BAB II

TINJAUN PUSTAKA

Bab ini akan membahas mengenai tinjauan pustaka yang berisi pustaka dan hasil penelitian yang pernah dilakukan, yang mana isi pustaka berhubungan dengan penelitian ini. Landasan teori membahas mengenai teori-teori dasar yang mendukung penelitian ini.

2.1. Tinjauan Pustaka

Perkembangan teknologi dan keleluasan akses jaringan internet saat ini bukan merupakan hal baru di kalangan masyarakat dunia. Internet merupakan salah satu media komunikasi global, sehingga perlu adanya sebuah keamanan yang mendasar mengingat informasi merupakan bagian penting dalam sebuah proses. Kriptografi merupakan teknik pengaman data dengan menggunakan metode enkripsi dan dekripsi. (Elminaam, 2010). Kriptografi adalah ilmu yang mempelajari tentang metode untuk membuat tulisan atau pesan rahasia. Pada transformasi ini terdapat dua macam masalah keamanan data, yaitu masalah privasi dan keotentikan. Privasi mengandung arti bahwa data yang dikirimkan hanya dapat dimengerti informasinya oleh penerima yang berhak. Sedangkan keotentikan mencegah pihak ketiga untuk mengirim data yang salah atau mengubah data yang dikirimkan (Foelyati, 2007).

Berbicara mengenai sebuah teknik pengamanan tentunya tidak terlepas dari bagaimana tahap-tahap untuk mengamankan sebuah data yang biasa disebut algortima. Dalam konteks ilmu matematika dan komputasi, algortima merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini

dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan setiap masalah ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria. Algoritma sering mempunyai langkah perulangan atau memerlukan keputusan sampai tugasnya selesai.

Pesatnya pertumbuhan perangkat elektronik *portable* dengan daya yang terbatas, memaksa para pengembang aplikasi perangkat *mobile* berupaya untuk mendesain aplikasi-aplikasi *mobile* yang tidak terlalu mengkonsumsi daya tinggi agar dapat menghemat daya *battery* yang terbatas tersebut. Telepon genggam, *PDA* dan perangkat *mobile (smartphone)* lainnya adalah contoh produk elektronik *portable* dan sangat rentan akan keamanan data. Oleh karena itu, untuk keamanan data menggunakan proses enkripsi menggunakan algoritma jenis asimetris tentunya membutuhkan *resource* perangkat *mobile* yang cukup memadai, sehingga kinerja algoritma *DES, AES* dan *ECC* menjadi optimal (Kumar, 2012).

Kriptografi yang muncul pertama kali pada masa kerajaan mesir, sekitar 4000 tahun yang lalu yang diperkenalkan oleh masyarakat mesir lewat *hieroglyph*. Dikisahkan pada zaman romawi kuno, pada saat *julius caesar* ingin mengirimkan pesan rahasia kepada seorang jendral di medan perang. Pesan tersebut harus dikirimkan melalui seorang kurir. Karena pesan tersebut mengandung rahasia, *julius* tidak ingin pesan tersebut sampai diketahui saat di jalan. *Julius caesar* kemudian mengacak pesan tersebut sehingga menjadi suatu pesan yang tidak dapat dipahami oleh siapapun terkecuali jendralnya saja (Dony Ariyus, 2008).

Pada saat itulah kriptografi lahir sebagai metode untuk mengamankan pesan rahasia.

Pada penelitian-penelitian sebelumnya di bidang teknologi informasi, menunjukkan bahwa yang menjadi masalah fundamental yang sedang dihadapi para perusahan-perusahan perangkat *mobile* saat ini yaitu masalah akan daya *battery*. Karenanya ini mempunyai hubungan yang sangat relevan dengan perangkat lunak atau aplikasi-aplikasi yang beroperasi pada peragkat *mobile* tersebut (Elminaan, 2010). Pada penelitian lainnya teknik kriptografi ini dimanfaatkan untuk mengamankan layanan pemesanan dan laporan penjualan. Berhubung karena adanya pertukaran data secara *online* antara *dealer* satu dengan yang lain, maka penerapan teknik kriptografi menggunakan algoritma *IDEA* diharapkan dapat menjaga keamanan data (Wardani, 2010).

Pada tahun 2009, Andriyanto, melakukan Studi perbandingan algoritma *IDEA* dan *Blowfish* untuk membandingkan kinerja algoritma *IDEA* dan *Blowfish* dalam hal kecepatan proses dan penggunaan memori pada saat proses enkripsi dan dekripsi suatu *file*. Selain itu implementasi algoritma enkripsi dan dekripsi *Twofish* pada perangkat *mobile* yang berbaasis *android* digunakan untuk mengamankan *Short Message Service* (SMS) (Wijaya, 2011). Dalam hal komunikasi data antar komputer maupun tranfer *file* dari satu *device* ke *device* yang lain, keamanan data sangatlah penting. Oleh karena itu penggunaan algoritma kriptografi *ElGamal*, *Grain V1* dan *AES* untuk proses enkripsi dan dekripsi terhadap resep masakan (Zulhazmi, 2012).

Keamanan data privasi mendapat ancaman besar khususnya data *video*. Oleh karena itu pengamanan kriptografi mempunyai peran penting dalam hal ini. Sehingga penilitian tentang perbandingan algoritma kriptogarfi *DES, 3DES* dan *AES* perlu dilakukan untuk mengetahui hasil algoritma yang tepat digunakan untuk proses enkripsi sebuah data *video*. Pada penelitian ini sesuai dengan faktor-faktor yang telah ditentukan, algoritma *AES* terbukti lebih baik dari *DES* dan *3DES* (Alanazi, 2010). Pada tahun 2010, Thulasimani, Sumber daya (*battery)* sangat berpengaruh pada aktivitas sebuah perangkat *mobile.* Pada penelitian kali ini, algotima *AES* dipilih sebagai algoritma yang akan menangani proses enkripsi dan dekripsi. Melalui beberapa hasil uji coba, algoritma *AES* bekerja dengan panjang kunci yang berbeda-beda yaitu 128, 192 dan 256 yang digunakan mampu menghemat konsumsi daya baterai pada perangkat *mobile.*

Untuk saat ini keamanan akan bertransaksi secara online membutuhkan tingkat keamanan yang sangat tinggi, karena menyangkut data-data privasi nasabah. Dengan *SMS Banking* yang dirancang menggunakan algortima *AES* diharapkan dapat meminimalisir celah-celah keamanan akses data privasi nasabah. Sehingga saat bertransaksi menggunakan aplikasi *SMS Banking* nasabah pun merasa *safety* (Manoj, 2011).

Selain itu pada penelitian sebelumnya pertimbangan penggunaan *resources* berupa *CPU Time, memory* dan daya baterai sangat diperhitungkan, mengingat terbatasnya daya baterai yang tidak sepadan dengan penggunaan aplikasi pada perangkat *mobile.* Sehingga pada pada tahun 2010, Elminaam, membuat penelitian tentang perbandingan kinerja algoritma simetris untuk proses

enkripsi menggunakan algortitma *AES, DES, 3DES, RC2, Blowfish* dan *RC6* untuk proses eknripsi *file text, file audio* dan *file video*. Hasil dari penelitian ini adalah dalam hal perubahan ukuran paket, disimpulkan bahwa algoritma *Blowfish* memiliki kinerja terbaik dalam kecepatan enkripsi. Kinerja algoritma *3DES* masih lebih rendah dibandingkan dengan algoritma *DES*. Akhirnya dalam hal perubahan ukuran kunci dapat dilihat bahwa ukuran kunci lebih mempengaruhi pada konsumsi baterai dan waktu.

Tabel 2.1 Perbandingan Penelitian

No.	Penelitian	Tujuan	Metode	Hasil
1.	Diaa Salama Abd Elminaam, Hatem Mohamed Abdual Kader dan Mohiy Mohamed Hadhoud, 2010. *Evaluating The Performance of Symmetric Encryption Algorithms*. Vol. 10, pp. 216-222.	Agar dapat meminimalisir penggunaan daya baterai. Untuk proses enkripsi file menggunakan algoritma AES, DES, 3DES, RC2, Blowfish dan RC6.	Melakukan uji coba perbandingan algoritma AES, DES, 3DES, RC2, Blowfish dan RC6.	Algoritma *Blowfish* memiliki kinerja terbaik dalam kecepatan enkripsi. Kinerja algoritma *3DES* masih lebih rendah dibandingkan dengan algoritma *DES*. Akhirnya dalam hal perubahan ukuran kunci dapat dilihat bahwa ukuran kunci lebih mempengaruhi pada konsumsi baterai dan waktu.
2.	Hamdan .O. Alanazi, B.B. Zaidan, A.A. Zaidan, Hamid A. Jalab, M. Shabbir dan Y. Al-Nabhani, 2010. *New Comparative Study Between DES, 3DES and AES within Nine Factors*. Vol.2, ISSUE 3, ISSN 2151-9617.	Meningkatkan keamanan pada file atau informasi yang bersifat privacy dan untuk mengetahu algoritma yang tepat yaitu antara *AES*, *DES* dan *3DES*.	Melakukan uji coba perbandingan algoritma *AES*, *DES* dan *3DES*.	Pada penelitian ini algoritma DES, AES, 3DES disajikan dengan sembilan faktor yaitu panjang kunci, jenis chiper, ukuran blok dan keamanan untuk pengujian kinerja masing-masing algoritma. Algoritma AES terbukti lebih baik dari DES dan 3DES dalam hal optimasi kinerja untuk proses enkripsi.
3.	Mani Arora, Dr. Derick Engeles, 2011. *Analysis of Chiper Text Size Produced by Various Encryption*	Penggunaan algoritma enkripsi simetris dan untuk mengetahui diantara Algortitma SDES, DES,	Melakukan uji coba enkripsi masing-masing algoritma.	Algoritma SDES terbukti paling lemah dan algoritma CAST-128 terbukti palikng kuat di antara algoritma-algoritma yang

	Algorithms. Vol. Vol No : 3, ISSN : 0975-5462.	2DES, 3DES, IDEA, Blowfish, RC5, dan RC2 dan CAST-128 yang lebih optimal untuk kebutuhan enkripsi.		dibandingkan.
4.	V. Rajesham Goud, Md. Hameed Pasha, 2011. *Textual Encryption Using Conventional Encryption Algorithm.* Vol. 2, Issue 2, ISSN 2231-2804.	Mengamankan *file text* menggunakan algoritma IDEA untuk proses enkripsi.	Melakukan uji coba enkripsi *file text* terhadap algoritma IDEA.	Algoritma IDEA terbukti berjalan optimal untuk proses enkripsi *file text.*
5.	L. Thulasimani, M. Madheswaran, 2010. *Design and Implementation of Reconfigurable Rijndael Encryption Algorithms for Reconfigurable Mobile Terminal*. Vol. 2, ISSN : 0975-3397.	Mengoptimalkan kinerja algortima enkripsi pada perangkat *mobile.*	Melakukan uji coba algoritma enkripsi AES dengan panjang kunci 128, 256 dan 192 bit.	Algoritma AES 128, 256 dan 192 bit dapat digunakan untuk proses enkripsi yang diimplementasikan untuk jaringan nirkabel.

2.2. Landasan Teori

2.2.1. Kriptografi secara umum

Kriptografi adalah ilmu yang memoelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan dari pengirim ke penerima tanpa mengalami gangguan dari pihak ketiga yang tidak bertanggung jawab (Goud, 2011). Menurut *Bruce Scheiner* dalam bukunya *Applied Cryptography,* kriptografi adalah ilmu pengetahuan dan seni menjaga pesan-pesan agar tetap aman *(secure)*.

2.2.2 Algoritma secara umum

Ditinjau dari asal-usulnya, kata algoritma mempunyai sejarah yang menarik. Kata ini muncul di dalam kamus *webster* sampai akhir tahun 1957. Kata *algorism* mempunyai arti proses perhitungan dalam bahasa Arab. Penemunya adalah seorang ahli matematika dari *persia* yang bernama Abu Ja'far Muhammad Ibnu Musa al-Khuwarizmi (770-840). Di dalam dunia literatur barat dia lebih terkenal dengan sebutan *Algorizm*. Panggilan inilah yang kemudian lambat laun berubah menjadi *algorithm* (Heriyanto, 2011). Dalam bahasa Indonesia, kita kemudian menyebutkannya algoritma. Defenisi algoritma adalah logika, metode dan tahapan atau urutan sistematis yang digunakan untuk memcahkan seatu permasalahan.

Dalam beberapa konteks, algoritma adalah spesifikasi urutan langkah untuk melakukan pekerjaan tertentu. Pertimbangan dalam pemilihan algoritma adalah pertama, algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli

sebagus apapun algoritma, kalau memberikan keluaran yang salah pastilah algoritma tersebut bukanlah algoritma yang baik.

Pertimbangan kedua yang harus diperhatikan adalah kita harus mengetahui seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil (hasil yang hanya berupa pendekatan). Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

Pertimbangan ketiga adalah efisiensi algoritma. Efisiensi Algoritma dapat ditinajau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar (paling mendekati), tetapi jika kita harus menunggu berjam-jam untuk mendapatkan hasilnya, algoritma tersebut biasanya tidak akan dipakai. Setiap orang menginginkan keluaran yang cepat. Begitu juga dengan memori, semakin besar memori yang terpakai maka semakin tidak baik algoritma tersebut (Sutrisno).

2.2.3. Algoritma Kriptografi

Defenisi terminologi algoritma adalah urutan langkah-langkah logis untuk menyelesaikan masalah yang disusun secara sistematis. Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang tidak berhak atas pesan tersebut. Algoritma kriptografi dari tiga fungsi dasar yaitu :

a) Enkripsi : merupakan hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirimkan agar terjaga kerahasiannya. Pesan asli

disebut *plaintext*, yang diubah menjadi kode-kode yang tidak dimengerti atau *chipertext*.

b) Dekripsi : merupakan kebalikan dari enkripsi, Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (teks asli), disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan algoritma yang digunakan untuk enkripsi.

c) Kunci : yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, kunci rahasia *(private key)* dan kunci umum *(public key)*.

Keamanan dari algoritma kriptografi tergantung pada bagaimana algoritma itu bekerja. Oleh sebab itu algoritma semacam ini disebut dengan algoritma terbatas. Algortima terbatas merupakan algoritma yang diapakai sekelompok orang untuk merahasiakan pesan yang mereka kirim. Keamanan kriptografi moderen didapat dengan merahasiakan kunci yang dimiliki dari orang lain, tanpa harus merahasiakan algoritma itu sendiri. kunci memiliki fungsi yang sama dengan *password*. Jika keseluruhan dari keamanan algoritma tergatung pada kunci yang diapakai maka algoritma ini bisa dipublikasikan dan dianalisis oleh orang lain. Jika algoritma yang telah dipublikasikan bisa dipecahkan dalam waktu singkat oleh orang lain maka berarti algoritma tersebut tidaklah aman untuk digunakan. Berikut jenis-jenis algoritma kriptografi :

1) Algoritma Simetri : menggunakan satu kunci untuk proses enkripsi dan dekripsi.

2) Algoritma Asimetri : menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsi.

3) Hash Function.

a. Algoritma Simetri

Algoritma ini sering disebut algoritma klasik karena memakai kunci yang sama untuk proses enkripsi dan dekripsi. Algoritma ini sudah ada sejak 4000 tahun yang lalu. Bila mengirim pesan dengan menggunakan algoritma ini, pada sisi penerima harus diberitahukan kunci dari pesan tersebut agar bisa mendekripsikan pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan. Algoritma yang menggunakan kunci simetris diantaranya adalah :

1) Data Encryption Standart (DES).

2) RC2, RC4, RC5, RC6.

3) *International Data Encryption Algorithm (IDEA).*

4) Anvanced Encryption Standart (AES).

5) *One Time Pad (OTP).*

6) A5.

b. Algoritma Asimetri

Algoritma asimetri sering juga disebut algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda. Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu :

1) Kunci umum *(public key)* : kunci yang boleh semua orang tahu (dipublikasikan).
2) Kunci rahasia *(private key)* : kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang).

Kunci-kunci tersebut berhubungan satu sama lain. Dengan kunci publik orang dapat mengenkripsi pesan tetapi tidak bisa mendekripsinya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsi pesan tersebut. Algoritma asimetri bisa mengirimkan pesan dengan lebih aman daripada algoritma simetri. Berikut algoritma yang menggunakan kunci pubilk :

1) Digital Signature Algorithm (DSA).
2) RSA
3) Diffie-Hellman (DH).
4) Elliptic Curve Cryptography (ECC).
5) Kriptografi Quantum.

2.2.4. *Data Encryption Standart (DES)*

Standar enkripsi data *Data Encryption Standart (DES)* merupakan algoritma enkripsi yang paling banyak dipakai didunia, yang diadopsi oleh *NIST (National Institue of Standards and Technology)* sebagai standar pengolahan informasi Federal AS. Secara umum standar enkripsi data terbagi menjadi tiga

kelompok, yaitu pemrosesan kunci, enkripsi data 64 bit dan dekripsi data 64 bit yang mana satu kelompok saling berinteraksi satu sama lain. Pada akhir tahun 1960 IBM memulai riset proyek *Lucifer* yang dipimpin oleh *Horst Feistel* untuk kriptografi komputer. Proyek ini berakhir pada tahun 1971 dan *Lucifer* pertama kali dikenal blok kode pada pengoperasian blok 64 bit dan menggunakan ukuran kunci 128 bit. Setelah IBM mengembangkan sistem enkripsi yang dikomersialkan maka *Lucifer* disebut dengan *DES (Data Encryption Standart)*.

DES (Data Encryption Standart) termasuk sistem kriptografi simetri dan tergolong jenis blok kode. *DES* beroperasi pada ukuran blok 64 bit. *DES* mengenkripsikan 64 bit teks asli menjadi 64 bit teks kode menggunakan 56 bit kunci internal *(internal key)* atau upa kunci *(subkey)*. Kunci internal dibangkitkan dari kunci eksternal *(external key)* yang panjangnya 64 bit, skema global dari algoritma *DES* adalah sebagai berikut :

a) Blok teks-asli dipermutasi dengan matriks permutasi awal *(initial permutation* atau IP). Bisa ditulis x0 = IP (x) = *L0R0,* dimana *L0* terdiri dari 32 bit pertama dari x0 dan 32 bit terakhir dari *R0.*

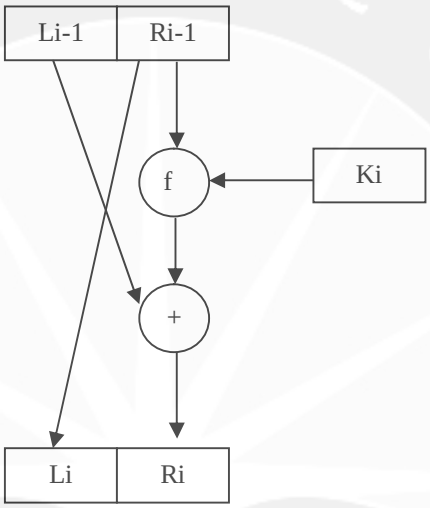
b) Hasil permutasi awal kemudian di-*enciphering* sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda dengan perhitungan *LiRi*

$1 \leq i \leq 16$, dengan mengikuti aturan berikut :

$Li = Ri - 1$

$Ri = Li - 1 \oplus f(Ri - 1, Ki)$

Dimana $\oplus$ merupakan *exclusive-or* dari dua. *f* adalah suatu fungsi dan $K_1,K_2,....,K_{16}$ dengan panjang 48 dari perhitungan fungsi dari kunci *K*. (Sebenarrnya K_1 adalah permutasi dari *K*). K_1, $K_2,....,K_{16}$ terdiri dari kunci skedul. Putaran pertama dari enkripsi tersebut ditunjukkan oleh gambar 2.1 berikut ini :

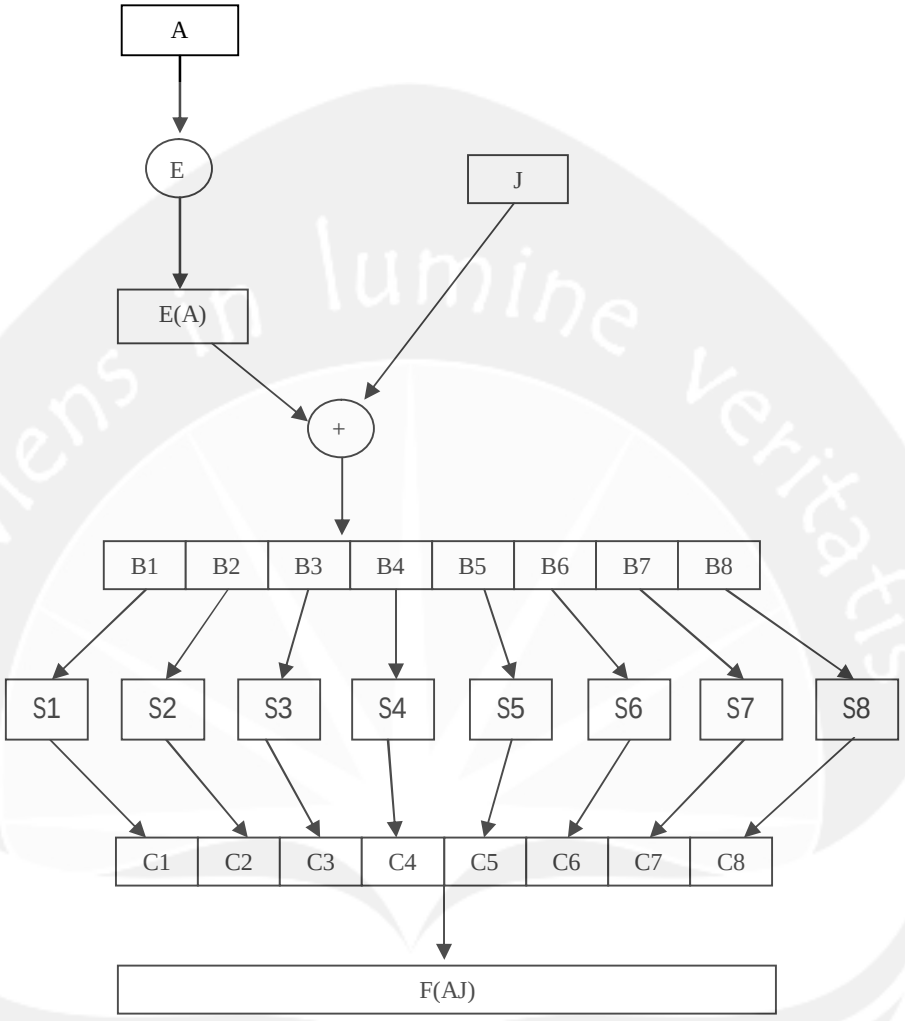


Gambar 2.1 Putaran pertama enkripsi DES (Dony Ariyus, 2008)

c) Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balik (*invers initial permutation* atau IP-1) menjadi blok teks-kode. IP^{-1} ke bitstring $R_{16}L_{16}$, memperoleh teks-kode *y*, kemudian $y = IP^{-1}(R_{16}L_{16})$.

Fungsi *f* diambil dari masukan pertama argumen A, dengan panjang *bitstring* 32, dan argumen yang kedua j dari panjang bitstring 48 dan prosedur keluaran dari *bitstring* adalah 32 seperti langkah berikut :

1) Argumen pertama A diperluas ke *bitstring* dengan panjang 48 menurut fix dari fungsi ekspansi E. E(A) yang berisi 32 bit dari A, permutasi dengan cara tertentu dengan 16 bit, sedikitnya muncul dua kali.



Gambar 2.2 Rincian DES Fungsi *f* (Dony Ariyus, 2008)

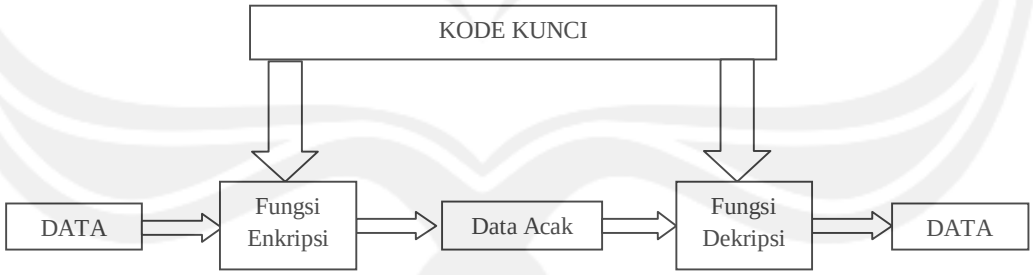
2) Perhitungan E(A) ⊕ *J* dan hasilnya ditulis pada penggabungan 6 bit string B =*B1B2B3B4B5B6B7B8.*

3) Langkah berikutnya menggunakan beberapa kotak-S Si,....,S8. Setiap Si adalah 4x16 larik yang dimasukkan dari integer 0-15. Dengan memberi suatu bitstring dengan panjang 6, seperti *Bj = b1 b2 b3 b4 b5 b6,* maka akan dapat dihitung *Sj(Bj)* dengan mengikuti dua bit *b1 b6,* tentunya degan representasi

binari row r of *Sj* ($0 \leq r \leq 3$), dan empat bit *b2 b3 b4 b5* dengan representasi binari i dari kolom *c of Sj* ($0 \leq c \leq 15$). Kemudian *Sj(Bj)* didefenisikan sebagai entry *Sj (r, c)* yang dituliskan ke dalam binari dengan panjang bitstring empat. Dengan begitu dapat dihitung $Cj=Sj(Bj)$, $1 \leq j \leq 8$.

4) *Bitstring C = C1 C2 C3 C4 C5 C6 C7 C8* dengan panjang 32 adalah permutasi dari hasil bitstring P(*C)* yang didefenisikan untuk *f(A,J)*.

Fungsi *f* dijelaskan pada gambar 2.2 Pada dasarnya berisi subsitusi dengan menggunakan S-box (kotak-s) dengan engikuti permutasi P. 16 iterasi dari *f* yang terdiri dari sistem kripto. Data dienkrip dalam blok-blok. 64 bit menggunakan kunci 56 bit. *DES* mentransformasikan masukan 64 bit dalam beberapa tahap enkripsi ke dalam keluaran 64 bit. Dengan demikian *DES* termasuk lama blok kode degan tahapan pemakaian kunci yang sama untuk dekripsinya.



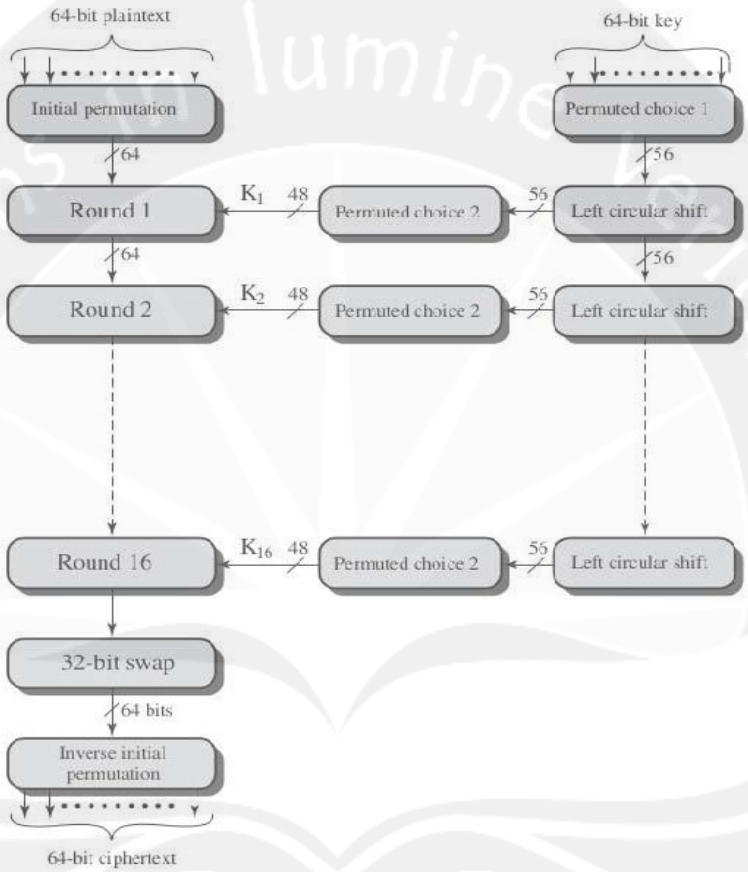
Gambar 2.3 Pemakaian Kunci pada *DES*

Secara umum skema *Data Encryption Standard (DES)* mempunyai dua fungsi masukan, yaitu :

a) Teks-asli untuk dienkripsi dengan panjag 64 bit.

b) Kunci dengan panjang 56 bit.

Skema dari pemrosesan *Data Encryption Standard (DES)* seperti gambar 2.4 berikut :



Gambar 2.4 Gambaran Umum Algoritma *DES*

Proses dari permutasi inisial (IP) teks asli ada tiga :

1) Teks asli 64 bit diproses di permutasi inisial (IP) dan menyusun kembali bit untuk menghasilkan permutasi masukan.

a) Langkah untuk melakukan perulangan kata dari teks asli sebanyak 16 kali dengan melakukan fungsi yang sama, yang menghasilkan fungsi permutasi

subsitusi, yang mana keluaran akhir dari hal tersebut berisi 64 bit (fungsi dari tek-asli dan kunci). Masuk ke swap dan menghasilkan *pre-output.*

b) *Pre-output* diproses dan permutasi diinvers dari permutasi inisial yang akan menghasilkan teks-kode 64 bit.

Proses dari kunci 56 bit :

1) Kunci melewati fungsi dari permutasi.
2) Pergeseran kunci, yang mana akan dipilih perulangan-perulangan permutasi kunci sebanyak 16 kali yang menghasilkan upa-kunci (Ki) yang diproses dengan kombinasi permutasi.
3) Perbedaan dari upa-kunci (Ki) akan dilakukan pergeseran kunci yang menghasilkan kombinasi teks-asli 64 bit dengan kunci 56 bit.

2.2.5. *Advanced Encryotion Standart (AES)*

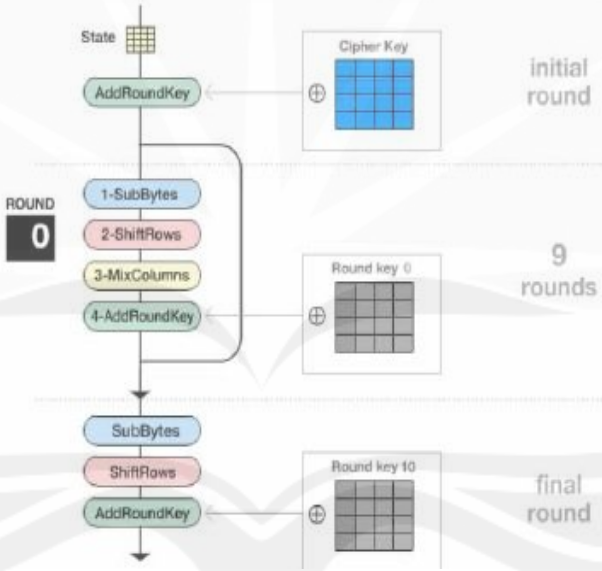
Advanced Encryption Standard (AES) merupakan algoritma *cryptographic* yang dapat digunakan untuk mengamankan data. Algoritma *AES* adalah blok chipertext simetrik yang dapat mengenkripsi *(encipher)* dan dekripsi *(decipher)* informasi. Enkripsi merubah data yang tidak dapat lagi dibaca disebut ciphertext;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

Kriteria pemilihan *AES* didasarkan pada 3 kriteria utama yaitu : keamanan, harga, dan karakteristik algoritma beserta implementasinya. Keamanan merupakan faktor terpenting dalam evaluasi (minimal seaman *3DES*), 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 *3DES*) 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*. Berikut tabel 2.2 jenis-jenis algoritma *AES :*

Tabel 2.2 Jenis-jenis Algoritma *AES*

	AES 128	**AES 192**	**AES 256**
Key size	4 word (16 byte)	6 word (24 byte)	8 word (32 byte)
Plaintext block size	4 word (16 byte)	4 word (16 byte)	4 word (16 byte)
Number of round	10	12	14
Round key size	4 word (16 byte)	4 word (16 byte)	4 word (16 byte)
Expanded key size	44 word (176 byte)	52 word (208 byte)	60 word (240 byte)

Advanced Encryption Standard (AES) mempunyai kunci 128, 192, 256 bit sehingga berbeda dengan panjang dari putaran *Rijndael*. Dari tabel tersebut *Advanced Encryption Standard (AES)* 128 bit menggunakan panjang kunci $N_k = 4$ *word* (kata) yang setiap katanya terdiri dari 32 bit sehingga total kunci 128 bit, ukuran blok teks-asli 128 bit dan memiliki 10 putaran. Sedangkan putaran untuk kunci terdiri dari $K_i = 4$ kata dan total putaran kunci 128 bit dan kunci yang diperluas mempunyai ukuran 44 kata dan 176 *byte*. Berikut adalah gambar 2.5 proses enkripsi algoritma *Advanced Encryption Standard (AES)*.



Gambar 2.5 Proses Enkripsi Algoritma *AES*

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*.

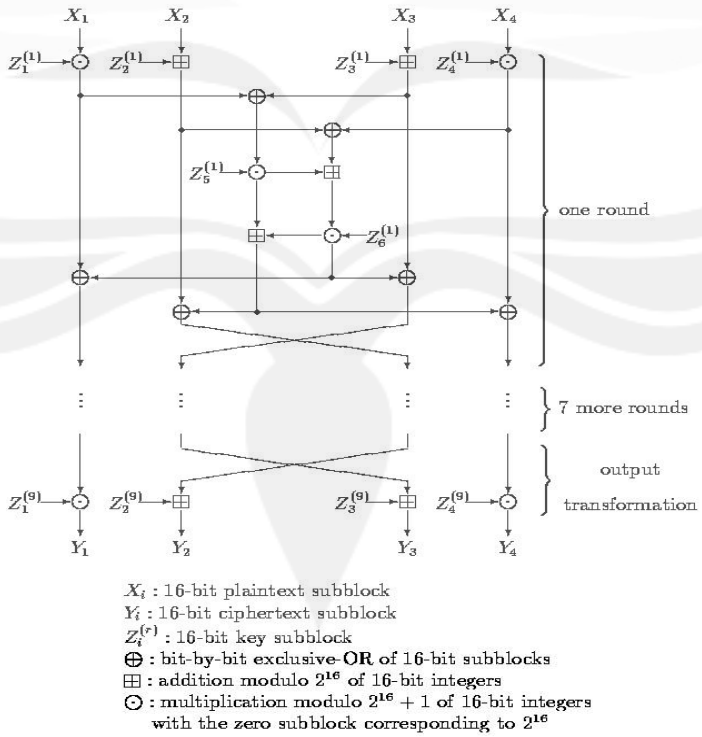
2.2.6. *International Data Encryption Standard (IDEA)*

Pada tahun 1990, *Xuejia Lai* dan *James Massey* dari Federal Institute Technology Swiss menemukan blok kode baru. Pada versi aslinya, blok algoritma *oriented encryption* disebut dengan *Propose Encryption Standard* (PES). PES masih lemah terhadap serang kriptografi difrensial. Pada tahun 1992, revisi PES menjadi *International Data Encryption Standard (IDEA)* yang merupakan blok kode dengan panjang blok teks-asli 64 bit dan panjang kunci 128 bit.

International Data Encryption Standart (IDEA) merupakan algoritma simetris yang beroperasi pada sebuah blok teks-asli yang panjangnya 64 bit dan kunci 128 bit. Algoritma *International Data Encryption Standart (IDEA)* menggunakan operasi campuran dari tiga operasi aljabar yang berbeda, yaitu operasi XOR ($\oplus$), operasi penjumlahan modulo 2^{16} ($\boxplus$) dan operasi perkalian modulo ($2^{16} + 1$) ($\odot$). Semua operasi ini dilakukan pada subblok 16 bit. Algoritma *International Data Encryption Standart (IDEA)* terdiri dari delapan putaran untuk mendapat keluaran transformasi akhir. Blok masukkan 64 bit dibagi menjadi 16 bit sub-blok yang diberi label X1, X2, X3 dan X4. Empat sub-blok yang akan menjadi masukkan putaran pertama dari algoritma *International Data Encryption Standart (IDEA)*, pembangkit upa-kunci mempunyai total 52 blok

upa-kunci yang akan dibangkitkan dari kunci asli enkripsi 128 bit. Setiap blok terdiri dari 16 bit.

Pada putaran pertama yang menggunakan enam upa-kunci 16 bit (Z1, Z2,...Z6), dan pada putaran yang terakhir menggunakan empat blok upa-kunci 16 bit, dengan hal tersebut dihasilkan teks-kode 64 bit. Empat kunci subblok 16 bit melakukan operasi XOR. Tambahan dan perkalian dari satu ke lainnya dan dengan 6 subblok kunci 16 bit. Diantara putaran ada yang ditukar tempat antara subblok 2 dan 4. Pertambahan dan pertukaran proses bit yang membuat algoritma *International Data Encryption Standart (IDEA)* sulit untuk dipecahkan oleh kriptanalis diferensial. Berikut gambar 2.6 proses enkripsi algoritma *International Data Encryption Standart (IDEA).*



Gambar 2.6 Proses Enkripsi Algoritma *IDEA*

2.2.7. *Android*

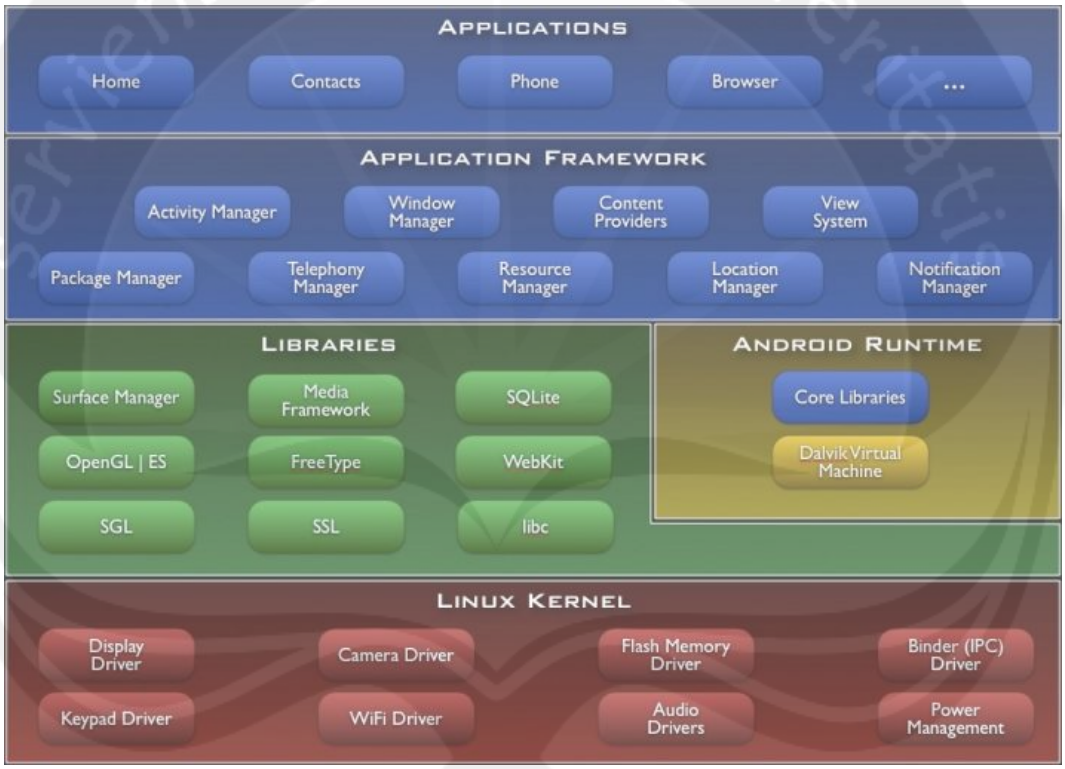
Pada Juli 2000, *Google* bekerjasama dengan *Android Inc*., perusahaan yang berada di Palo Alto, California Amerika Serikat. Para pendiri *Android Inc*. bekerja pada *Google*, di antaranya Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Saat itu banyak yang menganggap fungsi *Android Inc*. hanyalah sebagai perangkat lunak pada telepon seluler. Sejak saat itu muncul rumor bahwa *Google* hendak memasuki pasar telepon seluler. Di perusahaan *Google*, tim yang dipimpin Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh kernel *Linux*. *Android* adalah sistem operasi untuk telepon seluler-ponsel *(handphone)* yang berbasis *Linux. Android* menyediakan *platform* terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, *Google Inc.* membeli *Android Inc.*, pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google*, *HTC*, *Intel*, *Motorola*, *Qualcomm*, *T-Mobile*, dan *Nvidia*. Pada tanggal 12 november 2007 *google* besama *Open Handset Alliance (OHA)* yaitu konsorium perangkat *mobile* terbuka, merilis *Google Android SDK*. Berikut gambar 2.7 logo *Android*.

Gambar 2.7 Logo *Android*

Google bersama OHA merilis paket *software* SDK yang lengkap untuk mengembangkan aplikasi pada perangkat *mobile*. Yaitu, sistem operasi *Middleware* dan aplikasi utama untuk perangkat *mobile*. Sebagai *programer* atau *developer* kita bisa melakukan segalanya, mulai dari membuat aplikasi SMS hanya dengan dua baris kode hingga mengganti *event* pada *home screen* perangkat *android*. Selain itu, bahkan dengan mudah kita bisa membuat dan mengkustomisasi sistem operasinya, atau mengganti semua aplikasi *default google*. Semua aplikasi yang dibuat untuk *android* akan memiliki akses yang setara dalam mengakses seluruh kemampuan *handset*, tanpa membedakan apakah itu merupakann aplikasi inti atau aplikasi pihak ketiga. Dengan kata lain *paltform android* ini, *programer* atau *developer* secara penuh akan bisa mengkustomisasi perangkat *androidnya*.

Android builtin pada *linux kernel (Open Linux Kernel)*, dengan sebuah mesin *virtual* yang didesain dan untuk mengoptimalkan penggunaan sumber daya memori dan *hardware* pada lingkungan perangkat *mobile (mobile environment)*. *Davlik* adalah nama dari *android virtual* mesin yang akan mengeksekusi *file* kedalam format *davlik executable* (**.dex*). sebuah format yang telah dirancang untuk ruang penyimpanan yang efisien dan eksekusi memori yang terpetakan *(memory-mappable execution)*. *Davlik virtual* mesin *(davlik VM)* berbasis *register (register-based)*, dan dapat mengeksekusi kelas *(class)* yang telah terkompilasi pada *compiler bahasa java*, kemudian di transformasikan ke dalam *native* format dengan menggunakan *tool "dx"* yang telah terintegrasi.

Sistem Operasi *Google Android* di Ponsel memang terbilang masih baru, tetapi Sistem Operasi *Android* telah mengalami perkembangan yang cukup pesat. Diciptakan sebagai tandingan iOS, *Android* menunjukan grafik perkembangan yang signifikan, tentunya hal itu juga tidak lepas dari dukungan para pabrikan ponsel besar yang juga ikut andil menghadirkan ponsel-ponsel bersistem operasi *Android*. Berikut gambar 2.8 arsitektur *android* :



Gambar 2.8 Arsitektur *Android*

a) *Linux Kernel*

Android dibangun di atas kernel *Linux* 2.6. Namun secara keseluruhan *android* bukanlah *linux,* karena dalam *android* tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal

dalam manajemen memori dan proses. Oleh karenanya pada *android* hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan *driver*. *Kernel linux* menyediakan *driver* layar, kamera, *keypad*, *WiFi*, *Flash Memmory*, *audio*, dan IPC *(Interprocess Communication)* untuk mengatur aplikasi dan lubang keamanan.

b) *Libraries*

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution (BSD)* hanya setengah dari yang aslinya untuk tertanam pada *kernel Linux*. Beberapa pustaka diantaranya:

1) Media *Library* untuk memutar dan merekam berbagai macam format *audio* dan *video*.
2) *Surface* Manager untuk mengatur hak akses layer dari berbagai aplikasi.
3) *Graphic Library* termasuk didalamnya SGL dan OpenGL, untuk tampilan 2D dan 3D.
4) *SQLite* untuk mengatur relasi database yang digunakan pada aplikasi.
5) SSl dan WebKit untuk *browser* dan keamanan internet.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi *Android* 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

c). *Android Runtime*

Pada *android* tertanam paket pustaka inti yang menyediakan sebagian besar fungsi *android*. Inilah yang membedakan *Android* dibandingkan dengan sistem operasi lain yang juga mengimplementasikan *Linux*. *Android Runtime* merupakan mesin *virtual* yang membuat aplikasi *android* menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam *Android Runtime* terdapat 2 bagian utama, diantaranya:

1) Pustaka Inti, *android* dikembangkan melalui bahasa pemrograman Java, tapi *Android runtime* bukanlah mesin virtual Java. Pustaka inti *android* menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus *android*.

2) Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format Dalvik Executable (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti threading dan manajemen memori yang terbatas (Nicolas Gramlich, Andbook, anddev.org).

d). *Application Framework*

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi *android*. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan user interface dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi *android* adalah sebagai berikut :

1) *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”Backstack“ untuk navigasi penggunaan.

2) *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.

3) *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan *file layout*.

4) *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat *android* berada.

5) *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada status bar.

e). *Application Layer*

Puncak dari diagram arsitektur *android* adalah lapisan aplikasi dan widget. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika

menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam *Android runtime* dengan menggunakan kelas dan *service* yang tersedia pada *framework* aplikasi. Lapisan aplikasi *android* sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada *android* semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka *API (Application Programming Interface)* yang sama.

f). Komponen Aplikasi

Fitur penting *android* adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada. Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek *java* untuk bagian itu. Oleh karenanya *android* berbeda dari sistem-sistem lain, *Android* tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

1) *Activities*

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *activity* dideklarasikan dalam sebuah kelas yang bertugas untuk menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity* atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan. Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack*. Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan *acitivity, LIFO (Last In First Out)* dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol Back, atau menutup activity yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. Memory Manager *android* menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan activity, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut.

2) *Services*

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan

untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *activity* tidak aktif atau tidak tampak.

3) *Intents*

Intents merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman web, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misal pesan masuk. *Intents* dapat eksplisit atau implisit, contohnya jika suatu aplikasi ingin menampilkan URL, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

4) *Broadcast Receivers*

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *broadcast* berasal dari sistem misalnya, *battery* sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa *default* pada perangkat. Sama halnya dengan *service*, *Broadcast receivers* tidak menampilkan antarmuka pengguna. Namun, Broadcast Receivers dapat menggunakan *notification manager* untuk memberitahukan sesuatu kepada pengguna.

5) *Content Providers*

Content providers digunakan untuk mengelola dan berbagi *database*. Data dapat disimpan dalam *file sistem*, dalam *database SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *content provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya.