 3DES dan CAST, maka RSA dan ElGamal sangat lambat. Jadi penggunaan RSA dan ElGamal dalam enkripsi juga agak khusus yaitu mengenkripsi kunci enkripsi klasik atau mengenkripsi *digest* untuk keperluan *digital signature*.

Enkripsi kunci enkripsi klasik contohnya dalam suatu sesi SSL atau SSH, kunci sesi yang biasanya merupakan kunci AES, 3DES atau CAST, dienkripsi menggunakan kunci publik jenis RSA atau ElGamal. Demikian juga jika

mengenkripsi file yang ukurannya bisa cukup besar, kriptografi klasik digunakan untuk mengenkripsi file, lalu kunci klasik dienkripsi menggunakan kunci publik. Kriptografi *public key* tentunya juga berperan sangat besar dalam suatu *public key infrastructure*. Fungsi utama *public key infrastructure* adalah manajemen *digital signature* dan kunci publik untuk dekripsi, termasuk manajemen *certificate*, untuk kunci publik *digital signature* dan untuk kunci publik enkripsi. Aplikasi untuk kunci publik yang dikelolapun beragam, dari *certificate* untuk website, kunci publik untuk keperluan secure email, sampai dengan *certificate* untuk keperluan IPsec.

2.2.14 PKCS [24]

Dalam kriptografi, PKCS adalah bagian dari *public key cryptography standard* ditemukan dan dipublikasi oleh RSA Security sejak sekitar tahun 1990. RSA Security mempublikasi PKCS untuk mempromosikan teknik penggunaan kriptografi pada algoritma-algoritma yang sudah ada. Sampai saat ini, sudah terdapat sebanyak 15 standar yang digunakan. Standar PKCS dijelaskan pada tabel 2.2.

Tabel 2.2 Standar PKCS

	Versi	**Nama**	**Keterangan**
PKCS #1	2.2	RSA Cryptography Standard	Menjelaskan properti matematika dan format dari RSA Public Key dan Private Key. Standar ini juga menjelaskan algoritma dasar dan skema encoding/padding untuk menjalankan enkripsi, dekripsi, memproduksi dan memverifikasi *signature* RSA
PKCS #2		Dibatalkan	Tidak lagi aktif sejak 2010. Membahas enkripsi RSA tentang *message digest*. Standar ini sudah digabung ke PKCS #1
PKCS #3	1.4	Diffie-Hellman Key Agreement Standard	Sebuah protokol kriptografi yang memungkinkan 2 entitas yang tak saling mengenal untuk berinteraksi, berbagi kunci rahasia pada *channel* komunikasi yang tidak aman sekalipun.
PKCS #4		Dibatalkan	Tidak lagi aktif sejak 2010. Membahas sintaks kunci RSA. Standar ini sudah digabung ke PKCS #1
PKCS #5	2.0	Password-based Encryption Standard	Dijelaskan pada RFC 2898
PKCS #6	1.5	Extended-Certificate Syntax Standard	Menjelaskan penambahan spesifikasi *certificate* X.509

	Versi	Nama	Keterangan
PKCS #7	1.5	Cryptographic Message Syntax Standard	Digunakan untuk menanda tangan dan/atau enkripsi pesan menggunakan PKI. Digunakan juga untuk publikasi *certificate*. Dibentuk untuk format S/MIME berbasis standar RFC 5652, pembaharuan dari *Cryptographic Message Syntax Standard* (CMS). Biasa digunakan untuk SSO (Single-sign-on)
PKCS #8	1.2	Private-Key Information Syntax Standard	Digunakan untuk mengirimkan pasangan kunci *private certificate* (terenkripsi ataupun tidak)
PKCS #9	2.0	Selected Attribute Types	Menjelaskan beberapa jenis atribut untuk digunakan pada PKCS #6, PCKS #7, PKCS #8 dan PCKS #10
PKCS #10	1.7	Certification Request Standard	Menjelaskan format pesan untuk *certificate authority* untuk meminta *public key ceritificate*.
PKCS #11	2.3	Cryptographic Token Interface	Menjelaskan API untuk mengakses *cryptographic tokens*. Biasanya digunakan untuk SSO, *public key cryptography* dan *disk encryption*.
PKCS #12	1.1	Personal Information Exchange Syntax Standard	Menjeleaskan file format yang biasanya digunakan untuk menyimpan *private key* dengan menyertakan *public key*, diproteksi dengan *password-based symmetric key*.
PKCS #13		Elliptic Curve Cryptography Standard	
PKCS #14		Pseudo-random Number Generation	
PKCS #15	1.1	Cryptographic Token Information Format Standard	Menjelaskan standar yang mengizinkan user *cryptographic tokens* untuk menidentifikasi dirinya sendiri.

2.2.15 X.509 [25]

X.509 adalah bagian dari X.500, suatu percobaan yang sangat ambisius dari ITU (International Telecommunication Union) untuk standardisasi *directory services*. Judul dari standard X.509 adalah *The Directory*: *Public-key and attribute certificate frameworks*. X.500 sendiri dianggap terlalu rumit, namun format *certificate* X.509 dijadikan dasar untuk standard defacto oleh industri, dan kini standard *public key infrastructure* untuk internet yaitu PKIX (*Public Key Infrastructure* X.509) berdasarkan pada format X.509.

Berbeda dengan “cara” PGP yang cenderung demokratis, “cara” X.509 lebih bersifat struktural hirarkis. Ini adalah warisan dari X.500 yang semula ditujukan untuk membuat direktori pengguna jaringan komputer seperti direktori

telpon yang akan dikelola oleh berbagai perusahaan telpon di berbagai negara. Sifat-sifat struktural dalam X.509 antara lain:

a. Konsep *relative distinguished name* (RDN) yang cukup rumit dalam membuat *distinguished name* (DN) (identitas).
b. Konsep *certificate authority* yang semula dikaitkan dengan RDN

Konsep *relative distinguished* name (RDN) bertujuan untuk membuat identitas berdasarkan pada posisi dalam suatu hirarki. Hirarki ini diatur berdasarkan negara, propinsi, organisasi, sub-unit organisasi dan seterusnya. Berbagai negara dan organisasi melakukan lokalisasi standar yang kerap disebut *profile* yang menambah rumit konsep RDN. Semula, *certificate authority* (CA) juga dikaitkan dengan RDN, dimana satu CA mempunyai otoritas untuk satu subhirarki. Namun dalam prakteknya ini tidak berjalan.

Karena konsep RDN terlalu rumit dan dapat bertentangan dengan berbagai kepentingan seperti:

a. *privacy* (contohnya informasi pribadi), dan
b. *confidentiality* (contohnya banyak perusahaan yang tidak menginginkan informasi mengenai struktur organisasinya dipublikasikan),

belum lagi pada kenyataannya dunia tidak sepenuhnya tersusun rapi secara hirarkis tanpa tumpang tindih, akibatnya boleh dikatakan tidak ada yang mengetahui bagaimana cara membuat DN yang "benar." Meskipun demikian, DN tetap merupakan bagian tak terpisahkan dari *certificate* X.509, dan X.509 sudah merupakan defacto standard. Komponen DN yang kerap digunakan dijelaskan pada tabel 2.3

Tabel 2.3 Komponen DN yang digunakan

Komponen	Penjelasan
CountryName(C)	Kode negara 2 huruf berdasarkan ISO 3166 (contohnya ID).
Organization(O)	Nama organisasi (contohnya PT Sendiri).
OrganizationalUnit(OU)	Nama unit organisasi (contohnya Sales).
StateOrProvince(SP)	Nama negara bagian atau propinsi (contohnya DKI).
Locality(L)	Nama daerah (contohnya Kebayoran Baru)
CommonName(CN)	Nama pemegang *certificate*, bisa perorangan atau bagian dari organisasi, bahkan nama komputer (contohnya Budi Santoso).

Setiap komponen kecuali CountryName besarnya maksimum 64 karakter, sedangkan CountryName besarnya 2 karakter. Masih banyak komponen DN lainnya yang dapat digunakan, namun komponen yang terpenting sudah masuk dalam daftar diatas. Minimal suatu DN harus mempunyai komponen C dan komponen CN. DN dapat mengidentifikasi orang, unit organisasi atau perangkat, dan digunakan baik untuk identifikasi pemegang *certificate* maupun untuk identifikasi pembuat *certificate*.

Sejak penggunaan internet menjadi dominan, ada alternatif dari DN yang disebut *general name* (GN) yang lebih berorientasi pada internet dibandingkan dengan DN. Komponen GN termasuk antara lain DNS name, IP *address, email address* dan *uniform resource identifier*. Untuk identifikasi perangkat, penggunaan GN mungkin lebih cocok dibandingkan DN.

Sebetulnya ada bermacam jenis struktur untuk *certificate* X.509, termasuk *certificate* biasa, attribute *certificate*, *certification request* dan *certificate revocation list* (CRL). Untuk *certificate* biasa dijelaskan pada Tabel 2.4

Tabel 2.4 Komponen Certificate

Komponen	Penjelasan
Version	Versi X.509 yang digunakan.
SerialNumber	Nomor seri. Setiap *certificate* yang dibuat oleh suatu CA harus mempunyai nomor seri yang berbeda.
SignatureAlgorithm	Algoritma kriptografi yang digunakan untuk menanda-tangan *certificate*.
IssuerName	DN/GN untuk pembuat *certificate* (biasanya CA).
Validity	Masa berlaku *certificate*.
SubjectName	DN/GN untuk pemegang *certificate*.

Komponen	Penjelasan
SubjectPublicKeyInfo	Kunci publik pemegang *certificate*.
Extensions	Untuk berbagai macam informasi tambahan.

Attribute certificate fungsinya adalah sebagai *certificate* untuk berbagai atribut, jadi komponen SubjectPublicKeyInfo diganti oleh Attributes. *Certificate request* fungsinya adalah permintaan *certificate* untuk suatu kunci publik, jadi hanya berisi beberapa komponen yang dijelaskan pada tabel 2.5

Tabel 2.5 Komponen Certificate request

Komponen	Penjelasan
Version	Versi X.509 yang digunakan.
SerialNumber	DN/GN untuk pemegang kunci.
SignatureAlgorithm	Kunci publik yang diminta dibuatkan *certificate*.
IssuerName	Untuk berbagai macam informasi tambahan.

Certificate revocation list (CRL) dimaksudkan sebagai daftar *certificate* yang dibatalkan oleh suatu CA. Untuk berbagai macam alasan, termasuk kunci yang dicuri, suatu *certificate* dapat dibatalkan oleh CA yang membuatnya. Secara berkala, suatu CA mempublikasikan daftar *certificate* yang dibatalkannya dalam bentuk CRL. CRL dapat digunakan untuk menentukan validitas suatu *certificate*. Komponen dari CRL dijelaskan pada Tabel 2.6

Tabel 2.6 Komponen CRL

Komponen	Penjelasan
Version	Versi X.509 yang digunakan.
SignatureAlgorithm	Algoritma kriptografi yang digunakan untuk menanda-tangan *certificate*.
IssuerName	DN/GN untuk CA.
ThisUpdate	Waktu CRL ini dibuat.
NextUpdate	Waktu CRL berikutnya akan dibuat.
User*Certificate*	Daftar *certificate* yang dibatalkan.
*Revocation*Date	Tanggal *certificate* dibatalkan.

Meskipun dalam X.500 *certificate authority* (CA) dikaitkan dengan RDN, kini tidak ada lagi kaitan formal antara CA dengan RDN. *Certificate* bahkan tidak harus dibuat oleh CA. Namun konsep CA masih berguna, meskipun siapa yang patut menjadi CA tergantung pada aplikasi dan/atau situasi. Validitas dari suatu

kunci publik ditentukan oleh *certificate* untuk kunci tersebut: jika *certificate* dibuat oleh orang, badan atau CA yang dipercaya oleh seorang pengguna, maka kunci publik dianggap valid oleh pengguna tersebut. Perbedaan utama antara "cara" X.509 dan "cara" PGP dalam menentukan validitas *certificate* adalah dengan X.509, CA biasanya tidak ditentukan oleh pengguna, sedangkan dengan PGP pengguna berperan lebih besar dalam menentukan siapa yang dipercaya sebagai pembuat *certificate* menggunakan mekanisme *web of trust*.

Dari segi teknis, masalah yang dihadapi oleh X.509 dalam manajemen validitas *certificate* jauh lebih rumit dibandingkan masalah yang dihadapi PGP. Ini antara lain karena:

a. Berbagai CA dan aktor lainnya berperan aktif dalam manajemen validitas *certificate*.
b. Bermacam skenario penggunaan *certificate* X.509 dapat melibatkan aktor yang beragam dan dalam skala yang besar.
c. Konsep *certificate revocation* sukar untuk dipraktekkan secara efektif.

Sebagai contoh banyaknya aktor yang berperan dalam manajemen validitas *certificate*, kita bisa lihat begitu banyaknya *certificate authority* yang ada, baik yang bersifat komersial seperti Verisign, maupun yang non-komersial. Contoh penggunaan *certificate* X.509 yang beragam termasuk untuk keperluan *website authentication*, untuk *authentication* kunci publik suatu perangkat dalam jaringan yang menggunakan IPsec untuk VPN, dan untuk secure email menggunakan S/MIME. Sukarnya menggunakan konsep *certificate revocation* secara efektif pernah dijelaskan oleh Peter Gutmann (arsitek dari Cryptlib) dengan membuat analogi dengan basis data relasional dimana mempublikasikan *certificate* ibarat operasi *prepare* dalam suatu basis data relasional, sedangkan *certificate revocation* ibarat operasi *commit* dalam basis data relasional:

a. Dalam basis data relasional, status setelah *prepare* tetapi sebelum *commit* tidak bisa dijamin konsisten.
b. Analoginya untuk *certificate*, jika *certificate* telah dipublikasi tetapi *revocation* untuk *certificate* tersebut belum terlihat, maka status dari *certificate* tidak pasti, karena bisa saja *certificate* telah dibatalkan.

Akibat dari berbagai masalah tersebut, tidak ada satu standard protokol untuk manajemen *certificate*, tetapi ada beberapa protokol yang fungsinya beragam.

Salah satu standard protokol untuk manajemen *certificate* adalah standard *Lightweight Directory Access Protocol* (LDAP). Protokol ini digunakan untuk mengakses *certificate* store (tempat penyimpanan *certificate*) yang menggunakan direktori LDAP. Suatu LDAP server melayani client dengan operasi penyimpanan *certificate* di direktori, penghapusan *certificate* dari direktori, dan pencarian *certificate* dalam direktori. Karena LDAP berorientasi pada X.500 yang terkenal rumit, implementasi LDAP tidak bisa efisien sehingga tidak bisa digunakan dalam skala besar. Untuk skala besar, relational database banyak digunakan untuk *certificate* store.

Selain LDAP, berbagai protokol berjenis request/response yang digunakan untuk manajemen *certificate* antara lain:

a. CMP (*certificate management protocol*).
b. OCSP (*online certificate status protocol*).
c. TSP (*time-stamp protocol*).

CMP melayani *certificate* requests yaitu pembuatan *certificate* dan pembatalan (*revocation*) *certificate*. Komunikasi dengan CMP server dapat melalui http, email, atau cara lain. Karena CMP cukup rumit, sukar untuk menggunakannya dalam skala besar. Akibatnya, beberapa pembuat perangkat jaringan seperti Cisco menggunakan protokol yang lebih sederhana untuk digunakan dalam skala besar yaitu SCEP (*simple certificate enrollment protocol*).

OCSP adalah protokol untuk melayani pemeriksaan *certificate* yang berdasarkan pada *concept revocation*. Jadi OCSP *server* hanya dapat menjawab bahwa *certificate* telah dibatalkan (*revoked*) atau tidak, dan informasi yang digunakan biasanya tidak up-to-date. Seperti telah dibahas diatas, Peter Gutmann menganggap solusi ini tidak memuaskan. Peter Gutmann menyarankan penggunaan RTCS (*realtime certificate* status protocol) sebagai extension dari OCSP yang bisa menjawab apakah *certificate* valid atau tidak secara real-time. Buku ini merekomendasikan penggunaan RTCS seperti yang disarankan oleh Peter Gutmann.

TSP adalah protokol yang digunakan untuk melayani *time-stamping requests*. Suatu *time-stamping authority* (TSA) dapat diimplementasi menggunakan TSP server yang melayani *time-stamping requests* dengan membuat *time-stamps* yang ditanda-tangan menggunakan kunci TSA.

Kecuali untuk TSP, *server* untuk berbagai protokol *request/response* perlu menggunakan suatu *certificate store*. Tren saat ini adalah menggunakan basis data relasional sebagai *back end* untuk *certificate store* agar dapat digunakan dalam skala besar. PKI untuk komunitas terbuka (dimana tidak ada ikatan formal untuk anggota) biasanya tidak dapat berfungsi efektif. Ini antara lain karena:
Keperluan dan karakteristik anggota komunitas terbuka beragam.

a. Sukar untuk melakukan kontrol terhadap setiap anggota komunitas terbuka dan membuatnya bertanggung jawab atas penggunaan PKI.
b. Sukar untuk mendapatkan CA yang dapat dipercaya oleh semua pihak dalam komunitas terbuka.

PKI biasanya hanya dapat berfungsi efektif jika digunakan oleh komunitas tertutup dimana ada ikatan formal untuk anggota, misalnya

a. PKI untuk suatu perusahaan.
b. PKI untuk suatu pemerintahan atau badan pemerintahan.
c. PKI untuk komunitas jaringan bisnis tertentu misalnya jaringan bisnis suku cadang otomotif, dimana setiap anggota terdaftar secara formal dan mempunyai tanggung jawab yang jelas.
d. PKI untuk VPN.

2.2.16 S/MIME [26]

S/MIME (*Secure/Multipurpose Internet Mail Extensions*) adalah standar untuk *public key encryption* dan menanda tangani MIME data. Standar S/MIME sudah dijelaskan pada dokumen RFC 3369, 3370, 3850 dan 3851. S/MIME sebenarnya dikembangkan oleh RSA Data Security Inc. Spesifikasi aslinya menggunakan spesifikasi IETF MIME dengan standar industri de facto PKCS#7. S/MIME menyediakan layanan keamanan kriptografi untuk *electronic messaging*

applications seperti autentikasi, integritas pesan, tidak bisa disangkal, privasi dan keamanan data.

2.2.17 Bahasa Pemrograman PHP [27]

PHP merupakan secara umum dikenal sebagai bahasa pemrograman *script-script* yang membuat dokumen HTML secara *on the fly* yang diekseskuisi di *server web*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan editor teks atau editor HTML. Dikenal juga sebagai bahasa pemrograman *server side*.

Dengan menggunakan PHP maka *maintenance* suatu situs web menjadi lebih mudah. Proses pembaharuan data yang dilakukan dengan menggunakan aplikasi yang dibuat dan dengan menggunakan *script* PHP.

PHP/FI merupakan nama awl dari PHP. PHP – *Personal Home Page*, FI adalah *Form Interface*. Dibuat pertama kali oleh Rasmus Lerdorf. PHP awalnya merupakan program CGI yang dikhususkan untuk menerima input melalui form yang ditampilkan dalam peramban web. *Software* ini disebarkan dan dilisensikan sebagai perangkat lunak open source.

Integrasi PHP dengan server web dilakukan dengan teknisk CGI, FastCGI, dan modul *server web*. CGI (*Common Gateway Interface*) adalah suatu standar yang menghubungkan interface aplikasi eksternal dengan *server web*. Teknik CGI dan FastCGI memisahkan antara *server web* dan PHP; sedangkan modul *server web* menjadi PHP sebagai bagian dari *server web*.

Kini, PHP adalah kependekan dari *PHP:Hypertext Preprocessor* (rekursif, mengikut gaya penamaan di *nix), merupakan bahasa utama *script server side* yang disisipkan pada HTML yang dijalankan di *server* dan juga bisa digunakan untuk membuat aplikasi desktop.

2.2.18 *Framework* CodeIgniter

Codeigniter adalah sebuah *framework* PHP yang menggunakan arsitektur MVC. Codeigniter menggunakan pendekatan MVC agar bisa memisahkan logik aplikasi dan presentasi. Seperti yang sudah dijelaskan sebelumnya dimana sebuah

framework memiliki aturan tersendiri, berikut ini adalah aturan MVC yang diterapkan oleh Codeigniter [14] :

1. *Model*

   Model merepresentasikan struktur data. Biasanya *model* adalah sebuah *class* yang berinteraksi dengan sebuah basis data.

2. *View*

   View adalah informasi yang dipresentasikan ke pengguna. Biasanya *view* berisi halaman *web*, seperti sintaks HTML, tetapi di Codeigniter, *view* bisa berupa bagian dari HTML itu sendiri, misalnya *header* atau *footer*.

3. *Controller*

   Controller adalah sebuah *class* yang mengatur alur dari setiap permintaan pengguna.

Pada arsitektur MVC dengan platform *web*, dimana aplikasi *web* memiliki salah satu tolak ukur yaitu kecepatan akses, maka Codeiginter memiliki anomali atau perilaku yang berbeda dengan arsitektur MVC lainnya. Anomali itu diantaranya:

1. Anomali pertama, *Class Controller* adalah *class* yang pertama kali diakses oleh pengguna. Perilaku ini direpresentasikan pada skema URI (Uniform *Resource Information*) yang digunakan oleh Codeigniter. Contoh dari skema URI yang digunakan CodeIgniter adalah

   http://hostname/class/method/id

   Sehingga jika kita memiliki URL seperti

   http://if.unikom.ac.id/jadwal/semester/ganjil

   Sebenarnya kita mengakses *controller* jadwal dimana didalamnya terdapat *method* berama “semester” dan *method* “semester” memerlukan id “ganjil” untuk menampilkan data. Anomali ini akan berdampak pada *Sequence Diagram* yang akan diterapkan di bab 3, *dimana* aktor tidak berhubungan dengan *view* melainkan dengan *controller*. Sehingga anak panah akan menuju ke *class controller*. [28]
2. Anomali kedua, Codeigniter tidak secara tersurat menggunakan *class view* atau kita membuat sebuah *class* yang meng-*extend class view*. Codeigniter

menyediakan sebuah method khusus dimana method tersebut akan me-*load file view*/*file* berisi yang sintaks HTML. Berikut ini adalah sintaks untuk memanggil *file view* di Codeigniter

$*this*→*load*→*view*(‘file_*view*’);

Akan tetapi agar *view* tetap tercantum pada *class* diagram, *file view* bisa diasumsikan sebagai *class view*. [29]

3. Anomali ketiga, jika biasanya *class* model merepresentasikan ER Diagram, yang artinya bisa lebih dari 1 *class* model karena 1 tabel direpresentasikan dengan 1 *class* model maka di Codeiginter kita diperbolehkan menggunakan hanya 1 *class model* untuk mengelola 1 basis data. Alasan pertama adalah untuk kecepatan akses dan alasan kedua memudahkan dalam perawatan perangkat lunak. *Class model* yang digunakan cukup 1, bahkan dengan *library Query Builder* yang dimiliki Codeigniter kita diperbolehkan untuk tidak menggunakan *class model*, cukup *view* dan *controller* saja. [14]

Sebuah *class* model dapat mengelola 1 basis data dengan cara memanfaatkan parameter *method*. Parameter *method* bisa menggunakan *array* asosiatif atau objek dimana setiap elemen *array* menampung data-data yang jika dirangkai bisa membentuk sebuah *query* SQL. [30]

2.2.19 Basis Data MySQL [31]

MySQL adalah sebuah program database server yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi user serta menggunakan perintah dasar SQL (Structured Query Language).

MySQL memiliki dua bentuk lisensi, yaitu FreeSoftware dan Shareware. MySQL yang biasa kita gunakan adalah MySQL FreeSoftware yang berada di bawah lisensi GNU/GPL (General Public License). Selain itu, Anda juga dapat memiliki produk MySQL yang sifatnya komersial, biasa disebut dengan MySQL AB.

MySQL merupakan sebuah database server yang free, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya. MySQL pertama kali dirintis oleh seorang programmer

database bernama Michael Widenius. Selain sebagai database server, MySQL juga merupakan program yang dapat mengakses suatu database MySQL yang berposisi sebagai server. Pada saat itu berarti program kita berposisi sebagai client. Jadi MySQL adalah sebuah database yang dapat digunakan baik sebagai client maupun server.

Database MySQL merupakan suatu perangkat lunak database yang berbentuk database relational atau dalam bahasa basis data sering kita sebut dengan Relational Database Management System yang menggunakan suatu bahasa permintaan bernama SQL.

Terdapat perbedaan antara MySQL dan SQL. MySQL adalah sebuah program database, sedangkan SQL adalah bahasa perintah (query) dalam program MySQL. Selain MySQL ada beberapa program database server lain yang menggunakan standar query berupa SQL antara lain adalah Oracle, PostgreSQL dan SQL Server.

2.2.20 OpenSSL [32]

OpenSSL merupakan produk *open source* dengan lisensi yang berbeda dan tidak kompatibel dengan GPL. OpenSSL dapat diunduh dari website http://www.openssl.org. OpenSSL terdiri dari 3 komponen utama yaitu SSL/TLS *library*, *Crypto library*, dan openssl *command line tool*. SSL/TLS library memberikan fasilitas untuk melakukan sesi SSL/TLS dalam bentuk *loadable module* (DLL untuk Microsoft Windows). Fungsi API untuk SSL/TLS library dibagi menjadi 5 golongan:

1. Protocol methods, untuk menentukan versi SSL/TLS dan jenis layanan (*client*, *server*, atau *client-server*).
2. *Cipher*s, untuk menentukan berbagai jenis enkripsi yang dapat digunakan.
3. *Protocol contexts*, untuk menentukan context secara global selama program berjalan.
4. *Sessions*, untuk menentukan berbagai parameter per sesi.
5. *Connections*, untuk koneksi. Ini adalah fungsi utama SSL/TLS library, dan selama program berjalan fungsi dari golongan ini yang banyak digunakan.

Crypto library adalah implementasi dari berbagai algoritma enkripsi dan *hashing*. Selain digunakan oleh SSL/TLS library dan openssl *command line tool*, *crypto library* juga digunakan oleh OpenSSH, GnuPG dan implementasi standard kriptografi lainnya. Command line tool openssl adalah program yang berorientasi Unix yang dapat digunakan untuk:

1. Pembuatan dan manajemen kunci privat/publik.
2. Operasi kriptografi *public key* termasuk *key* management, digital signing dan *digital signature* checking.
3. Pembuatan dan manajemen *certificate* X.509.
4. Kalkulasi *digest*.
5. Enkripsi dan dekripsi.
6. Testing SSL/TLS *client* dan *server*.
7. S/MIME.
8. Membuat time-stamp request, membuat time-stamp dan melakukan verifikasi time-stamp.

2.2.21 Metode OCTAVE Allegro [33]

Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) mendefinisikan komponen-komponen penting secara komprehensif, sistematis, berbasis konteks (*context-driven*) evaluasi risiko keamanan informasi. Dengan menggunakan metode OCTAVE, organisasi dapat membuat perlindungan terhadap informasi berbasis pengambilan keputusan risiko berdasarkan CIA (*Confidentiality, Integrity, Authentication*) untuk aset teknologi informasi kritis.

OCTAVE merupakan metodologi untuk mengidentifikasi dan mengevaluasi risiko keamanan sistem informasi. Penggunaan OCTAVE ditujukan untuk membantu organisasi dalam hal:

a. Mengembangkan kriteria evaluasi risiko kualitatif yang menggambarkan toleransi risiko operasional organisasi.
b. Mengidentifikasi aset - aset penting untuk mencapai misi organisasi.
c. Mengidentifikasi kerentanan dan ancaman terhadap aset tersebut.

d. Menentukan dan melakukan evaluasi untuk menghadapi konsekuensi yang terjadi pada organisasi jika ancaman tersebut terjadi.

Metoda OCTAVE memiliki tiga varian yaitu OCTAVE, OCTAVE-S dan OCTAVE Allegro. OCTAVE merupakan seperangkat peralatan, teknik dan metode untuk penilaian dan perencanaan keamanan sistem informasi berbasis risiko. OCTAVE Allegro merupakan metode yang disederhanakan dengan fokus pada aset informasi. OCTAVE Allegro dapat dilakukan dengan metoda *workshop-style* dan kolaboratif. OCTAVE Allegro terdiri dari delapan langkah di bagi dalam empat fase.

1. Membangun Kriteria Pengukuran Risiko.

   Langkah ini terdapat dua aktivitas, diawali dengan membangun *organizational drivers* digunakan untuk mengevaluasi dampak risiko pada misi dan tujuan bisnis, serta mengenali *impact area* yang paling penting. Aktivitas 1 yaitu membuat definisi ukuran kualitatif yang di dokumentasikan pada *Risk Measurement Criteria Worksheets*. Aktivitas dua melakukan pemberian nilai prioritas *impact area* menggunakan *Impact Area Ranking Worksheet.*
2. Mengembangkan Profil Aset Informasi

   Terdiri dari delapan aktivitas, diawali dengan identifikasi aset informasi selanjutnya dilakukan penilaian risiko terstruktur pada aset yang kritis. Aktivitas tiga dan empat mengumpulkan i nformasi mengenai information aset yang penting di lanjutkan dengan membuat dokumentasi alasan pemilihan aset informasi kritis. Aktivitas lima dan enam membuat deskripsi aset informasi kritis kemudian mengidentifikasi kepemilikan dari aset informasi kritis tersebut. Aktivitas tujuh mengisi kebutuhan keamanan untuk *confidentiality*, *integrity* dan *availaibility*. Aktivitas delapan mengidentifikasi kebutuhan keamanan yang paling penting untuk aset informasi.

3. Mengidentifikasi Kontainer dari Aset Informasi

   Hanya ada satu aktivitas pada langkah tiga, perhatikan tiga poin penting terkait dengan keamanan dan konsep dari kontainer aset informasi yaitu cara aset informasi dilindung, tingkat perlindungan atau pengaman aset informasi dan kerentanan serta ancaman terhadap kontainer dari aset informasi.

4. Mengidentifikasi Area Masalah

   Aktivitas pada langkah empat yaitu diawali dengan pengembangan profil risiko dari aset informasi dengan cara bertukar pikiran untuk mencari komponen ancaman dari situasi yang mungkin mengancam aset informasi. Dengan berpedoman pada dokumen *Information Asset Risk Environment Maps* dan *Information Asset Risk Worksheet* maka dapat dicatat *area of concern*. Berpedoman pada dokumen *Information Asset Risk Worksheet* lakukan review dari kontainer untuk membuat *Area of Concern* dan mendokumentasikan setiap *Area of Concern*.

5. Mengidentifikasi Skenario Ancaman

   Aktivitas satu pada langkah lima yaitu melakukan identifikasi skenario ancaman tambahan pada aktivitas ini dapat menggunakan *Threat Scenarios Questionnaries*. Aktivitas dua melengkapi *Information Asset Risk Worksheets* untuk setiap *threat scenario* yang umum.

6. Mengidentifikasi Risiko

   Aktivitas satu pada langkah 6 menentukan *threat scenario* yang telah di dokumentasikan di *Information Asset Risk Worksheet* dapat memberikan dampak bagi instansi.

7. Menganalisis Risiko

   Aktivitas harus dilakukan mengacu pada dokumentasi yang terdapat pada *Information Asset Risk Worksheet*. Aktivitas satu dimulai dengan melakukan *review risk measurement criteria* dilanjutkan dengan aktivitas kedua menghitung nilai risiko relatif yang dapat digunakan untuk menganalisis risiko dan memutuskan strategi terbaik dalam menghadapi risiko.

8. Memilih Pendekatan Pengurangan Resiko

   Aktivitas satu pada langkah delapan yaitu mengurutkan setiap risiko yang telah diidentifikasi berdasarkan nilai risikonya. Hal ini dilakukan untuk membantu dalam pengambilan keputusan status mitigasi risiko tersebut. Aktivitas dua melakukan pendekatan mitigasi untuk setiap risiko dengan berpedoman pada kondisi yang unik di instansi tersebut.

2.2.22 *Unified Modified Language* (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, Men-spesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software*. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Berikut ini merupakan beberapa bagian dari UML [34]:

a. *Use Case Diagram*

   Menggambarkan sejumlah eksternal aktor dan hubungannya ke *use case* yang diberikan oleh sistem. Komponen utama pembentuk *use case* antara lain:

   - Aktor (*actor*), menggambarkan suatu hal diluar sistem yang berinteraksi dengan sistem.
   - *Use case* , kegiatan/aktivitas yang disiapkan oleh sistem. Menekankan pada "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem itu bekerja.
   - Hubungan (*link*), penjelasan tentang hubungan suatu komponen *use case* diagram dengan komponen lainnya.

   Pada *use case* terdapat *use case relationship* yang menunjukkan sifat *reusable, optional,* and bahkan menspesialisasikan *use case behavior* antar tiap *use case* yang ada. Terdapat 4 jenis *Relationship* pada *use case* yaitu Asosiasi, Generalisasi, <<*include*>> dan <<*extend*>>. [35]

1. Asosiasi

   Asosiasi digunakan untuk menggambarkan interaksi antara aktor dengan setiap *use case* tertentu. Relasi ini digambarkan sebagai garis penghubung aktor terhadap *use case* yang berelasi dengan aktor tersebut.

2. Generalisasi

   Suatu relasi antara *use case* umum (induk) dan *use case* yang lebih spesifik (anak). Relasi Generalisasi memungkinkan *use case* yang lebih spesifik memiliki perilaku (*behaviour*) yang sama dengan *use case* yang lebih umum atau bisa disebut *use case* induk. Relasi generalisasi digambarkan dengan anak panah segitiga. *Use case* yang terletak di sisi anak panah adalah *use case* induk dan yang terletak di sisi lainnya adalah *use case* yang mewarisi perilaku *use case* induk.

3. <<*include*>> *relationship*

   Relasi *include* memungkinkan terjadinya penambahan perilaku (*behaviour*) ke dalam *use case* awal yang pada dasarnya *use case* ini tidak dapat berdiri sendiri tanpa adanya penambahan *use case* , dan *use case* awal tidak akan lengkap tanpa adanya *use case* tambahan ini. *Use case* yang berada pada kepala anak panah adalah *use case* awal, dan pada sisi lain adalah *use case* penambah.

4. <<*extend*>> *relationship*

   Relasi *extend* memungkinkan terjadinya penambahan beberapa *behaviour* (tingkah laku) ke dalam *use case* awal yang pada dasarnya *use case* tersebut sudah dapat berdiri sendiri tanpa adanya penambahan. *Extend* digambarkan dengan anak panah yang mempunyai garis putus-putus. *Use case* yang berada pada kepala anak panah adalah *use case* awal dan yang berada di lain sisi adalah *use case* tambahan.

b. *Activity Diagram*

   Activity Diagram digunakan untuk menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainya seperti *use case*.

c. *Class Diagram*

Menggambarkan struktur dan deskripsi *class*, *package* (paket) dan objek beserta hubungan satu sama lain seperti *containment* (penahanan), pewarisan, asosiasi dan lain-lain.

d. *Sequence Diagram*

Sequence diagram menggambarkan kolaborasi dinamis antara sejumlah objek dan untuk menunjukan rangkaian pesan yang dikirim antara objek juga interaksi. Sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.