BAB II

TINJAUAN PUSTAKA

II.1. Definisi Kriptografi

Menurut Dony Ariyus Kriptografi merupakan suatu ilmu sekaligus seni yang digunakan untuk menjaga keamanan data dengan metode-metode tertentu, dan dilakukan oleh *cryptopher* (orang yang membuat kriptografi). Disebut ilmu karena di dalamnya terdapat metode (rumusan) yang yang digunakan dan disebut seni karena dalam membuat suatu teknik kriptografi itu sendiri merupakan ciri tersendiri dari si pembuat dan memerlukan cara khusus dalam mendisainnya, sedangkan *cryptoanalysis* adalah suatu ilmu dan seni membuka (*breaking*) *chipertext* dan orang yang melakukannya disebut *cryptanalyst*.

Kriptografi (*Cryptography*) berasal dari bahasa Yunani yaitu dari kata *Crypto* dan *Graphia* yang berarti penulisan rahasia. Kriptografi menurut teminologinya adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat yang lain.

II.1.1. Sejarah Kriptografi

Kriptografi mempunyai sejarah yang sangat menarik dan panjang. Kriptografi sudah digunakan 4000 tahun yang lalu yang dipekenalkan oleh orang-orang mesir untuk mengirim pesan ke pasukan militer yang berada di lapangan dan supaya pesan tersebut tertangkap oleh musuh.

Pada zaman Romawi kuno dikisahkan pada suatu saat, ketika Julius Caesar ingin mengirimkan satu pesan rahasia kepada orang Jendral di medan perang. Pesan tersebut harus dikirimkan melalui seseorang kurir, tetapi karena pesan tersebut mengandung rahasia, Julius Caesar tidak ingin pesan tersebut terbuka ditengah jalan. Di sini Julius Caesar memikirkan bagaimana mengatasinya yaitu dengan cara mengacak pesan tersebut menjadi suatu pesan yang tidak dapat dipahami oleh siapapun kecuali hanya dapat dipahami oleh Jenderalnya saja. Tentu sang Jendral telah diberi tahu sebelumnya cara membaca pesan yang teracak tersebut, karena telah mengetahui kuncinya. Yang dilakukan Julius Caesar adalah mengganti semua susunan alfabet dari a, b, c, & yaitu a menjadi d, b menjadi e, c menjadi f dan seterusnya. (Dony Ariyus, 2006: 9-10).

II.1.2. Istilah-Istilah Dalam Kriptografi

Di dalam melakukan suatu proses kerjanya, terdapat banyak istilah-istilah yang cendrung dipergunakan. Untuk memudahkan pemahaman akan materi yang akan dibahas tentang kriptografi, maka terlebih dahulu istilah-istilah tersebut harus dipahami.

1. *Plaintext*

Plaintext merupakan pesan/data asli yang belum disandikan atau informasi yang ingin dikirim dan dijaga keamanannya. *Plaintext* tidak harus berupa teks, namun dapat berupa *file* gambar (**.gif,*.jpg*), *fil*e *biner* (**.exe,*.doc,*.docx*), *file* suara (**.wav,*.mp3*), dan sebagainya.

2. *Ciphertext*

Ciphertext merupakan pesan yang telah disandikan (dikodekan) sehingga siap untuk dikirimkan. *Ciphertext* inilah yang biasanya dikirimkan melalui saluran internet yang memiliki tingkat penyadapan yang tinggi. *Ciphertext* harus dapat dirancang sedemikian rupa agar tidak dapat diakses oleh pihak-pihak yang tidak berhak, sehingga pesan yang dikirim dapat terjamin keamanannya.

3. Enkripsi

Enkripsi merupakan hal yang sangat penting dalam kriptografi yang merupakan pengamanan data yang dikirimkan terjaga rahasianya. Pesan asli tersebut disebut *plaintext* yang dirubah menjadi kode-kode yang tidak dimengerti

4. Dekripsi

Dekripsi merupakan kebalikan dari enkripsi, pesan yang telah dienkripsi dikembalikan kebentuk asalnya (*plaintext*) disebut dengan dekripsi pesan (Dony Ariyus, 2006: 13)

5. *Kriptosistem*

Kriptosistem merupakan sistem yang dirancang untuk mengamankan suatu pesan atau informasi dengan memanfaatkan ilmu kriptografi. Suatu sistem kriptogafi (*kriptosistem)* bekerja dengan cara menyandikan suatu pesan menjadi suatu kode tertentu yang dimengerti oleh pembuat *kriptositem* tersebut. Pada dasarnya mekanisme kerja semacam ini telah dikenal sejak zaman dahulu oleh bangsa Mesir sekitar 4000 tahun yang lalu bahkan telah mempraktekkannya dengan cara yang sangat *primitif*. Dalam era teknologi informasi sekarang, mekanisme yang sama masih digunakan tetapi implemetasi sistemnya berbeda.

6. Algoritma Kriptografi

Berdasarkan kunci yang dipakai, algoritma kriptografi dibagi menjadi tiga macam.

1. *Hash Function*

Fungsi *hash* sering disebut sebagai fungsi satu arah *(one-way function*). Fungsi ini mengubah suatu *input* menjadi *output,* tetapi *output* tersebut tidak dapat dikembalikan menjadi bentuk semula. Salah satu manfaatnya adalah penggunaan sidik jari *(fingerprint*). Sidik jari digunakan sebagai identitas pengirim pesan. Fungsi lain adalah untuk kompresi dan message *digest*. Contoh algoritma fungsi ini adalah MD-5 dan SHA.

2. Asimetri

Pada algoritma ini, digunakan dua buah kunci yang berhubungan yang disebut dengan kunci umum dan kunci pribadi. Kunci umum dapat dipublikasikan sehingga pesan dapat dienkripsikan tetapi tidak dapat didekripsikan dengan kunci tersebut. Kunci pribadi hanya boleh digunakan oleh pihak yang berhak untuk mendekripsikan pesan yang terenkripsi. Algoritma yang menggunakan kunci umum dan publik ini antara lain *Digital Signature Algorithm* (DSA), *Rivest-Shamir-Adleman* (RSA), *Diffie-Hellman* (DH), dan sebagainya.

3. Simetri

Algoritma ini menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data. Untuk mendekripsikan data, penerima menggunakan kunci yang sama dengan kunci yang digunakan pengirim untuk mengenkripsi data.

Contoh dari algoritma ini adalah *Data Encryption Standard* (DES), *International Data Encryption Algorithm* (IDEA) (Bernardino; 2005; 1)

II.2. Algoritma AES

Advanced Encryption Standard (AES) merupakan algoritma cryptographic yang dapat digunakan untuk mengamakan data. Algoritma AES adalah blok chipertext simetrik yang dapat mengenkripsi (*encipher*) dan dekripsi (decipher) infoermasi. Enkripsi merubah data yang tidak dapat lagi dibaca disebut *ciphertext* sebaliknya dekripsi adalah merubah *ciphertext* data menjadi bentuk semula yang kita kenal sebagai plaintext. Algoritma AES is mengunkan kunci kriptografi 128, 192, dan 256 bits untuk mengenkrip dan dekrip data pada blok 128 bits.

AES (*Advanced Encryption Standard*) adalah lanjutan dari algoritma enkripsi standar DES (*Data Encryption Standard*) yang masa berlakunya dianggap telah usai karena faktor keamanan. Kecepatan komputer yang sangat pesat dianggap sangat membahayakan DES, sehingga pada tanggal 2 Maret tahun 2001 ditetapkanlah algoritma baru Rijndael sebagai AES. Kriteria pemilihan AES didasarkan pada 3 kriteria utama yaitu : keamanan, harga, dan karakteristik algoritma beserta implementasinya. Keamanan merupakan faktor terpenting dalam evaluasi (minimal seaman triple DES), yang meliputi ketahanan terhadap semua analisis sandi yang telah diketahui dan diharapkan dapat menghadapi analisis sandi yang belum diketahui. Di samping itu, AES juga harus dapat digunakan secara bebas tanpa harus membayar royalti, dan juga

murah untuk diimplementasikan pada smart card yang memiliki ukuran memori kecil. AES juga harus efisien dan cepat (minimal secepat Triple DES) dijalankan dalam berbagai mesin 8 bit hingga 64 bit, dan berbagai perangkat lunak. DES menggunakan stuktur Feistel yang memiliki kelebihan bahwa struktur enkripsi dan dekripsinya sama, meskipun menggunakan fungsi F yang tidak invertibel. Kelemahan Feistel yang utama adalah bahwa pada setiap ronde, hanya setengah data yang diolah. Sedangkan AES menggunakan struktur SPN (Substitution Permutation Network) yang memiliki derajat paralelisme yang lebih besar, sehingga diharapkan lebih cepat dari pada Feistel.

Kelemahan SPN pada umumnya (termasuk pada Rijndael) adalah berbedanya struktur enkripsi dan dekripsi sehingga diperlukan dua algoritma yang berbeda untuk enkripsi dan dekripsi. Dan tentu pula tingkat keamanan enkripsi dan dekripsinya menjadi berbeda. AES memiliki blok masukan dan keluaran serta kunci 128 bit. Untuk tingkat keamanan yang lebih tinggi, AES dapat menggunakan kunci 192 dan 256 bit. Setiap masukan 128 bit plaintext dimasukkan ke dalam state yang berbentuk bujursangkar berukuran 4×4 byte. State ini di-XOR dengan key dan selanjutnya diolah 10 kali dengan subtitusi-transformasi linear-Addkey. Dan diakhir diperoleh ciphertext. Berikut ini adalah operasi Rijndael (AES) yang menggunakan 128 bit kunci:

- Ekspansi kunci utama (dari 128 bit menjadi 1408 bit)
- Pencampuran subkey.

• Ulang dari i=1 sampai i=10 Transformasi : ByteSub (subtitusi per byte) ShiftRow (Pergeseren byte perbaris) MixColumn (Operasi perkalian GF(2) per kolom)

• Pencampuran subkey (dengan XOR)

• Transformasi : ByteSub dan ShiftRow

• Pencampuran subkey Kesimpulan yang didapat adalah :

• AES terbukti kebal menghadapi serangan konvensional (linear dan diferensial attack) yang menggunakan statistik untuk memecahkan sandi.

• Kesederhanaan AES memberikan keuntungan berupa kepercayaan bahwa AES tidak ditanami trapdoor.

• Namun, kesederhanaan struktur AES juga membuka kesempatan untuk mendapatkan persamaan aljabar AES yang selanjutnya akan diteliti apakah persamaan tersebut dapat dipecahkan

• Bila persamaan AES dapat dipecahkan dengan sedikit pasangan plaintext/ciphertext, maka riwayat AES akan berakhir.

• AES didesain dengan sangat hati-hati dan baik sehingga setiap komponennya memiliki tugas yang jelas

• AES memiliki sifat cipher yang diharapkan yaitu : tahan menghadapi analisis sandi yang diketahui, fleksibel digunakan dalam berbagai perangkat keras dan lunak, baik digunakan untuk fungsi hash karena tidak memiliki weak(semi weak) key, cocok untuk perangkat yang membutuhkan key agility yang cepat, dan cocok untuk stream cipher.

II.2.1. Sejarah AES

Pada rahun 1997, *National Institute of Standard and Technology* (NIST) *of United States* mengeluarkan *Advanced Encryption Standard* (AES) untuk menggantikan Data Encryption Standard (DES). AES dibangun dengan maksud untuk mengamankan pemerintahan diberbagai bidang. Algoritma AES di design menggunkan blok chiper minimal dari blok 128 bit input dan mendukung ukuran 3 kunci (3-key-sizes), yaitu kunci 128 bit, 192 bit, dan 256 bit. Pada agustus 1998, NIST mengumumkan bahwa ada 15 proposal AES yang telah diterima dan dievaluasi, setelah mengalami proses seleksi terhadap algoritma yang masuk, NIST menumumkan pada tahun 1999 bahwa hanya ada 5 algoritma yang diterima, algoritma tersebut adalah :

1. MARS
2. RC6
3. Rijndael
4. Serpent
5. Twofish

Algoritma-algoritma tersebut manjalani berbagai macam pengetesan. Pada bulan oktober 2000, NIST mengumumkan bahwa Rijndael sebagai algoritma yang terpilih untuk standar AES yang baru. Baru pada februari 2001 NIST mengirimkan draff kepada Federal Information Processing Standards (FIPS) untuk standar AES. Kemudian pada 26 November 2001, NIST mengumumkan produk akhir dari Advanced Encryption Standard.

II.3. UML (*Unifield Modelling language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia perkembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi perkembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan dengan baik (Munawar ; 2005 : 17).

UML merupakan kesatuan bahasa pemodelan yang dikembangkan oleh *Booch, Object Modeling Technique* (OMT) dan *Object Oriented Engineering* (OOSE). Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan interatif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode *Booch* adalah pada detail dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstuktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, disain sistem, desain objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan dan *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, desain dan implementasi dan model pengujian (test Model). Keunggulan

metode ini adalah mudah dipelajari karena memiliki notaasi sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode *Booch*, OMT dan OOSE digabungkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya. Unsur-unsur yang membentuk UML ditunjukkan dalam Gambar II.1.

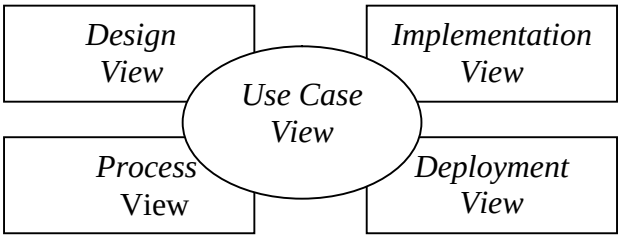
Gambar II.1 Unsur-unsur yang membentuk UML

Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 18

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam OOAD (*Object Oriented Analysis* dan *Design*). UML tidak hanya domain dalam penotasian dilingkungan OO tetapi juga populer di luar lingkungan OO. Ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan bahasa *pemrograman*. Sebagai sebuah sketsa

UML bisa berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detil tentang coding program (*Forward engineering*) atau bahkan membaca program dan mengimplementasikannya kembali ke dalam *diagram* (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokkumentasi asli hilang atau bahkan belum perna dibuat sama sekali. Sebagai bahasa pemrograman, UML dapat diterjemahkan diagram yang ada di UML menjadi kode program siap untuk dijalankan.

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahhwa struktur sebuah sistem dideskripsikan dalam *view* dimana salah satu diantaranya *use case view*. *Use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain. Model 4+1 *view* ditunjukkan pada gambar II.2



Gambar II.2 Model 4+1 *View*

Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 20

Kelima *view* tersebut tidak berhubungan dengan *diagram* yang dideskripsikan di UML. Setiap *view* berhubungan dengan perspektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut.

Use case view mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tarik bagi *end user*, analis dan *tester*. Pandangan ini mendefenisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari peran dan sering dikatakan yang mendrive proses perkembangan perangkat lunak.

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi definisi komponen program, *class-class* utama bersama-sama dengan spesifikasi *data*, perilaku dan interaksinya. Informasi yang terkandung di *view* menjadi pergantian para progremer karena menjelaskan secara detil bagaimana fungsionalitas *sistem* akan diimplementasikan.

Implementasi *view* menjelaskan komponen-komponen *visi* yang akan dibangun. Hal ini berbeda dengan komponen *logic* yang dideskripsikan pada *design view*. Termasuk disini diantaranya *file exe*, *library* dan *database*. Informasi yang ada di *view* dan integrasi sistem.

Proses *view* berhubungan dengan hal-hal yang berkaitan dengan *concurrency do* dalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti jaringan komputer dimana sistem akan dijalankan. Kedua *view* ini menunjukan kebutuhan *non fungsional* dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja (Munawar;2005:17-21).

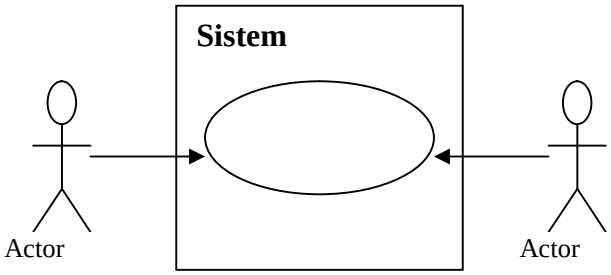
II.3.1 *Use Case Diagram*

Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain *object* dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*.

Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system / sub system boundary. Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Ilustrasi *actor, use case* dan *system* ditunjukkan pada gambar II.3



Gambar II.3 *Use Case* Diagram

Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 64

Untuk mengidentifikasi *actor*, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks targer sistem. *Actor* adalah *abstraction* dari orang dan sistem yang lain mengaktifkan fungsi dari target sistem. Orang atau sistem bila muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*.

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. Oleh kerena itu sangat penting untuk memilih abstraksi yang cocok. Use case dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ’apa’ yang dikerjakan *software* aplikasi, bukan ’bagaimana’ *software* aplikasi mengerjakannya. Setiap *use case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Namun *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama (Munawar ; 2005 : 63-66).

II.3.2 *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Kelas memiliki apa yang disebut *Atribut* dan metode atau operasi :

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Susunan kelas suatu sistem yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main, kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem, kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case*, kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
4. Kelas yang diambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

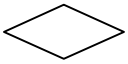
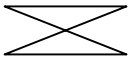
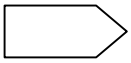
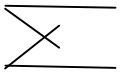
Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke *basis data*, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohension* dan *coupling*. *Cohension* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah (Rosa A.S dan M. Shalahuddin ; 2011 : 122-123).

II.3.3 *Activity Diagram*

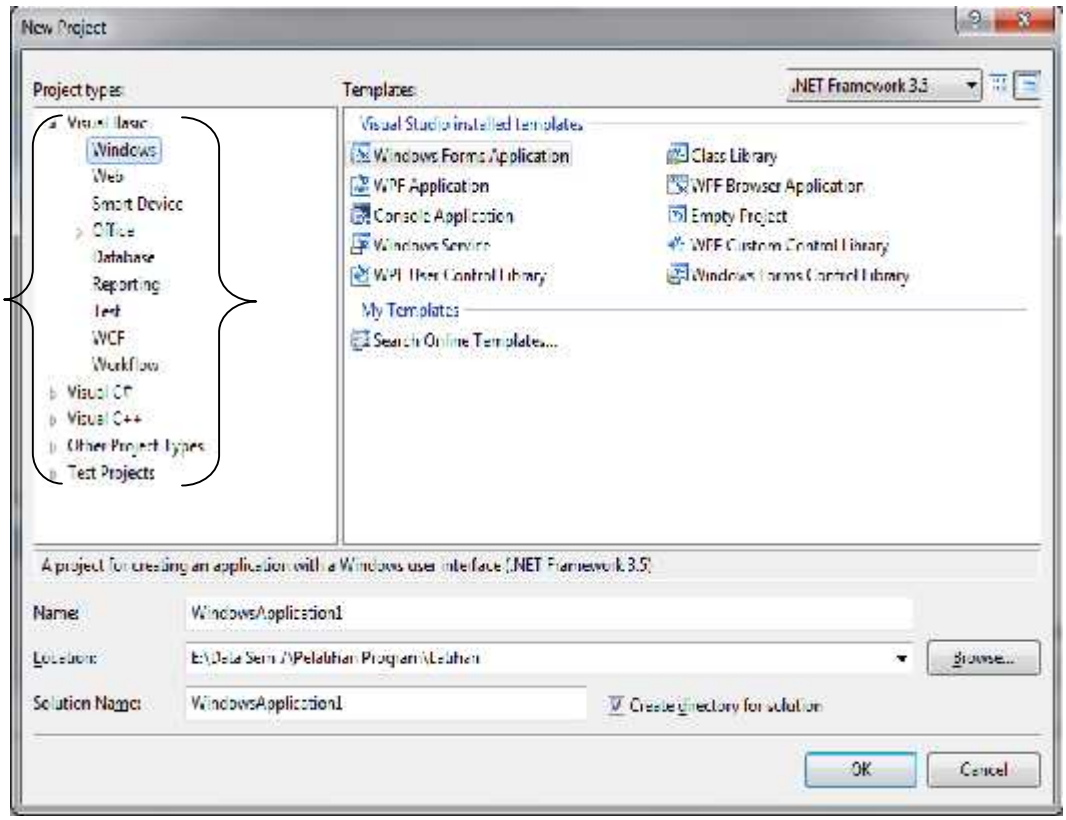
Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa (Munawar ; 2005 : 87).

Tabel II.1 Simbol *Activity Diagram*

Gambar	Nama
	Titik awal
	Titik akhir
	*Activity*
	Pilihan untuk pengambilan keputusan
	*Fork*, digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan parallel menjadi satu
	*Rake*, menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	*Flow final*

II.4. Pengenalan *Visual Basic. NET*

Visual Basic.Net dilihat dari sejarahnya merupakan pengembangan dari bahasa *Basic*, sehingga aturan penulisan bahasanya pun sama dengan bahasa *Basic*. Akan tetapi, oleh karena adanya tuntutan perkembangan teknologi maka bahasa *Visual Basic.Net* memiliki beberapa tambahan yang tidak ada di bahasa *Basic* aslinya (Wahana Komputer, 2006 ; hal 1).

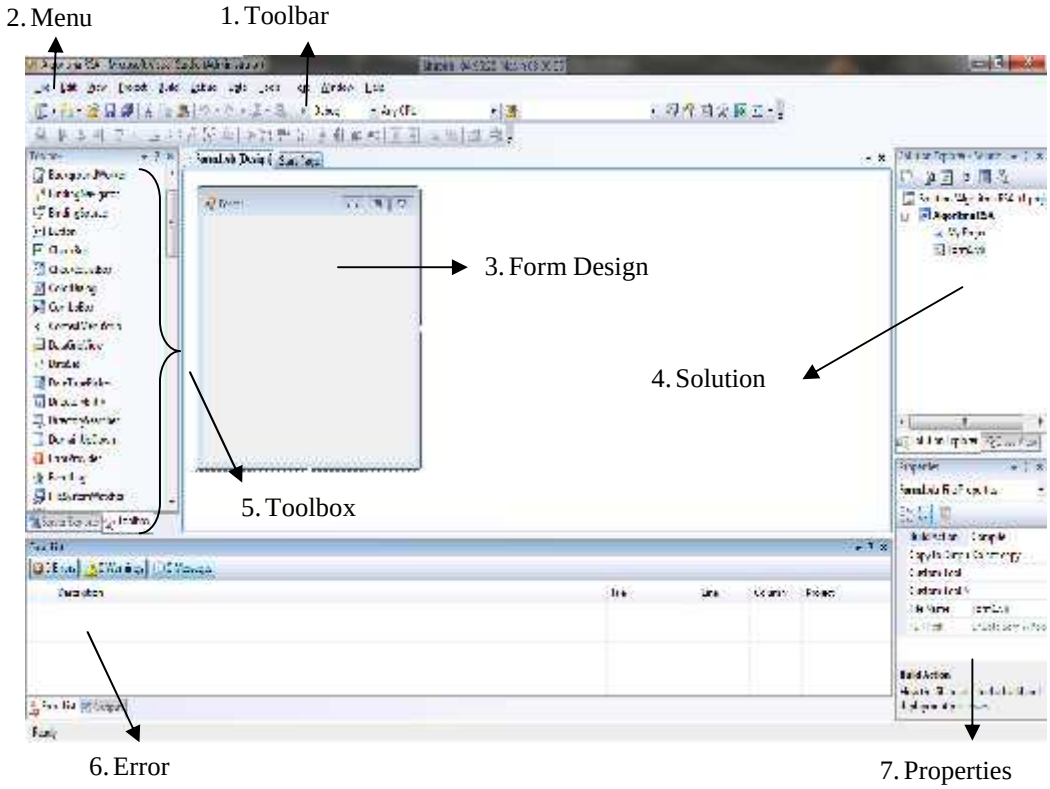


Gambar II.4. Jendela *New Project VB 2008*
Sumber (Muhammad Sadeli ; 2009 ; 5)

1. ***Windows*** adalah *project* yang sering digunakan untuk membangun aplikasi-aplikasi *dekstop* seperti (membuat aplikasi *dekstop*, komponen *ActiveX*, *file* DLL, dan lain sebagainya). Karena menggunakan *interface windows* baik

command line ataupun *windows form* yang memiliki *form* dan kontrol, yang terbaru di *VB 2008* adalah WPF (*Windows Presentation Foundatio*n (*Windows, Web*) XAML) yang memugkinkan suatu pekerjaan GUI (*Grafik User Interface*) dan kode untuk program dibuat secara terpisah.

2. ***Web*** adalah *project* yang dapat digunakan untuk membuat aplikasi berbasis *web* menggunakan *ASP.Net 3.5*.
3. ***Smart Device*** digunakan untuk membuat perangkat aplikasi pemograman yang dapat diimplimentasikan pada *Handphone* tertentu seperti PDA (*Personal Digital Assistant*). Pemograman *Smart Device* pada *VB 2008* menggunakan *.NET Compact Framework* 3.5 dan berjalan diatas sistem *Windows CE* (*Compact Edition*).
4. ***Office*** adalah suatu *project* yang dapat menyediakan atau menjalankan atau memanggil aplikasi yang terdapat pada program *Office* (2003, 2007) seperti *Excel, Word, PowerPoin*, dan lain sebagainya.



Gambar II.5 Halaman Kerja *Visual Basic* 2008
Sumber (Muhammad Sadeli ; 2009 ; 7)

1. ***Menu Bar*** adalah suatu menu yang terdiri dari 11 menu utama, masing-masing memiliki *sub menu* dan perintah lengkap dengan *shortcut key*.
2. ***Toolbar Standart*** adalah suatu baris menu yang mempunyai fungsi yang sama pada setiap *Tool Standart* pada umumnya, seperti fungsi untuk menyimpan, meng-*copy*, menambah *project* baru, mengatur tampilan program dan masih banyak lagi.
3. ***Form Design*** adalah suatu lembar *form* yang berfungsi untuk merancang tampilan aplikasi secara visual dengan menempatkan kontrol-kontrol yang diperlukan.

4. ***Toolbox*** adalah suatu jendela yang berfungsi untuk menampung komponen-komponen *standard*.
5. ***Solution Explorer*** adalah suatu jendela yang berfungsi untuk menampilkan *object* yang digunakan untuk membuat aplikasi seperti : *form, class,* dan *object* lainnya.
6. ***Properties Windows*** adalah suatu jendela yang berfungsi untuk mengatur nilai *properties* dari masing-masing komponen yang akan digunakan.
7. ***Error List*** adalah suatu jendela yang berfungsi untuk menampilkan setiap kesalahan dari pembuatan kode program suatu aplikasi. (Muhammad Sadeli ; 2009 ; 7 - 8)