BAB 2
TINJAUAN PUSTAKA

2.1 Tinjauan Penelitian Terdahulu

Dari topik yang dipilih, terdapat kumpulan penelitian terdahulu yang diangkat sebagai pembanding serta acuan sehingga bisa mengetahui hasil dan kesimpulan maupun menghindari adanya kesamaan dengan penelitian terdahulu.

Tabel 2.1 Ringkasan Penelitian Pembanding

No.	Nama Penulis	Judul dan Tahun	Metode	Hasil	Perbedaan
1	Aditya Hermawan , Erik Iman Heri Ujianto	Implementasi Enkripsi Data Menggunakan Kombinasi AES dan RSA (2021)	Algoritma AES dan Algoritma RSA	Penggunaan ideal pada pembangkitan kunci yaitu 2048 bit. Dan Proses dekripsi lebih lama dibanding enkripsi.	*Platform* sistem, Panjang kunci algoritma AES
2	Hendra Pasaribu, Delima Sitanggang , Rudolfo Rizki Damanik, Alex Chandra Rudianto Sitompul	*Combination of advanced encryption standard 256 bits with md5 to secure documents on android smartphone* (2018)	Algoritma AES 256 bit dan MD5 hashing	Waktu proses enkripsi dan dekripsi lebih cepat dibanding hanya menggunakan algoritma AES saja.	Kombinasi algoritma, Telepon pintar untuk uji coba data
3	Rizki Nanda Nasution, Budi Triandi	Implementasi Metode RSA dan AES untuk Mengamankan *File WinRAR* dan ZIP (2021)	Algoritma RSA dan Algoritma AES	Aplikasi pengamanan *file Winrar* & *ZIP* menggunakan kombinasi algoritma RSA dan AES berbasis	*Platform* aplikasi, Pengujian data

				Desktop.	
4	Rinmar Siringoringo	Analisis dan Implementasi Algoritma Rijndael (AES) dan Kriptografi RSA pada Pengamanan File (2020)	Algoritma Rijndael (AES) dan Algoritma RSA	Sistem operasi enkripsi dan operasi dekripsi menggunakan kombinasi Algoritma AES dan RSA berbasis *Windows*	*Platform* aplikasi, Panjang kunci algoritma AES
5	Abhishek Guru, Asha Ambhaikar	*AES and RSA-Based Hybrid Algorithms For Message Encryption & Decryption* (2021)	Algoritma AES dan Algoritma RSA	*Hybrid algorithm* dari algoritma AES dan RSA tidak hanya untuk enkripsi tapi terdapat manfaat efisiensi dan pengamanan lebih.	*Platform* aplikasi, Jenis dokumen yang digunakan untuk pengujian
6	Suci Ramadani, Diana, Siti Sauda	Penerapan Algoritma AES dan DSA Menggunakan Hybrid Cryptosystem untuk Keamanan Data (2020)	Algoritma AES dan Algoritma DSA	Aplikasi enkripsi dekripsi file dengan tingkat keberhasilan enkripsi sebesar 86% dan 83% untuk dekripsi. Dan ukuran *file* akan naik setelah enkripsi dan kembali semula ketika dekripsi.	*Platform* aplikasi, Kombinasi algoritma
7	Noveline Aziz Fauziah, Eko Hari Rachmawanto, De Rosal Ignatius Moses Setiadi	*Design and Implementation of AES and SHA-256 Cryptography for Securing Multimedia File over Android Chat Application* (2018)	Algoritma AES dan SHA-256 *hashing*	Enkripsi dan dekripsi berkas multimedia berhasil dilakukan, serta ukuran berkas akan bertambah dari ukuran awal.	Kombinasi algoritma, Jenis dokumen yang digunakan untuk pengujian

8	Santi Pattanavichai	*Program for Simulation and Testing of Apply Cryptography of Advance Encryption Standard (AES) Algorithm with Rivest-Shamir-Adleman (RSA) Algorithm for Good Performance* (2022)	Algoritma AES dan Algoritma RSA	Waktu yang dibutuhkan ketika enkripsi dan dekripsi dari algoritma AES & RSA tidak jauh beda dengan algoritma AES. Dan proses efisiensi untuk *encoding* dan *decoding* bertambah baik.	*Platform* aplikasi, Jenis dokumen yang yang digunakan untuk pengujian
9	Mohamad Ali Sadikin, Rini Wisnu Wardhani	*Implementation of RSA 2048-Bit and AES 256-Bit with Digital Signature for Secure Electronic Health Record Application* (2016)	Algoritma RSA 2048 bit dan Algoritma AES 256 bit	Hasil dari pengujian *black box* menunjukan semua *output* sesuai dengan ekspektasi. Dan *white box testing* mendapatkan *error density* sebesar 11.67, 7.54 dan 8.29 untuk projek dengan kuranng dari 16,000 baris kode	*Platform* aplikasi, Jenis dokumen yang digunakan untuk pengujian
10	Samir G. Chaloop, Mahmood Z. Abdullah	*Enhancing Hybrid Security Approach Using AES and RSA Algorithms* (2021)	Algoritma AES dan Algoritma RSA	Algoritma digunakan untuk pengamanan data di *cloud application*. Hasil eksperimen membandingkan waktu proses dan didapatkan bahwa *hybrid algorithm* lebih baik dari segi pengamanan.	*Platform* aplikasi, Telepon pintar untuk uji coba data

Berdasarkan tabel 2.1, dari perbandingan yang dilakukan terdapat perbedaan dan persamaan. Persamaanya adalah menggunakan teknik kriptografi enkripsi dan dekripsi dari salah satu algoritma simetris atau asimetris. Perbedaan penelitian ini dari jurnal pembanding yang digunakan cukup beragam, terdiri dari perbedaan *platform* dari aplikasi yang dibuat, penerapan kombinasi algoritma, jenis data atau dokumen yang ditetapkan untuk pengujian, panjang kunci AES hingga perangkat telepon pintar untuk uji coba data. Kontribusi yang diberikan oleh penelitian terdahulu pada penelitian ini antara lain penggunaan algoritma yang sama, contoh data yang digunakan sebagai pedoman peneliti dalam membuat hipotesis.

2.2 Tinjauan Teoritis

2.2.1 Kriptografi

Keilmuan tentang teknik untuk mengamankan informasi digital pada teknologi yang berbeda mulai dari transaksi keuangan hingga komputasi terdistribusi didefinisikan sebagai kriptografi (Katz & Lindell, 2020). Kata kriptografi dalam bahasa Yunani secara etimologi diartikan sebagai *kryptos* (tersembunyi) atau menyembunyikan dan *graphein* (menulis). Menurut peristilahan, kriptografi adalah ilmu menjaga kerahasiaan sebuah informasi yang dipelajari, agar membuat tetap aman informasi yang disampaikan ke penerima.

Penggunaan kriptografi sebelum tahun 1980-an secara histori banyak digunakan pada organisasi militer dan badan rahasia (Katz & Lindell, 2020). Namun saat ini, kriptografi ada di mana-mana. Mekanisme keamanan yang mengandalkan kriptografi merupakan bagian integral dari hampir semua sistem komputer. Kriptografi digunakan untuk mengatur akses kontrol dalam sistem operasi multi-pengguna, untuk mencegah pencuri mengekstraksi informasi rahasia dari laptop curian dan lainnya.

Singkatnya, kriptografi telah berubah dari bentuk seni yang berurusan dengan keamanan komunikasi rahasia untuk militer menjadi ilmu yang membantu mengamankan sistem untuk masyarakat umum di seluruh dunia. Ini juga berarti bahwa kriptografi menjadi topik yang semakin sentral dalam ilmu komputer di mana banyak perangkat lunak yang sudah mengimplementasikan enkripsi, autentikasi, dan lainnya untuk pengamanan informasi.

Gambar 2.1 Proses Enkripsi dan Dekripsi

Kriptografi memiliki beberapa terminologi yang sering dikenakan pada prosesnya, antara lain:

a. *Plaintext* adalah data asli yang masih dapat dibaca dan masih utuh.
b. *Ciphertext*; Hasil proses operasi enkripsi menggunakan algoritma kriptografi.
c. Kunci sebagai suatu nilai yang disembunyikan untuk dipergunakan pada operasi enkripsi maupun operasi dekripsi.
d. Enkripsi adalah tahapan penyandian data asli (*plaintext*) ke data acak (c*iphertext*).
e. Dekripsi adalah kontradikisi enkripsi dengan proses pengubahan data acak (*ciphertext*) ke data asli (*plaintext*).

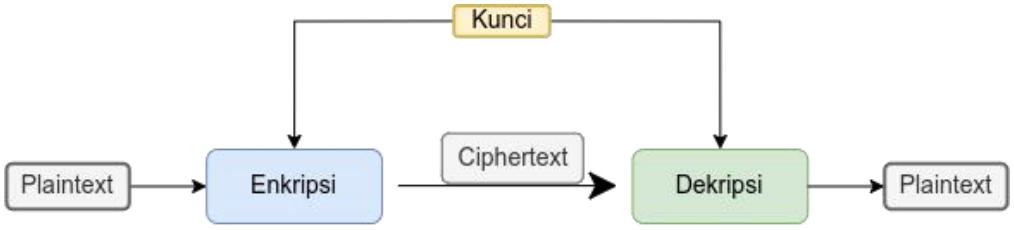
Secara umum ada dua tipe algoritma kriptografi berdasarkan kuncinya yaitu algoritma simetris dan algoritma asimetris. Algoritma simetris adalah algoritma yang memiliki kunci enkripsi dan dekripsi yang sama, sedangkan algoritma asimetris terdiri atas dua buah kunci yaitu kunci publik untuk melakukan enkripsi dan kunci privat untuk melakukan dekripsi.

Penggunaan algoritma kriptografi yang baik tidak hanya ditentukan oleh kerumitan dalam mengamankan informasi, tetapi juga harus memenuhi 4 prinsip dasar keamanan sebagai berikut (Munir, 2019):

a. Kerahasiaan. Hanya pihak tertentu saja yang dapat membaca pesan.
b. Autentikasi. Pengirim harus terpercaya, teridentifikasi dan terdata dengan pasti
c. Integritas. Pesan harus masih dalam keadaan utuh ketika sampai ke penerima tanpa adanya perubahan informasi sedikit pun.
d. *Non-Repudiation*. Penolakan pesan tidak bisa dilakukan oleh pengirim yang sudah dikirimkan.

2.2.2 Algoritma Simetris

Semua kriptografi dari zaman dulu hingga 1976 secara eksklusif didasarkan pada algoritma simetris yang melakukan proses enkripsi untuk mengacak informasi dan dilanjutkan pengembalian informasi awal dengan istilah dekripsi di mana memiliki kunci rahasia yang sama (Paar & Pelzl, 2010) . algoritma kunci simetris memiliki berbagai tipe algoritma mulai dari DES (*Data Encryption Standard*), kemudian *blowfish*, *twofish*, dilanjutkan MARS, IDEA, 3DES (DES diaplikasikan 3 kali), dan AES (*Advanced Encryption Standard*) yang banyak dipakai pada kemanan data saat ini.



Gambar 2.2 Algoritma Simetris

Pada gambar 2.2 menjelaskan sebuah *plaintext* dienkripsi menggunakan kunci sehingga menjadi *ciphertext*, dan setelah itu bentuk *ciphertext* didekripsi pada kunci yang sama ketika melakukan enkripsi sehingga menjadi *plaintext* kembali.

Kelebihan algoritma simetris adalah (Stallings, 2017):

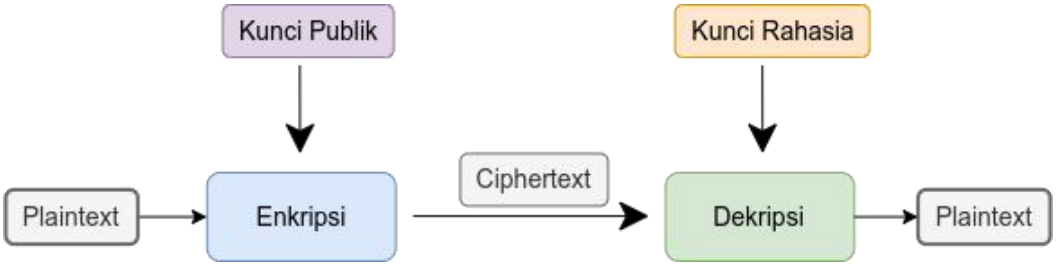
- Proses enkripsi ataupun dekripsi membutuhkan waktu yang sangat singkat.
- Cipher yang dihasilkan lebih kuat karena susunan yang dilakukan
- Ukuran kunci simetris pendek.

Kekurangan algoritma simetris adalah (Stallings, 2017):

- Pertukaran kunci simetris membutuhkan jaminan keamanan yang tinggi, karena pengirim dan penerima wajib memperhatikan kerahasiaan kunci.
- Kunci perlu sering diganti, ketika melakukan pertukaran.

2.2.3 Algoritma Asimetris

Suatu algoritma kriptografi yang memiliki dua kunci yang berbeda ketika melakukan proses operasi enkripsi dan dekripsi disebut algoritma asimetris. Operasi enkripsi memakai kunci publik yang diketahui dan diperlihatkan oleh siapapun, sedangkan dekripsi hanya penerima pesan yang mengetahui kunci yang digunakan yaitu kunci rahasia. Salah dua algoritma asimetris terdiri dari *Elliptic Curve Cryptography* (ECC) dan Rivest Shamir Adleman (RSA).



Gambar 2.3 Algoritma Asimetris

Pada gambar 2.3 menjelaskan sebuah *plaintext* dienkripsi menggunakan kunci publik (*public key*) sehingga menjadi *ciphertext*, dan setelah itu bentuk *ciphertext* didekripsi menggunakan kunci rahasia (*private key*). Operasi antara enkripsi dan dekripsi memiliki perbedaan kunci yang digunakan, artinya pengubahan *plaintext* ke *ciphertext* dan sebaliknya harus memerlukan dua kunci yang berbeda.

Kelebihan algoritma asimetris adalah (Stallings, 2017):

- Pertukaran kunci lebih terjamin untuk masalah keamanannya, yang perlu diawasi yaitu kunci rahasia.
- Pengubahan kunci publik dan rahasia bisa digunakan dalam periode yang lama.
- Dapat diaplikasikan dalam mengamankan pertukaran kunci simetris.
- Beberapa algoritma kunci publik dapat digunakan untuk memberi tanda tangan digital pada pesan.

Kekurangan algoritma asimetris adalah (Stallings, 2017):

- Proses lebih lama dibandingkan algoritma simetris ketika melakukan enkripsi dan dekripsi, karena memerlukan perhitungan dengan nilai besar.
- Dengan memiliki dua kunci, ukuran bertambah menjadi lebih besar dari kunci simetris.
- Hasil *ciphertext* lebih besar daripada ukuran *plaintext*-nya.

Dari kelebihan dan kekurangan yang sudah dijabarkan dari setiap algoritma kriptografi simetris maupun asimetris, penerapan kombinasi bisa dilakukan dengan menerapkan prinsip *hybrid cryptography* yang menggabungkan antara kriptografi simetris dan asimetris. Penggabungan ini akan meningkatkan keamanan serta efisiensi operasi yang cepat.

2.2.4 Algoritma *Advanced Encryption Standard (AES)*

A. Sejarah *Advanced Encryption Standard (AES)*

Tahun 1997 awal bulan Januari, NIST (*National Institute of Standard and Technology)* atau disebut sebagai lembaga standar teknologi Amerika Serikat mengumumkan diadakannya kontes pemilihan algoritma untuk berpindah dari DES (*Data Encryption Standard*). Pada tahun 1999, NIST mengumumkan lima algoritma dari hasil seleksi yang ketat serta beberapa tahapan pengujian yang sudah dilakukan.

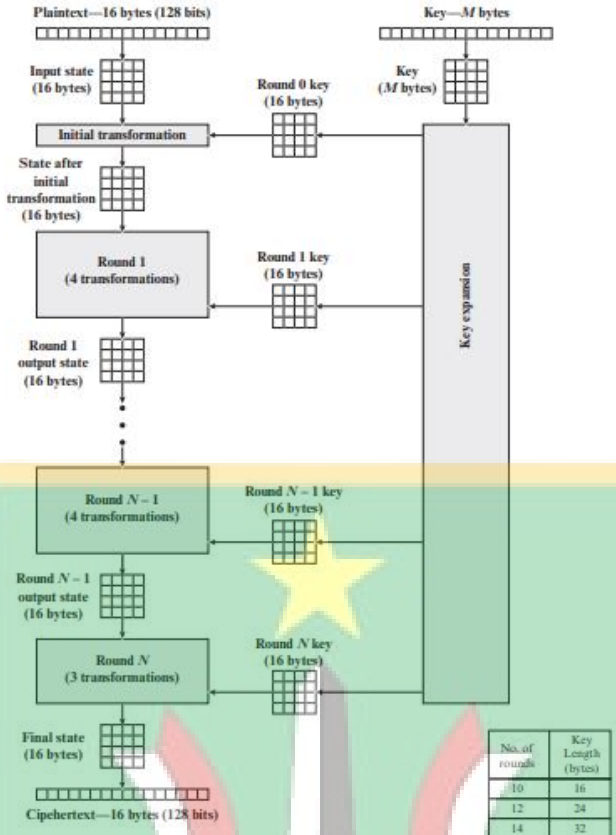
Algoritma tersebut terdiri dari MARS dari IBM negara Amerika Serikat, RC6 dari RSA *Laboratories* negara Amerika Serikat, *Rijndael*/AES dari negara Belgia diwakili oleh

kriptografer bernama Joan Daemen dan Vincent Rijmen, *Serpent* dari gabungan tiga negara Inggris diwakili oleh Ross Anderson, Israel oleh Eli Biham, serta Norwegia oleh Lars Knudsen, *Twofish* dari negara Amerika Serikat yang diwakili oleh Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall dan Niels Ferguson.

Akhirnya melalui pemilihan yang selektif dan kualifikasi yang sulit dari NIST, Rijndael/AES terpilih sebagai pemenang pada 2 Oktober 2000, dibuat oleh Dr. Vincent Rijmen dan Dr. Joan Daemen. Algoritma ini dipilih karena kinerjanya yang baik secara konsisten atas penerapannya di perangkat lunak, perangkat keras, firmware, dan kartu pintar. Kemudian algoritma AES digunakan untuk menggantikan DES pada algoritma simetris *block cipher* (Paar & Pelzl, 2010).

B. Proses Enkripsi *Advanced Encryption Standard (AES)*

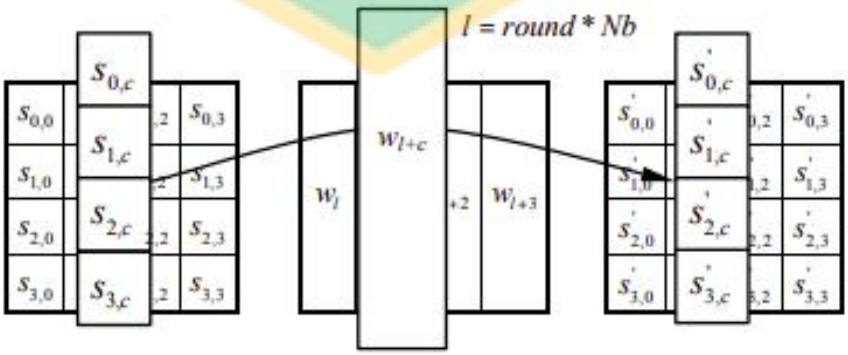
128 bit merupakan jumlah dari kedua blok seperti input dan output serta state dari algoritma AES. Dengan ukuran kunci 128 bit yang dipakai di algoritma AES tidak harus memiliki blok input yang besar. Penerapan algoritma AES pada *Cipher key* (K) bisa mengaplikasikan kunci dengan panjang 128 bit, 192 bit, atau 256 bit (Paar & Pelzl, 2010; Stallings, 2017). Algoritma AES-128 memiliki 10 putaran yang paling sedikit dibanding dua lainnya, AES-192 memiliki 12 putaran dan AES-256 memiliki 14 putaran yang paling lama.



Gambar 2.4 Proses Enkripsi Algoritma AES (Stallings, 2017)

Operasi enkripsi algoritma AES dilakukan dengan sejumlah tahapan sebagai berikut:

- *AddRoundKey*, disebut juga sebagai *initial round* yang melakukan operasi *exclusive or (XOR)* antara *state* awal dengan *cipher key*. Transformasi *AddRoundKey* diimplementasikan pertama kali pada *round = 0*, dimana kunci yang digunakan adalah *cipher key*.



Gambar 2.5 Proses *AddRoundKey*

- *Round,* Putaran yang dilakukan sebanyak N - 1 kali. Pada setiap prosesnya akan dilakukan transformasi seperti:
    1. *SubBytes*: substitusi *byte* non-linear memakai table substitusi (S-box) secara independent.
    2. *ShiftRows*: pertukaran tiga baris terakhir dari *state* pada jumlah *byte* yang berbeda.
    3. *MixColumns*: mengubah *state value* di setiap kolom secara acak.
    4. *AddRoundKey*: menambahkan *state* sekarang dengan operasi *exclusive or* (XOR) sederhana.
- Putaran akhir hanya dikurangi operasi *MixColumns* dari langkah-langkah yang dilakukan pada tahapan *Round* sebanyak N – 1.

2.2.5 Algoritma *Rivest Shamir Adleman (RSA)*

Tiga kriptografer yang bernama Ron Rivest dari Amerika Serikat, Adi Shamir dari Israel, dan Leonard Adleman dari Amerika Serikat di Massachusetts Institute of Technology mengajukan algoritma yang bernama Rivest-Shamir-Adleman (RSA) dengan konsep *private-public key* pada tahun 1977. Algoritma tersebut lalu dipatenkan pada tahun 1983 oleh Massachusetts Institute of Technology sampai 21 September 2000.

Algoritma RSA sebagai algoritma asimetris (algoritma *public key*) yang diskemakan pada saat operasi enkripsi memakai kunci publik dan dekripsi menggunakan kunci rahasia. Algoritma ini dianggap aman karena menyelesaikan pemfaktoran bilangan yang tidak kecil sehingga dari hasil perhitungan tersebut dianggap aman.

Keamanan algoritma RSA memakai pemfaktoran bilangan non-prima menjadi prima sebagai tingkat kesulitan yang tinggi untuk dipecahkan, yang dalam hal ini $r = p\,x\,q$. Penemu menganjurkan agar nilai p dan q memiliki panjang lebih dari 100 digit, agar hasil perkalian akan lebih dari 200 digit. Ukuran kunci sandi juga menjadi salah satu keamanan pada algoritma RSA, karena semakin besar ukuran kunci, maka makin besar pula peluang kombinasi kunci yang harus ditembus (Paar & Pelzl, 2010).

A. Pembangkit Kunci RSA

Sebelum penerapan algoritma RSA, dibutuhkan sepasang kunci untuk dipakai ketika operasi penyandian dan dekode. Sepasang kunci didapatkan dari membangkitkan kunci adalah menentukan dua bilangan prima besar. Dalam pembangkitan prima menggunakan algoritma pengujian, misalnya algoritma Miller-Rabin. Besaran bilangan prima ini

menentukan keaman dari sistem kriptografi RSA, semakin besar semakin sulit untuk difaktorisasi. Fungsi dari pembangkit kunci adalah menerima masukan panjang bit *n* dan mengembalikan sebuah struktur *linked list* yang berisi dua objek yaitu kunci publik dan kunci rahasia.

B. Konsep Dasar Enkripsi RSA

Ketika sebuah pesan (*plaintext*) akan dikirimkan melalui media komunikasi yang tidak aman, maka perlu melakukan proses enkripsi untuk menjaga kerahasiaan dari pesan tersebut. Untuk melakukan proses enkripsi RSA, blok $x1, x2, \ldots,$ dari pesan disusun semacam blok sehingga setiap blok bisa menggambarkan nilai di dalam rentang 0 sampai $r - 1$. Fungsi enkripsi memiliki masukan pesan sebagai sebuah *array byte p* dan sebuah kunci publik. Elemen *n* dan *e* merupakan hasil dari kunci publik.

C. Konsep Dasar Dekripsi RSA

Setelah pesan tersandi (*ciphertext*) diterima oleh penerima, maka dilakukan proses dekripsi untuk mengembalikan pesan tersandi (*ciphertext*) menjadi pesan semula (*plaintext*). Proses dekripsi hanya bisa dilakukan dengan menggunakan kunci privat. Fungsi dekripsi memiliki masukan teks sandi sebagai sebuah *array byte c* dan sebuah kunci privat. Terdapat tiga elemen yaitu *d, p* dan *q* dari kunci privat.

2.2.6 Android

Android muncul sekitar tahun 2000-an ketika sebuah perusahaan bernama Android Inc. didirikan oleh Andy Rubin. Saat itu, Google sudah mendukung Android Inc. tetapi belum memilikinya. Google mengakuisisi Android Inc. sekitar musim 2005 bulan Agustus lanjut pada tahun 2007, OHA (*Open Handset Alliance*) muncul, dan sistem operasi Android secara resmi menjadi *open source*. Tahun 2011 merupakan pencapaian penting karena sampai saat itu, sistem operasi Android masih terbatas pada ponsel. *Honeycomb*, penerus *Gingerbread*, adalah versi Android pertama yang dipasang di tablet (Hagos, 2018).

Sistem operasi adalah suatu hal yang kompleks, tapi untuk sebagian besar ini berdiri antara sebuah pengguna (*user*) dan perangkat keras (*hardware*). Pengguna dalam maksud ini bukan seseorang, melainkan istilah untuk mempermudah penjelasan. Sebagai contoh aplikasi *e-mail*, ketika mengetikan setiap karakter aplikasi membutuhkan komunikasi ke perangkat keras untuk membuat pesan tampil pada layar dan penyimpanan *hard drive* dan juga mengirim ke *cloud* melewati jaringan perangkat.

2.2.7 Kotlin

Pada tahun 2011, JetBrains meluncurkan Kotlin kemudian tahun berikutnya, mereka membuka Kotlin sebagai *open source project* di bawah lisensi Apache 2. Di Google I/O 2017, Google mengumumkan Kotlin menjadi bahasa pemrograman pertama untuk di platform Android. Kotlin merupakan sebuah bahasa pemrograman baru yang ditargetkan untuk *java platform*; program ini akan jalan pada JVM (*Java Virtual Machine*). Nama Kotlin diambil dari nama sebuah pulau yang dekat dengan St. Petersburg, tempat ini juga merupakan dimana sebagian besar tim Kotlin berada. Kotlin memiliki banyak karakteristik dan kapabilitas sebagai bahasa pemrograman, yaitu *Object Oriented Programming* (OOP) sama seperti Java, statis dan kuat dan lebih sedikit kode yang ditulis dibanding Java (Hagos, 2018).

2.2.8 Android Studio

Pada 2013 *Android Studio* dirilis, masih dalam versi beta, tetapi tujuan sudah jelas. Ini akan menjadi IDE pengembangan Android yang sempurna dan resmi. *Android Studio* didasarkan pada *IntelliJ JetBrains*. Selain editor kode, fitur pada *Android Studio* bisa meningkatkan produktivitas saat mengembangkan aplikasi. Selain itu *Android Studio* telah terintegrasi dengan banyaknya *plugins* yang tersedia dan bahasa pemrograman Kotlin. Terdapat beberapa bahasa JVM, tetapi Java selalu dijadikan prioritas bahasa pemrograman utama untuk mengembangkan aplikasi Android, sampai pada tahun 2017, ketika acara Google I/O diumumkan bahwa Android akan menggunakan prioritas utamanya menggunakan bahasa pemrograman Kotlin dan *Android Studio* 3 akan otomatis memiliki bantuan untuk Kotlin (Hagos, 2018).

2.2.9 Android SDK (Software Development Kit)

Android Software Development Kit (SDK) merupakan kumpulan beberapa alat untuk pengembangan bagi para developer. Semua proses mulai dari alat untuk *debugging*, emulator untuk menguji aplikasi, contoh kode dan tutorial, serta *software library. Android SDK* terus dikembangkan dengan bersamaan *platform* Android. SDK juga bisa mendukung versi Android yang lebih lama jika *developer* ingin menjalankan aplikasi ke perangkat lama. (Hagos, 2018).