

## BAB II

### TINJAUAN PUSTAKA

#### 2.1. Penelitian Terkait

Penelitian terkait yang sudah dilakukan yaitu Analisis Kombinasi Metode *Caesar Cipher*, *Vernam cipher*, Dan *Hill Cipher* Dalam Proses Kriptografi oleh (Puspita, 2015). Penelitian tersebut dilakukan untuk menganalisis kombinasi dari ketiga metode kriptografi klasik diatas sehingga menghasilkan suatu teknik enkripsi data dengan tingkat keamanan yang sangat baik dan sulit untuk dipecahkan.

Implementasi Algoritma Kriptografi Kunci Publik Elgamal Untuk Proses Enkripsi Dan Dekripsi Guna Pengamanan File Data oleh (Aribowo, 2014). Penelitian ini mengimplementasikan sebuah program aplikasi menggunakan bahasa pemrograman *Visual Basic* untuk mengenkripsi dan mendekripsi file-file baik itu file txt, rtf ataupun pesan singkat menggunakan salah satu metode kriptografi yaitu Elgamal.

Implementasi Algoritma *Hill Cipher* Dalam Penyandian Data. Penelitian ini menjelaskan bahwa *Hill Cipher* termasuk kepada algoritma kriptografi klasik yang sangat sulit dipecahkan oleh kriptanalis apabila dilakukan hanya dengan mengetahui berkas ciphertext saja. Karena *Hill Cipher* tidak mengganti setiap abjad yang sama pada plaintext dengan abjad lainnya yang sama pada ciphertext karena menggunakan perkalian matriks pada dasar enkripsi dan dekripsinya (Halim, 2013).

Pengembangan Prototype Sistem Kriptografi Untuk Enkripsi Dan Dekripsi Data Office Menggunakan Metode Blowfish Dengan Bahasa Pemrograman Java, Penelitian yang dilakukan oleh (Natsir, 2016). Penelitian ini menghasilkan sebuah aplikasi desktop untuk mengenkripsi dan mendekripsi file-file *Ms.Office* berekstensi docx, xlsx dan

pptx. Selain file-file *Office* penelitian ini juga telah menguji enkripsi data seperti pdf, mp3 dan video berekstensi flv.

Penyembunyian Data Untuk Seluruh Ekstensi File Menggunakan Kriptografi *Vernam Cipher dan Bit Shifting*. Pada penelitian ini akan digunakan media berupa seluruh jenis file sebagai media inputan. Adapun algoritma yang digunakan yaitu Vernam cipher dan *Bit shifting*. Kedua algoritma ini dikenal cepat, mudah dan aman untuk digunakan. Percobaan yang dilakukan menggunakan 30 file berbeda ukuran maupun jenis file serta telah diuji melalui aplikasi yang dibangun dengan Visual Basic telah menghasilkan proses enkripsi dan dekripsi data yang berjalan dengan baik. File hasil enkripsi dapat dibuka dengan kunci yang telah ditetapkan dan tidak mengalami kerusakan, dan sebaliknya untuk proses dekripsi data juga demikian Hasil percobaan menggunakan sampling file berukuran 1 kb hingga 24000 kb, dimana waktu terlama untuk mengenkripsi file yaitu 28,661 detik dan untuk proses dekripsi terlama membutuhkan waktu 27,222 detik (Sari, 2016).

Menurut (Renaldy, 2015), dalam penelitiannya yang berjudul Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan Menggunakan Metode Aes-128 (*Advanced Encryption Standard-128*) Dan Sha-1 (*Secure Hash Algorithm-1*), mengatakan bahwa untuk menjaga keamanan privasi yang ditulis pada diary, penulis mencoba membuat aplikasi untuk menulis diary yang keamanannya dapat dijaga dengan cukup baik dan dapat digunakan dimana saja. Ada beberapa kekurangan yang penulis lihat dari kegiatan yang sudah umum ini, yaitu keamanan dari isi cerita kita yang masih bisa dilihat atau dicuri oleh orang lain jika masih menggunakan buku sebagai mediana. Aplikasi diary yang menggunakan metode enkripsi dan dapat dijalankan di *handphone* atau *smartphone* kita, sehingga bisa digunakan dimana saja karena dijalankan di *handphone*

kita serta cukup aman karena menggunakan enkripsi 2 (dua) metode yaitu AES-128 untuk isi diary dan SHA-1 untuk key nya.

Berikut ini merupakan tabel 2.1 perbandingan penelitian terkait yang telah dilakukan penelitiannya adalah sebagai berikut:

Tabel 2.1 Perbandingan Penelitian Terkait

No	Nama	Judul	Tahun	Hasil
1	Khairani Puspita, M. Rhifky Wayahdi.	Analisis Kombinasi Metode Caesar Cipher, Vernam Cipher, Dan Hill Cipher Dalam Proses Kriptografi	2015	Hasil yang disimpulkan bahwa metode Caesar Cipher, Vernam Cipher, dan Hill Cipher adalah jenis kriptografi klasik yang cukup kuat jika dilihat dari segi keamanannya dengan sedikit modifikasi. Ketiga metode ini dapat dikombinasikan menjadi satu dalam proses kriptografi (enkripsi dan dekripsi) dengan tingkat keamanan yang sangat baik dan sulit untuk dipecahkan.
2	Faqihuddin Al-Anshori dan Eko Aribowo.	Implementasi Algoritma Kriptografi Kunci Publik Elgamal Untuk Proses Enkripsi Dan Dekripsi Guna Pengamanan File Data	2014	Hasil dari penelitian ini akan diimplemtasikan dalam sebuah program aplikasi menggunakan bahasa pemrograman Visual Basic yang dapat

				memberikan kemudahan bagi setiap orang yang akan mengamankan file-file penting.
3	Abdul Halim Hasugian	Implementasi Algoritma Hill Cipher Dalam Penyandian Data	2013	Hasil dari penelitian adalah aplikasi yang dapat digunakan sebagai penyandian data yang berupa karakter yang berbentuk huruf dengan cara membagi perblok setiap karakter yang dienkripsi. Sistem ini dirancang dengan menggunakan bahasa pemrograman visual basic 6.0 dengan menggunakan algoritma hill cipher.
4	Mohamad Natsir	Pengembangan Prototype Sistem Kriptografi Untuk Enkripsi Dan Dekripsi Data <i>Office</i> Menggunakan Metode <i>Blowfish</i> Dengan Bahasa Pemrograman Java	2016	Menghasilkan aplikasi kriptografi yang dapat mengamankan data office menggunakan algoritma <i>Blowfish</i> . Penerapan kriptografi khususnya algoritma Blowfish dalam sistem pengamanan data office menggunakan enkripsi dan dekripsi dapat

				memberikan otentikasi, integritas dan non repudiation sebagai persyaratan penting bagi interaksi antara penerima dan pengirim data untuk identitas diri dari pemilik data.
5	Christy Atika Sari, Eko Hari Rachmawanto, Danang Wahyu Utomo, Ramadhan Rakhmat Sani.	Penyembunyian Data Untuk Seluruh Ekstensi File Menggunakan Kriptografi Vernam Cipher dan Bit Shiffting	2016	Aplikasi ini berhasil melakukan proses enkripsi dan dekripsi dengan baik menggunakan gabungan algoritma Vernam Cipher dan Bit Shiffting. Gabungan Vernam cipher dan Bit shiffting terbukti dapat melakukan proses enkripsi dan dekripsi menggunakan seluruh ekstensi file. Penelitian ini menggunakan 30 file dengan berbagai format dan ukuran file untuk menguji algoritma Vernam cipher dan Bit shifting.
6	Muammar Renaldy	Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan	2015	Aplikasi diary yang menggunakan metode enkripsi dan dapat

		Menggunakan Metode Aes-128 ( <i>Advanced Encryption Standard-128</i> ) Dan Sha-1 ( <i>Secure Hash Algorithm-1</i> )		dijalankan di <i>handphone</i> atau <i>smartphone</i> kita, sehingga bisa digunakan dimana saja karena dijalankan di <i>handphone</i> kita serta cukup aman karena menggunakan enkripsi 2 (dua) metode yaitu AES-128 untuk isi diary dan SHA-1 untuk <i>key</i> nya.
<b>Penelitian yang akan dilakukan</b>				
No	Nama	Judul	Tahun	Hasil
1	Billy Vansius Olo Pattiasina	Penerapan Enkripsi Dan Dekripsi Dokumen Word Menggunakan Kombinasi Metode Vernam Cipher Dan Hill Cipher	2020	Implementasi gabungan algoritma Vernam Cipher dan Hill Cipher pada aplikasi berbasis Web. Meningkatkan keamanan data pada file berekstensi docx. Mengkombinasikan kedua algoritma tersebut menjadi satu dalam proses kriptografi (enkripsi dan dekripsi) dengan tingkat keamanan yang sangat baik dan sulit untuk dipecahkan.

## 2.2. Aplikasi

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju (Andi Juansyah, 2015). Menurut kamus komputer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputasi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan.

Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu”

## 2.3. Sejarah Kriptografi

Kriptografi memiliki sejarah yang panjang dan mengagumkan. Penulisan rahasia ini dapat dilacak kembali ke 3000 tahun SM saat digunakan oleh bangsa Mesir. Mereka menggunakan *hieroglyphcs* untuk menyembunyikan tulisan dari mereka yang tidak diharapkan. *Hieroglyphcs* diturunkan dari bahasa Yunani *hieroglyphica* yang berarti ukiran rahasia. *Hieroglyphs* berevolusi menjadi *hieratic*, yaitu *stylized script* yang lebih mudah untuk digunakan.

Sekitar awal tahun 400 SM, kriptografi militer digunakan oleh bangsa Spartan di Yunani. Mereka menggunakan alat yang disebut *scytale*. *Scytale* merupakan pita panjang dari daun *papyrus* + sebatang silinder. Pesan ditulis horizontal (baris per baris). Bila pita dilepaskan, maka huruf-huruf di dalamnya telah tersusun membentuk pesan rahasia. Untuk membaca pesan, penerima melilitkan kembali silinder yang diameternya sama

dengan diameter silinder pengirim. Sejarah lengkap kriptografi dapat ditemukan di dalam buku David Kahn, "*The Codebreakers*". Empat kelompok orang yang menggunakan dan berkontribusi pada kriptografi:

1. Militer (termasuk intelijen dan mata-mata)
2. Korp diplomatik
3. *Diarist*
4. *Lovers*

Kriptografi juga digunakan untuk alasan keagamaan, untuk menjaga tulisan religius dari gangguan otoritas politik atau budaya yang dominan saat itu. Contoh: "666" atau "Angka si Buruk Rupa (*Number of the Beast*) di dalam Kitab Perjanjian Baru.

Sekitar 50 SM, Julius Caesar, kaisar Roma, menggunakan cipher substitusi untuk mengirim pesan ke Marcus Tullius Cicero. Pada cipher ini, huruf-huruf alfabet disubstitusikan dengan huruf-huruf yang lain pada alfabet yang sama. Karena hanya satu alfabet yang digunakan, cipher ini merupakan substitusi *monoalfabetik*. Cipher semacam ini mencakup penggeseran alfabet dengan 3 huruf dan mensubstitusikan huruf tersebut. Substitusi ini kadang dikenal dengan C3 (untuk Caesar menggeser 3 tempat). Secara umum sistem cipher Caesar dapat ditulis sebagai berikut:

$$Z_i = C_n (P_i)$$

Dimana  $Z_i$  adalah karakter-karakter ciphertext,  $C_n$  adalah transformasi substitusi alfabetik,  $n$  adalah jumlah huruf yang digeser, dan  $P_i$  adalah karakter-karakter plaintext.

*Disk* mempunyai peranan penting dalam kriptografi sekitar 500 tahun yang lalu. Di Italia sekitar tahun 1460, Leon Battista Alberti mengembangkan *disk cipher* untuk enkripsi. Sistemnya terdiri dari dua disk konsentris. Setiap *disk* memiliki alfabet di sekelilingnya, dan



dengan memutar satu disk berhubungan dengan yang lainnya, huruf pada satu alfabet dapat ditransformasi ke huruf pada alfabet yang lain.

Tidak ditemukan catatan kriptografi di Cina dan Jepang hingga abad 15. Di India, kriptografi digunakan oleh pencinta (*lovers*) untuk berkomunikasi tanpa diketahui orang. Bukti ini ditemukan di dalam buku Kama Sutra yang merekomendasikan wanita seharusnya mempelajari seni memahami tulisan dengan cipher.

Pada Abad ke-17, sejarah kriptografi pernah mencatat korban di Inggris. *Queen Mary of Scotland*, dipancung setelah pesan rahasianya dari balik penjara (pesan terenkripsi yang isinya rencana membunuh Ratu Elizabeth I) pada Abad Pertengahan berhasil dipecahkan oleh Thomas Phelippes, seorang pemecah kode. Bangsa Arab menemukan *cryptanalysis* karena kemahirannya dalam bidang matematika, statistik, dan linguistik. Karena setiap orang muslim harus menambah pengetahuannya, mereka mempelajari peradaban terdahulu dan mendekodekan tulisan-tulisannya ke huruf-huruf Arab. Pada tahun 815, Caliph al-Mamun mendirikan *House of Wisdom* di Baghdad yang merupakan titik pusat dari usaha-usaha translasi. Pada abad ke-9, filsuf Arab al-Kindi menulis risalat (ditemukan kembali tahun 1987) yang diberi judul "*A Manuscript on Deciphering Cryptographic Messages*".

Pada 1790, Thomas Jefferson mengembangkan alat enkripsi dengan menggunakan tumpukan yang terdiri dari 26 *disk* yang dapat diputar secara individual. Pesan dirakit dengan memutar setiap *disk* ke huruf yang tepat di bawah batang berjajar yang menjalankan panjang tumpukan *disk*. Kemudian, batang berjajar diputar dengan sudut tertentu, A, dan huruf-huruf di bawah batang adalah pesan yang terenkripsi. Penerima akan menjajarkan karakter-karakter cipher di bawah batang berjajar, memutar

batang kembali dengan sudut A dan membaca pesan *plaintext*. Sistem *disk* digunakan secara luas selama perang sipil US. *Federal Signal Officer* mendapatkan hak paten pada sistem *disk* mirip dengan yang ditemukan oleh Leon Battista Alberti di Italia, dan dia menggunakannya untuk mengkode dan mendekodekan sinyal-sinyal bendera diantara unit-unit.

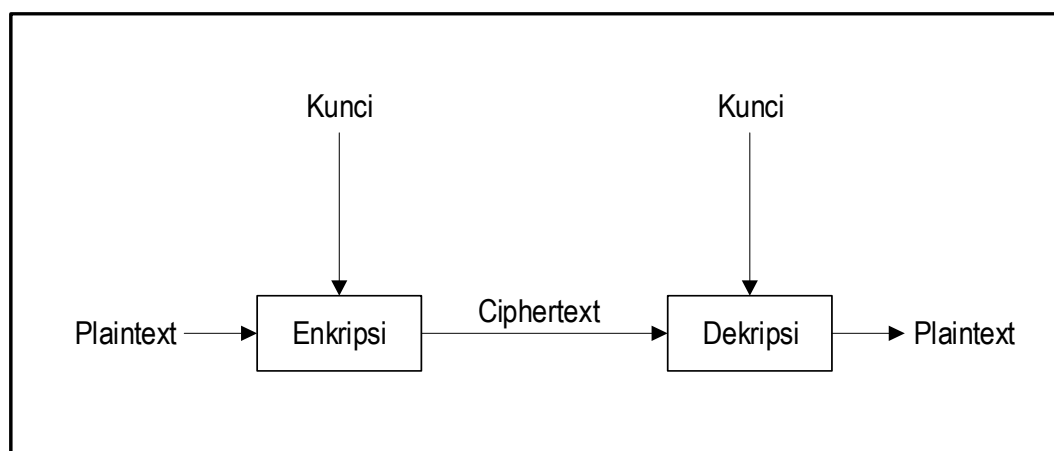
Pada Perang Dunia ke II, Pemerintah Nazi Jerman membuat mesin enkripsi yang dinamakan Enigma. *Enigmapher* berhasil dipecahkan oleh pihak Sekutu. Keberhasilan memecahkan *Enigmasering* dikatakan sebagai faktor yang memperpendek perang dunia ke-2.

Sistem *Unix* menggunakan cipher substitusi yang disebut ROT 13 yang menggeser alfabet sebanyak 13 tempat. Penggeseran 13 tempat yang lain membawa alfabet kembali ke posisi semula, dengan demikian mendekodekan pesan. Mesin kriptografi mekanik yang disebut *Hagelin Machine* dibuat pada tahun 1920 oleh Boris Hagelin di Sockholm, Swedia. Di US, mesin Hagelin dikenal sebagai M-209. Pada tahun 20-an, Herbert O. Yardley bertugas pada organisasi rahasia US MI-8 yang dikenal sebagai "*Black Chamber*". MI-8 menjebol kode-kode sejumlah negara. Selama konferensi Angkatan Laut Washington tahun 1921-1922, US membatasi negosiasi dengan Jepang karena MI-8 telah memberikan rencana negosiasi Jepang yang telah disadap kepada sekretaris negara US. Departemen negara menutup MI-8 pada tahun 1929 sehingga Yardley merasa kecewa. Sebagai wujud kekecewaanya, Yardley menerbitkan buku *The American Black Chamber*, yang menggambarkan kepada dunia rahasia dari MI-8. Sebagai konsekuensinya, pihak Jepang menginstal kode-kode baru. Karena kepeloporannya dalam bidang ini, Yardley dikenal sebagai "Bapak Kriptografi Amerika" (Wirdasari, 2008).

## 2.4. Kriptografi

Kriptografi adalah ilmu yang mempelajari tentang cara menjaga keamanan suatu pesan atau informasi. Pesan atau informasi dapat dikategorikan ke dalam dua jenis, yaitu pesan yang dapat dibaca dengan mudah *plaintext* dan pesan yang tidak mudah dibaca *ciphertext* (Nugroho, 2010).

Untuk melakukan kriptografi digunakan algoritma kriptografi. Algoritma kriptografi terdiri dari dua bagian, yaitu fungsi enkripsi dan dekripsi. Enkripsi adalah proses untuk mengubah *plaintext* menjadi *ciphertext*, sedangkan dekripsi adalah kebalikannya yaitu mengubah *ciphertext* menjadi *plaintext*. Proses enkripsi dan dekripsi memerlukan kunci dalam mekanismenya dan biasanya berupa *string* atau deretan bilangan. Proses enkripsi dan dekripsi yang digunakan dalam pengiriman pesan dapat dilihat pada Gambar 2.1.



Gambar 2.1 Skema Enkripsi dan Dekripsi dengan Menggunakan Kunci (Sumandri, 2017)

Terdapat dua jenis algoritma kriptografi berdasar jenis kuncinya, yaitu :

1. Algoritma Simetri adalah algoritma yang menggunakan kunci yang sama untuk enkripsi dan dekripsinya (Sumandri, 2017). Algoritma kriptografi simetris sering disebut algoritma kunci rahasia, algoritma kunci tunggal, atau algoritma satu kunci, dan mengharuskan pengirim dan penerima menyetujui suatu kunci tertentu.

Kelebihan dari algoritma kriptografi simetris adalah waktu proses untuk enkripsi dan dekripsi relatif cepat. Hal ini disebabkan efisiensi yang terjadi pada pembangkit kunci. Karena prosesnya relative cepat maka algoritma ini tepat untuk digunakan pada sistem komunikasi digital secara *real time* seperti GSM. Aplikasi dari algoritma simetris digunakan oleh beberapa algoritma seperti: *Data Encryption Standard* (DES), *Advance Encryption Standard* (AES), *International Data Encryption Algoritma* (IDEA), A5, dan RC4.

2. Algoritma Asimetris adalah pasangan kunci kriptografi yang salah satunya digunakan untuk proses enkripsi dan satu lagi lagi deskripsi (Sumandri, 2017). Semua orang yang mendapatkan kunci publik dapat menggunakannya untuk mengenkripsi suatu pesan, sedangkan hanya satu orang saja yang memiliki rahasia itu, yang dalam hal ini kunci rahasia, untuk melakukan pembongkaran terhadap kode yang dikirim untuknya. Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA (merupakan singkatan dari nama penemunya, yakni Rivest, Shamir dan Adleman).

## 2.5. Tujuan Kriptografi

Tujuan dari kriptografi yang juga merupakan aspek keamanan informasi adalah sebagai berikut:

1. Kerahasiaan (*confidentiality*) adalah layanan yang digunakan untuk menjaga isi informasi dari semua pihak kecuali pihak yang memiliki otoritas terhadap informasi. Ada beberapa pendekatan untuk menjaga kerahasiaan, dari pengamanan secara fisik hingga penggunaan algoritma matematika yang membuat data tidak dapat dipahami. Istilah lain yang senada dengan

*confidentiality* adalah *secrecy* dan *privacy*.

2. Integritas data adalah layanan penjagaan perubahan data dari pihak yang tidak berwenang. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubtitusian data lain ke dalam pesan yang sebenarnya. Di dalam kriptografi, layanan ini direalisasikan dengan menggunakan tanda-tangan digital (*digital signature*). Pesan yang telah ditandatangani menyiratkan bahwa pesan yang dikirim adalah asli.
3. Otentikasi adalah layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication* atau *entity authentication*) maupun mengidentifikasi kebenaran sumber pesan (*data origin authentication*). Dua pihak yang saling berkomunikasi harus dapat mengotentikasi satu sama lain sehingga ia dapat memastikan sumber pesan. Pesan yang dikirim melalui saluran komunikasi juga harus diotentikasi asalnya. Otentikasi sumber pesan secara implisit juga memberikan kepastian integritas data, sebab jika pesan telah dimodifikasi berarti sumber pesan sudah tidak benar. Oleh karena itu, layanan integritas data selalu dikombinasikan dengan layanan otentikasi sumber pesan. Di dalam kriptografi, layanan ini direalisasikan dengan menggunakan tanda-tangan digital (*digital signature*). Tanda-tangan digital menyatakan sumber pesan.
4. Nirpenyangkalan (*non-repudiation*) adalah layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan (Zelviana, 2012).

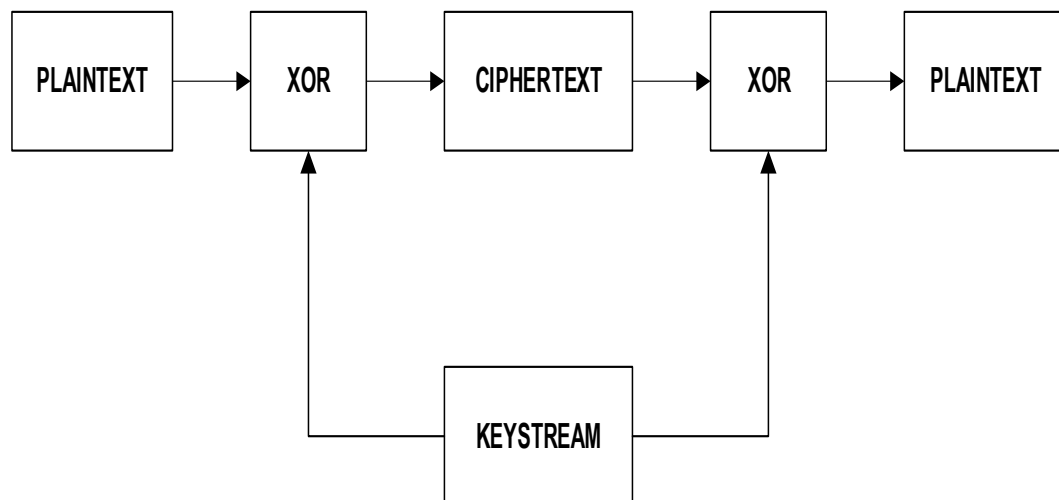
## 2.6. Algoritma Kriptografi Klasik

Algoritma kriptografi klasik digunakan sejak sebelum era komputerisasi dan kebanyakan menggunakan teknik kunci simetris. Metode menyembunyikan pesannya adalah dengan teknik substitusi atau transposisi atau keduanya. Teknik substitusi adalah menggantikan karakter dalam *plaintext* menjadi karakter lain yang hasilnya adalah *ciphertext*. Sedangkan transposisi adalah teknik mengubah *plaintext* menjadi *ciphertext* dengan cara permutasi karakter. Kombinasi keduanya secara kompleks adalah yang melatarbelakangi terbentuknya berbagai macam algoritma kriptografi modern. Beberapa contoh algoritma kriptografi klasik yaitu *Caesar Cipher*, *Vigenere Cipher*, *Vernam Cipher*, dan *Hill Cipher*.

## 2.7. Algoritma Vernam Cipher

*Vernam cipher* merupakan algoritma kriptografi yang ditemukan oleh Mayor J. Maugborne dan G. Vernam. Algoritma *Vernam cipher* diadopsi dari *one-time pad cipher*, dimana dalam hal ini karakter diganti dengan bit (0 atau 1). Dengan kata lain, *vernham cipher* merupakan versi lain dari *one-time pad cipher* (Jumeidi, 2016).

Algoritma kriptografi *Vernam Cipher* merupakan algoritma kriptografi berjenis *symmetric key*. Kunci yang digunakan untuk melakukan enkripsi dan dekripsi menggunakan kunci yang sama. Dalam melakukan proses enkripsi, algoritma *vernham cipher* menggunakan cara *stream cipher* dimana *cipher* berasal dari hasil operasi XOR antara bit *plaintext* dan bit *key*. Pada cipher aliran, bit hanya mempunyai dua buah nilai, sehingga proses enkripsi hanya menyebabkan dua keadaan pada bit tersebut, yaitu berubah atau tidak berubah (Jumeidi, 2016). Secara sederhana proses kriptografi (enkripsi dan dekripsi) pada algoritma *Vernam Cipher* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Proses Enkripsi dan Dekripsi Algoritma *Vernam cipher*  
(Jumeidi, 2016)

Pembentukan *keystream* dapat dibuat independen terhadap *plaintext* dan *ciphertext*, menghasilkan *synchronous stream cipher*, atau dapat dibuat tergantung pada data dan enkripsinya, dalam hal mana *stream cipher* disebut sebagai *self-synchronizing*. Kebanyakan bentuk *stream cipher* adalah *synchronous stream cipher*.

*Keystream* memiliki panjang sama dengan pesan *plaintext*; *string random* digabungkan dengan menggunakan *bitwise XOR* dengan *plaintext* untuk menghasilkan *ciphertext*. Karena *keystream* seluruhnya adalah *random*, walaupun dengan sumber daya komputasi tak terbatas seseorang hanya dapat menduga *plaintext* jika melihat *ciphertext*. Metode *cipher* seperti ini disebut memberikan kerahasiaan yang sempurna (*perfect secrecy*).

Satu hal yang menarik bahwa mode operasi tertentu dari suatu *block cipher* dapat mentransformasikan secara efektif hasil operasi tersebut ke dalam satu *keystream generator* dan dalam hal ini, *block cipher* apa saja dapat digunakan sebagai suatu *stream cipher*, seperti dalam DES, CFB atau OFB. Akan tetapi, *vernem cipher* dengan desain khusus biasanya jauh lebih cepat (Eko Hari Rachmawanto, 2016).

*Cipherteks* diperoleh dengan melakukan penjumlahan *modulo 2* satu bit *plainteks* dengan satu bit kunci:

$$C_i = (P_i + K_i) \bmod 26 \quad (1)$$

Dimana,  $P_i$  adalah bit *plainteks*,  $K_i$  adalah bit kunci, dan  $C_i$  adalah bit *cipherteks*. *Plainteks* diperoleh dengan melakukan penjumlahan *modulo 2* satu bit *cipherteks* dengan satu bit kunci:

$$C_i = (P_i - K_i) \bmod 26 \quad (2)$$

Aliran-bit-kunci dibangkitkan dari sebuah pembangkit yang dinamakan pembangkit aliran-bit-kunci (*keystream generator*). Aliran-bit-kunci (sering dinamakan *running key*) di-XOR-kan dengan aliran bit-bit *plainteks*,  $P_1, P_2, \dots, P_i$ , untuk menghasilkan aliran bit-bit *cipherteks*:

$$C_i = P_i \oplus K_i \quad (3)$$

Di sisi penerima, bit-bit *cipherteks* di-XOR-kan dengan aliran-bit-kunci yang sama untuk menghasilkan bit-bit *plainteks*:

$$P_i = C_i \oplus K_i \quad (4)$$

Merancang pembangkit bit-aliran-kunci yang bagus cukup sulit karena membutuhkan pengujian statistik untuk menjamin bahwa keluaran dari pembangkit tersebut sangat mendekati barisan acak yang sebenarnya (Eko Hari Rachmawanto, 2016).

## 2.8. Algoritma Hill Cipher

Algoritma *Hill Cipher* adalah suatu fungsi matematis yang digunakan untuk melakukan enkripsi dan dekripsi. Ada dua macam algoritma kriptografi, yaitu algoritma simetris (*symmetric algorithm*) dan algoritma asimetris (*asymmetric algorithms*). *Hill Cipher* merupakan *poly alphabetic cipher* dapat dikategorikan sebagai *block cipher*, karena teks



yang akan diproses dibagi menjadi blok-blok dengan ukuran tertentu. Setiap karakter dalam satu blok akan saling mempengaruhi karakter lainnya dalam proses enkripsi dan dekripsinya, sehingga karakter yang sama tidak dipetakan menjadi karakter yang sama pula (Wahyuningsih, 2016).

### 2.8.1. Sejarah *Hill Cipher*

Sejak kekaisaran Romawi, kriptosistem yang lebih rumit dikembangkan oleh orang seperti ahli Matematika Italia Leon Battista Alberti (lahir pada tahun 1404), Matematikawan Jerman Johannes Trithemius (lahir pada tahun 1492), seorang kriptographer dan diplomat Perancis Blaise de Vigenere (1523-1596), Lester S. Hill, yang menemukan *Hill Cipher* pada tahun 1929. *Hill Cipher* merupakan jenis lain dari *polygraphic cipher*. Sandi ini mengenkripsi suatu *string* huruf menjadi bentuk *string* yang lain dengan panjang yang sama. Teknik *Hill Cipher* dikembangkan oleh Lester S. Hill pada *Hunter Collage* dan dipublikasikan pada *American Mathematical Monthly*, Volume 36, Issue 6 (Juni-Juli, 1929) halaman 302-312.

*Hill Cipher* menggunakan matriks untuk mentransformasikan *string* berupa blok huruf. Berdasarkan pada aljabar linier dan sandi *Vigenere*, *Hill Cipher* merupakan *block cipher*. *Hill Cipher* tidak mengganti setiap abjad yang sama pada *plaintext* dengan abjad lainnya yang sama pada *chipertext* karena menggunakan perkalian matriks pada dasar enkripsi dan dekripsinya. *Hill Cipher* termasuk pada algoritma kriptografi klasik yang sangat sulit dipecahkan oleh kriptanalis apabila dilakukan hanya dengan mengetahui berkas *chipertext* saja. Namun, teknik ini dapat dipecahkan dengan cukup mudah apabila kriptanalis memiliki berkas *chipertext* dan potongan berkas *plaintext*. Teknik kriptanalis ini disebut *known-plaintext attack* (Wahyuningsih, 2016).

### 2.8.2. Dasar Teknik Hill Cipher

Dasar dari teknik *Hill Cipher* adalah aritmatika *modulo* terhadap matriks. Dalam penerapannya, *Hill Cipher* menggunakan teknik perkalian matriks dan Teknik *invers* terhadap matriks. Kunci pada *Hill Cipher* adalah matriks  $n \times n$  dengan  $n$  juga merupakan ukuran blok. Jika kunci disebut dengan  $K$ , maka  $K$  dapat dirumuskan sebagai berikut (Wahyuningsih, 2016):

$$K = \begin{bmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{bmatrix}$$

Matriks  $K$  yang menjadi kunci harus merupakan matriks yang *invertible*, yaitu memiliki *inverse*  $K^{-1}$ , sehingga:

$$K \cdot K^{-1} = I$$

Kunci harus memiliki *invers* karena matriks  $K^{-1}$  tersebut adalah kunci yang digunakan untuk melakukan dekripsi.

### 2.8.3. Enkripsi pada Hill Cipher

Proses enkripsi pada *Hill Cipher* dilakukan per blok *plaintext*. Ukuran blok tersebut sama dengan ukuran matriks kunci. Sebelum membagi teks menjadi deretan blok-blok, *plaintext* terlebih dahulu dikonversi menjadi angka. Secara sistematis, proses enkripsi pada *Hill Cipher* adalah:

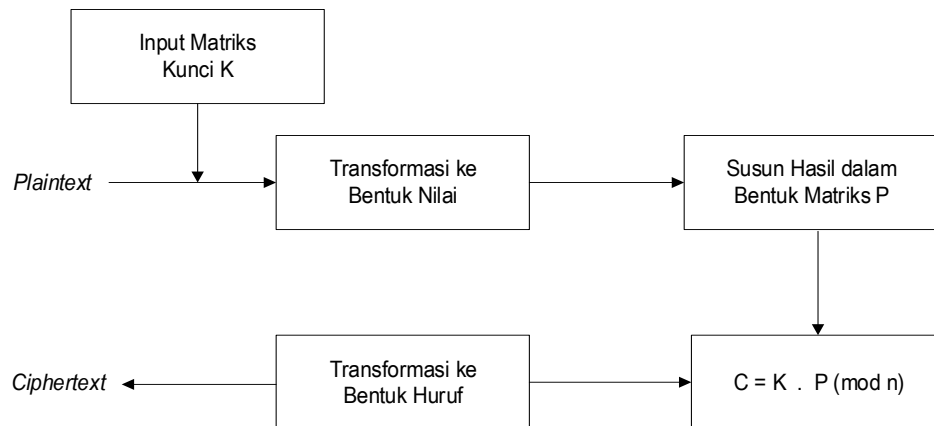
$$C = K \cdot P \dots\dots\dots (1)$$

Keterangan:

$C$  = *Chipertext*

$K$  = Kunci

$P$  = *Plaintext*



Gambar 2.3 Ilustrasi Proses Enkripsi *Hill Cipher* (Wahyuningsih, 2016)

Proses enkripsi pada *hill cipher* dilakukan per blok plainteks. Ukuran blok tersebut sama dengan ukuran matriks kunci. Sebelum membagi teks menjadi deretan blok-blok, plainteks terlebih dahulu dikonversi menjadi angka, masing-masing sehingga A=0, B=1, hingga Z=25.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Secara matematis, proses enkripsi pada hill cipher adalah:

$$C = K \cdot P$$

$$C = \text{Ciphertext}$$

$$K = \text{Kunci}$$

$$P = \text{Plaintext}$$

Contoh kasus misalnya ingin menyembunyikan sebuah pesan berisi nama ABDUL HALIM dengan kunci matriks ukuran  $2 \times 2$ ,  $K = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$ , kemudian bagi plainteks sesuai dengan ukuran matriks.

Blok I	Blok II	Blok III	Blok IV	Blok V
A B	D U	L H	A L	I M
0 1	3 20	11 7	0 11	8 12

Blok I

$$\begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 * 0 + 3 * 1 \\ 3 * 0 + 2 * 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Setelah itu hasil di mod 26 seperti berikut:

$$3 \bmod 26 = 3$$

$$2 \bmod 26 = 2$$

Hasilnya 3 dan 2 atau D dan C.

Blok II

$$\begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 20 \end{bmatrix} = \begin{bmatrix} 5 * 3 + 3 * 20 \\ 3 * 3 + 2 * 20 \end{bmatrix} = \begin{bmatrix} 75 \\ 49 \end{bmatrix}$$

Setelah itu hasil di mod 26 seperti berikut:

$$75 \bmod 26 = 23$$

$$49 \bmod 26 = 23$$

Hasilnya 23 dan 23 atau X dan X.

Blok III

$$\begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 11 \\ 7 \end{bmatrix} = \begin{bmatrix} 5 * 11 + 3 * 7 \\ 3 * 11 + 2 * 7 \end{bmatrix} = \begin{bmatrix} 76 \\ 47 \end{bmatrix}$$

Setelah itu hasil di mod 26 seperti berikut:

$$76 \bmod 26 = 24$$

$$47 \bmod 26 = 21$$

Hasilnya 24 dan 21 atau Y dan V.

Blok IV

$$\begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 11 \end{bmatrix} = \begin{bmatrix} 5 * 0 + 3 * 11 \\ 3 * 0 + 2 * 11 \end{bmatrix} = \begin{bmatrix} 33 \\ 22 \end{bmatrix}$$

Setelah itu hasil di mod 26 seperti berikut:

$$33 \bmod 26 = 7$$

$$22 \bmod 26 = 22$$

Hasilnya 24 dan 21 atau H dan W.

Blok V

$$\begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 8 \\ 12 \end{bmatrix} = \begin{bmatrix} 5 * 8 + 3 * 12 \\ 3 * 8 + 2 * 12 \end{bmatrix} = \begin{bmatrix} 76 \\ 48 \end{bmatrix}$$

Setelah itu hasil di mod 26 seperti berikut:

$$76 \bmod 26 = 24$$

$$48 \bmod 26 = 22$$

Hasilnya 24 dan 21 atau Y dan W.

Maka hasil ciphertext dari ABDUL HALIM adalah DCXXYVHWYW (Halim, 2013). Hasil enkripsi akan didekripsi dengan cara membagi dengan blok pada hasil ciphertexts ini.

#### 2.8.4. Dekripsi Pada Hill Cipher

Proses dekripsi pada *Hill Cipher* pada dasarnya sama dengan proses enkripsinya. Namun, matriks kunci harus dibalik (*invers*) terlebih dahulu. Secara matematis, proses dekripsi pada *Hill Cipher* dapat diturunkan pada persamaan:

$$C = K \cdot P$$

$$K^{-1} \cdot C = K^{-1} \cdot K \cdot P$$

$$K^{-1} \cdot C = I \cdot P$$

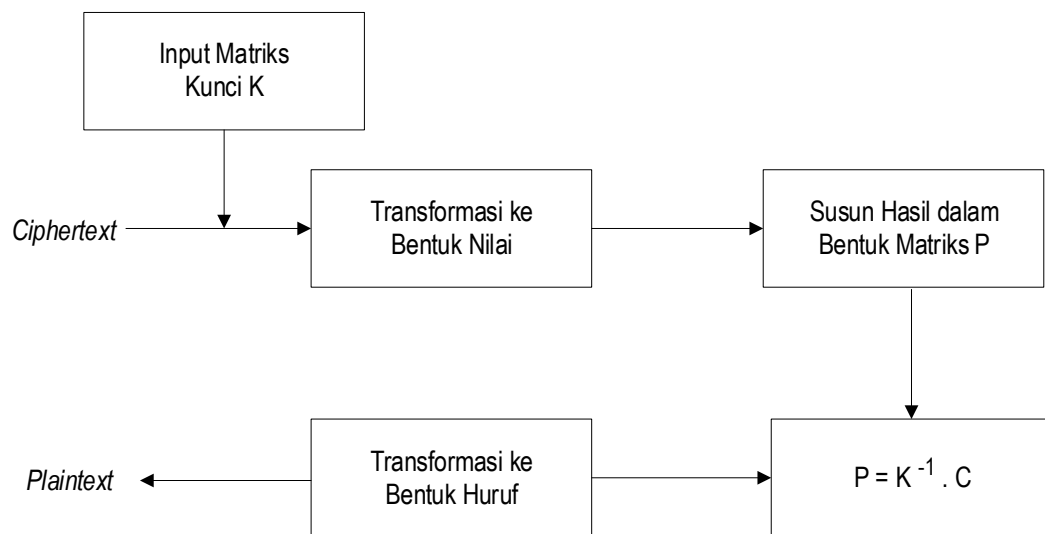
$$P = K^{-1} \cdot C$$

Sehingga diperoleh persamaan dekripsinya:

$$P = K^{-1} \cdot C \dots\dots\dots (2)$$

Dimana untuk menentukan invers  $K$  atau yang biasa disingkat  $K^{-1}$  dengan menggunakan rumus:

$$K^{-1} = \frac{1}{|K|} \text{Adj} (K)$$



Gambar 2.4 Ilustrasi Proses Dekripsi *Hill Cipher* (Wahyuningsih, 2016)

Proses dekripsi diawali dengan menghitung *invers* dari matriks  $K$ . Maka proses dekripsi sebagai berikut:

$$K = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix} \text{ mod } 26$$

$$\text{Det} (K) = 5 \times 2 - 3 \times 3 = 1$$

Maka untuk mencari  $K^{-1}$  adalah:

$$K^{-1} = \frac{1}{K} \text{Adj} (K)$$

$$K^{-1} = \frac{1}{1} \begin{bmatrix} 2 & -3 \\ -3 & 5 \end{bmatrix}$$

$$K^{-1} = \begin{bmatrix} 2 & -3 \\ -3 & 5 \end{bmatrix}$$

Setiap yang bernilai negatif ditambah 26 agar nilai tetap positif, ini digunakan karena yang digunakan yaitu 0-25, sehingga  $K^{-1} = \begin{bmatrix} 2 & 23 \\ 23 & 5 \end{bmatrix}$ . Hasil dari enkripsi tersebut, yaitu DCXXYVHWYW akan didekripsi dengan cara membagi dengan blok pada hasil *ciphertext* yang sudah ada. Proses dekripsinya sama dengan proses enkripsi, namun dengan kunci yang berbeda. Setelah semua blok selesai didekripsi, maka didapatkan plainteks semula, yaitu ABDULHALIM (Halim, 2013).

## 2.9. ASCII (American Standard Code for Information Interchange)

ASCII (American Standard Code for Information Interchange) merupakan Kode Standar Amerika untuk Pertukaran Informasi atau sebuah standar internasional dalam pengkodean huruf dan simbol seperti Unicode dan Hex tetapi ASCII lebih bersifat universal (Nurhidayat, 2018).

Tabel 2.2 Tabel konversi standarisasi ASCII ([www.ascii-code.com](http://www.ascii-code.com))

Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
0	00000000	NUL	Null (tidak terlihat)
1	00000001	SOH	Start of heading (tidak terlihat)
2	00000010	STX	Start of text (tidak terlihat)
3	00000011	ETX	End of text (tidak terlihat)
4	00000100	EOT	End of transmission (tidak terlihat)
5	00000101	ENQ	Enquiry (tidak terlihat)
6	00000110	ACK	Acknowledge (tidak terlihat)
7	00000111	BEL	Bell (tidak terlihat)
8	00001000	BS	Backspace
9	00001001	HT	Horizontal tabulation
10	00001010	LF	Pergantian baris (Line feed)
11	00001011	VT	Tabulasi vertikal
12	00001100	FF	Pergantian baris (Form feed)
13	00001101	CR	Pergantian baris (carriage return)
14	00001110	SO	Shift out (tidak terlihat)
15	00001111	SI	Shift in (tidak terlihat)
16	00010000	DLE	Data link escape (tidak terlihat)

Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
17	00010001	DC1	Device control 1 (tidak terlihat)
18	00010010	DC2	Device control 2 (tidak terlihat)
19	00010011	DC3	Device control 3 (tidak terlihat)
20	00010100	DC4	Device control 4 (tidak terlihat)
21	00010101	NAK	Negative acknowledge (tidak terlihat)
22	00010110	SYN	Synchronous idle (tidak terlihat)
23	00010111	ETB	End of transmission block (tidak terlihat)
24	00011000	CAN	Cancel (tidak terlihat)
25	00011001	EM	End of medium (tidak terlihat)
26	00011010	SUB	Substitute (tidak terlihat)
27	00011011	ESC	Escape (tidak terlihat)
28	00011100	FS	File separator
29	00011101	GS	Group separator
30	00011110	RS	Record separator
31	00011111	US	Unit separator
32	00100000	spasi	Spasi
33	00100001	!	Tanda seru (exclamation)
34	00100010	"	Tanda kuti dua
35	00100011	#	Tanda pagar (kres)
36	00100100	\$	Tanda mata uang dolar
37	00100101	%	Tanda persen
38	00100110	&	Karakter ampersand (&)
39	00100111	'	Karakter Apostrof
40	00101000	(	Tanda kurung buka
41	00101001	)	Tanda kurung tutup
42	00101010	*	Karakter asterisk (bintang)
43	00101011	+	Tanda tambah (plus)
44	00101100	,	Karakter koma
45	00101101	-	Karakter hyphen (strip)
46	00101110	.	Tanda titik
47	00101111	/	Garis miring (slash)
48	00110000	0	Angka nol
49	00110001	1	Angka satu
50	00110010	2	Angka dua
51	00110011	3	Angka tiga
52	00110100	4	Angka empat
53	00110101	5	Angka lima
54	00110110	6	Angka enam



Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
55	00110111	7	Angka tujuh
56	00111000	8	Angka delapan
57	00111001	9	Angka sembilan
58	00111010	:	Tanda titik dua
59	00111011	;	Tanda titik koma
60	00111100	<	Tanda lebih kecil
61	00111101	=	Tanda sama dengan
62	00111110	>	Tanda lebih besar
63	00111111	?	Tanda tanya
64	01000000	@	A keong (@)
65	01000001	A	Huruf latin A kapital
66	01000010	B	Huruf latin B kapital
67	01000011	C	Huruf latin C kapital
68	01000100	D	Huruf latin D kapital
69	01000101	E	Huruf latin E kapital
70	01000110	F	Huruf latin F kapital
71	01000111	G	Huruf latin G kapital
72	01001000	H	Huruf latin H kapital
73	01001001	I	Huruf latin I kapital
74	01001010	J	Huruf latin J kapital
75	01001011	K	Huruf latin K kapital
76	01001100	L	Huruf latin L kapital
77	01001101	M	Huruf latin M kapital
78	01001110	N	Huruf latin N kapital
79	01001111	O	Huruf latin O kapital
80	01010000	P	Huruf latin P kapital
81	01010001	Q	Huruf latin Q kapital
82	01010010	R	Huruf latin R kapital
83	01010011	S	Huruf latin S kapital
84	01010100	T	Huruf latin T kapital
85	01010101	U	Huruf latin U kapital
86	01010110	V	Huruf latin V kapital
87	01010111	W	Huruf latin W kapital
88	01011000	X	Huruf latin X kapital
89	01011001	Y	Huruf latin Y kapital
90	01011010	Z	Huruf latin Z kapital
91	01011011	[	Kurung siku kiri
92	01011100	/	Garis miring terbalik ( <i>backslash</i> )
93	01011101	]	Kurung sikur kanan

Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
94	01011110	^	Tanda pangkat
95	01011111	_	Garis bawah (underscore)
96	01100000	`	Tanda petik satu
97	01100001	a	Huruf latin a kecil
98	01100010	b	Huruf latin b kecil
99	01100011	c	Huruf latin c kecil
100	01100100	d	Huruf latin d kecil
101	01100101	e	Huruf latin e kecil
102	01100110	f	Huruf latin f kecil
103	01100111	g	Huruf latin g kecil
104	01101000	h	Huruf latin h kecil
105	01101001	i	Huruf latin i kecil
106	01101010	j	Huruf latin j kecil
107	01101011	k	Huruf latin k kecil
108	01101100	l	Huruf latin l kecil
109	01101101	m	Huruf latin m kecil
110	01101110	n	Huruf latin n kecil
111	01101111	o	Huruf latin o kecil
112	01110000	p	Huruf latin p kecil
113	01110001	q	Huruf latin q kecil
114	01110010	r	Huruf latin r kecil
115	01110011	s	Huruf latin s kecil
116	01110100	t	Huruf latin t kecil
117	01110101	u	Huruf latin u kecil
118	01110110	v	Huruf latin v kecil
119	01110111	w	Huruf latin w kecil
120	01111000	x	Huruf latin x kecil
121	01111001	y	Huruf latin y kecil
122	01111010	z	Huruf latin z kecil
123	01111011	{	Kurung kurawal buka
124	01111100		Garis vertikal (pipa)
125	01111101	}	Kurung kurawal tutup
126	01111110	~	Karakter gelombang (tilde)
127	01111111	DEL	Delete
128	10000000	€	Euro sign
129	10000001		
130	10000010	,	Single low-9 quotation mark
131	10000011	<i>f</i>	Latin small letter f with hook
132	10000100	”	Double low-9 quotation mark

Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
133	10000101	...	Horizontal ellipsis
134	10000110	†	Dagger
135	10000111	‡	Double dagger
136	10001000	^	Modifier letter circumflex accent
137	10001001	‰	Per mille sign
138	10001010	Š	Latin capital letter S with caron
139	10001011	‹	Single left-pointing angle quotation
140	10001100	Œ	Latin capital ligature OE
141	10001101		
142	10001110	Ž	Latin capital letter Z with caron
143	10001111		
144	10010000		
145	10010001	‘	Left single quotation mark
146	10010010	’	Right single quotation mark
147	10010011	“	Left double quotation mark
148	10010100	”	Right double quotation mark
149	10010101	•	Bullet
150	10010110	—	En dash
151	10010111	—	Em dash
152	10011000	~	Small tilde
153	10011001	™	Trade mark sign
154	10011010	š	Latin small letter S with caron
155	10011011	›	Single right-pointing angle quotation mark
156	10011100	œ	Latin small ligature oe
157	10011101		
158	10011110	ž	Latin small letter z with caron
159	10011111	ÿ	Latin capital letter Y with diaeresis
160	10100000		Spasi yang bukan pemisah kata
161	10100001	¡	Tanda seru terbalik
162	10100010	¢	Tanda sen (Cent)
163	10100011	£	Tanda Poundsterling
164	10100100	¤	Tanda mata uang (Currency)
165	10100101	¥	Tanda Yen
166	10100110	¦	Garis tegak putus-putus
167	10100111	§	Section sign
168	10101000	¨	Spacing diaeresis – umlaut
169	10101001	©	Tanda hak cipta (Copyright)
170	10101010	ª	Feminine ordinal indicator

Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
171	10101011	«	Left double angle quotes
172	10101100	¬	Not sign
173	10101101		Tanda strip (hyphen)
174	10101110	®	Tanda merk terdaftar
175	10101111	—	Spacing Macron (Macron)
176	10110000	°	Tanda derajat
177	10110001	±	Tanda kurang lebih (plus-minus)
178	10110010	²	Tanda kuadrat (pangkat dua)
179	10110011	³	Tanda kubik (pangkat tiga)
180	10110100	´	Acute accent
181	10110101	μ	Micro sign
182	10110110	¶	Pilcrow sign
183	10110111	·	Middle dot
184	10111000	¸	Spacing cedilla
185	10111001	¹	Superscript one
186	10111010	º	Masculine ordinal indicator
187	10111011	»	Right double angle quotes
188	10111100	¼	Fraction one quarter
189	10111101	½	Fraction one half
190	10111110	¾	Fraction three quarters
191	10111111	¿	Inverted question mark
192	11000000	À	Latin capital letter A with grave
193	11000001	Á	Latin capital letter A with acute
194	11000010	Â	Latin capital letter A with circumflex
195	11000011	Ã	Latin capital letter A with tilde
196	11000100	Ä	Latin capital letter A with diaeresis
197	11000101	Å	Latin capital letter A with ring above
198	11000110	Æ	Latin capital letter AE
199	11000111	Ç	Latin capital letter C with cedilla
200	11001000	È	Latin capital letter E with grave
201	11001001	É	Latin capital letter E with acute
202	11001010	Ê	Latin capital letter E with circumflex
203	11001011	Ë	Latin capital letter E with diaeresis
204	11001100	Ì	Latin capital letter I with grave
205	11001101	Í	Latin capital letter I with acute
206	11001110	Î	Latin capital letter I with circumflex
207	11001111	Ï	Latin capital letter I with diaeresis
208	11010000	Ð	Latin capital letter ETH
209	11010001	Ñ	Latin capital letter N with tilde

Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
210	11010010	Ò	Latin capital letter O with grave
211	11010011	Ó	Latin capital letter O with acute
212	11010100	Ô	Latin capital letter O with circumflex
213	11010101	Õ	Latin capital letter O with tilde
214	11010110	Ö	Latin capital letter O with diaeresis
215	11010111	×	Multiplication sign
216	11011000	Ø	Latin capital letter O with slash
217	11011001	Ù	Latin capital letter U with grave
218	11011010	Ú	Latin capital letter U with acute
219	11011011	Û	Latin capital letter U with circumflex
220	11011100	Ü	Latin capital letter U with diaeresis
221	11011101	Ý	Latin capital letter Y with acute
222	11011110	Þ	Latin capital letter THORN
223	11011111	ß	Latin small letter sharp s - ess-zed
224	11100000	à	Latin small letter a with grave
225	11100001	á	Latin small letter a with acute
226	11100010	â	Latin small letter a with circumflex
227	11100011	ã	Latin small letter a with tilde
228	11100100	ä	Latin small letter a with diaeresis
229	11100101	å	Latin small letter a with ring above
230	11100110	Æ	Latin small letter ae
231	11100111	Ç	Latin small letter c with cedilla
232	11101000	È	Latin small letter e with grave
233	11101001	É	Latin small letter e with acute
234	11101010	Ê	Latin small letter e with circumflex
235	11101011	Ë	Latin small letter e with diaeresis
236	11101100	Ì	Latin small letter i with grave
237	11101101	Í	Latin small letter i with acute
238	11101110	Î	Latin small letter i with circumflex
239	11101111	Ï	Latin small letter i with diaeresis
240	11110000	Ð	Latin small letter eth
241	11110001	Ñ	Latin small letter n with tilde
242	11110010	Ò	Latin small letter o with grave
243	11110011	Ó	Latin small letter o with acute
244	11110100	Ô	Latin small letter o with circumflex
245	11110101	Õ	Latin small letter o with tilde
246	11110110	Ö	Latin small letter o with diaeresis
247	11110111	÷	Division sign
248	11111000	Ø	Latin small letter o with slash

Nilai ANSI ASCII (Desimal)	Biner	Karakter	Keterangan
249	11111001	Ù	Latin small letter u with grave
250	11111010	Ú	Latin small letter u with acute
251	11111011	Û	Latin small letter u with circumflex
252	11111100	Ü	Latin small letter u with diaeresis
253	11111101	Ý	Latin small letter y with acute
254	11111110	Þ	Latin small letter thorn
255	11111111	ÿ	Latin small letter y with diaeresis

## 2.10. Metode Pengembangan Sistem Model *Prototyping*

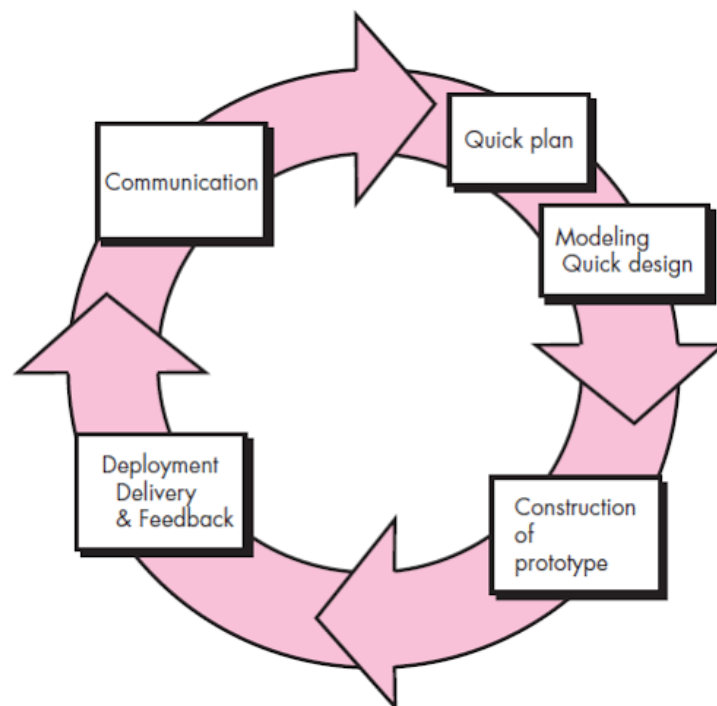
*Prototyping* adalah salah satu pendekatan dalam rekayasa perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan (Affandi, 2013).

Menurut (Pressman, 2012), Pembuatan *Prototype* (*Prototyping*). Seringkali pelanggan mendefinisikan sejumlah sasaran perangkat lunak secara umum, tetapi tidak bisa mengidentifikasi spesifikasi kebutuhan yang rinci untuk fungsi-fungsi dan fitur-fitur yang nantinya akan dimiliki perangkat lunak yang akan dikembangkan. Dalam kasus yang lain, pengembang perangkat lunak mungkin merasa tidak pasti tentang efisiensi suatu algoritma yang akan digunakan dalam pengembangan perangkat lunak, atau juga merasa tidak pasti akan kemampuan perangkat lunak untuk beradaptasi dengan sistem operasi yang akan digunakan, akan merasa tidak pasti akan bentuk interaksi manusia-komputer yang akan digunakan. Dalam kasus-kasus seperti ini dan dalam banyak situasi yang lain, paradigma pembuatan *prototype* (*prototyping*) mungkin menawarkan pendekatan yang paling baik.

Meskipun pembuatan *prototype* dapat digunakan sebagai model proses yang berdiri sendiri, pembuatan *prototype* lebih umum digunakan sebagai teknik yang dapat

diimplementasikan di dalam konteks setiap model proses perangkat lunak. Dalam hal ini, tidak terlalu peduli dengan di mana ia diterapkan, paradigma pembuatan prototype seringkali membantu tim pengembang perangkat lunak dan para stakeholder untuk memahami lebih baik apa yang akan dikembangkan saat spesifikasi kebutuhan belum jelas.

Tahapan model pengembangan *prototyping* ini digambarkan pada gambar 2.5 Model Pengembangan *Prototyping*.



Gambar 2.5 Model Pengembangan *Prototyping* (Pressman, 2012)

Metode prototipe dimulai dari tahap komunikasi. Tim pengembang perangkat lunak melakukan pertemuan dengan para *stakeholder* untuk menentukan kebutuhan perangkat lunak yang saat itu diketahui dan untuk menggambarkan area-area dimana definisi lebih jauh untuk iterasi selanjutnya. Perencanaan iterasi pembuatan prototipe dilakukan secara cepat. Setelah itu dilakukan pemodelan dalam bentuk “rancangan cepat”. Pembuatan rancangan cepat berdasarkan pada representasi aspek-aspek perangkat lunak yang akan

terlihat oleh para *end user* (misalnya rancangan antarmuka pengguna atau format tampilan). Rancangan cepat merupakan dasar untuk memulai konstruksi pembuatan prototipe.

Prototipe kemudian diserahkan kepada para *stakeholder* untuk mengevaluasi *prototype* yang telah dibuat sebelumnya dan memberikan umpan-balik yang akan digunakan untuk memperbaiki spesifikasi kebutuhan. Iterasi terjadi saat pengembang melakukan perbaikan terhadap prototipe tersebut.

### **2.11. Database MySQL**

MySQL adalah sebuah software database. Database merupakan sebuah tempat untuk meyimpan data yang jenisnya beraneka ragam. MySQL meyimpan datanya dalam bentuk tabel-tabel yang saling berhubungan (Gusmaliza, 2019) dalam buku (Edy Winarno: 120:2014).

MySQL adalah perangkat lunak untuk sistem manajemen database (database management system), karena sifatnya yang open source dan memiliki kemampuan menampung kapasitas yang sangat besar. MySQL memberikan hasil yang optimal dari sisi kecepatan dan reabilitas manajemen data, sifat dari MySQL yang open source menyebabkan MySQL berkembang secara pesat dan digunakan begitu banyak pengguna yang tidak ingin mengeluarkan dana besar untuk sebuah sistem basis data, jika menggunakan sistem basis data komersial (Gusmaliza, 2019) dalam buku (Iskandar Sulaini, 194:2007)

### **2.12. PHP (*Hypertext Preprocessor*)**

*Hypertext Preprocessor* (PHP) merupakan suatu teknologi *scripting* yang berbasis *server* (*server-side programming*) untuk membangun halaman web yang dinamis dan



*interactive*, dimana perintah-perintah diproses terlebih dahulu di *web server*. Ketika seorang *user* memasukkan alamat tertentu di *browser*, maka *browser* akan mengirimkan permintaan tersebut ke *web server* yang dimaksud dan menunggu hasilnya. Jika *file* yang diminta adalah sebuah dokumen HTML, maka *web server* akan memberikan *file* tersebut ke *web browser* apa adanya. Namun, jika *file* yang diminta adalah *file* yang mengandung *script server-side*, maka *web server* akan memproses terlebih dahulu *script* tersebut dan mengirimkan hasilnya ke *browser* (San Pratama, 2016).

Pada prinsipnya server akan bekerja apabila ada permintaan dari *client*. Dalam hal ini *client* menggunakan kode-kode PHP untuk mengirimkan permintaan ke *server*. Ketika menggunakan PHP sebagai *server-side embedded script language* maka *server* akan melakukan hal-hal sebagai berikut:

1. Membaca permintaan dari *client/browser*.
2. Mencari halaman/*page* di *server*.
3. Melakukan instruksi yang diberikan oleh PHP untuk melakukan modifikasi pada halaman/*page*.
4. Mengirim kembali halaman tersebut kepada *client* melalui *internet* atau *intranet*.

Berikut adalah contoh *syntax* dari PHP untuk menampilkan sebuah *string* di layar:

```
<?php
Echo "Hello World";
?>
```

### 2.13. Microsoft Visio

*Microsoft Visio* (atau sering disebut *Visio*) adalah sebuah *program* aplikasi komputer yang sering digunakan untuk membuat diagram, diagram alir (*flowchart*),

*brainstorm*, dan skema jaringan yang dirilis oleh *Microsoft Corporation*. Aplikasi ini menggunakan grafik vektor untuk membuat diagram-diagramnya, *Visio* menyediakan banyak fasilitas yang membantu dalam pembuatan diagram untuk menggambarkan informasi dan sistem dari penjelasan dalam bentuk teks menjadi suatu diagram dalam bentuk gambar (Irawan, 2014).

#### **2.14. Flowchart**

*Flow chart* atau diagram alir merupakan sebuah diagram dengan symbol simbol grafis yang menyatakan aliran algoritma atau proses yang menampilkan langkah-langkah yang disimbolkan dalam bentuk kotak, beserta urutannya dengan menghubungkan masing-masing langkah tersebut menggunakan tanda panah (Natsir, 2016). Diagram ini bisa memberi solusi selangkah demi selangkah untuk penyelesaian masalah yang ada di dalam proses atau algoritma tersebut.

*Flowchart* digunakan dalam merancang dan mendokumentasikan proses yang kompleks atau program. Seperti jenis lain diagram, mereka membantu memvisualisasikan apa yang terjadi dan dengan demikian membantu pengunjung untuk memahami proses, dan mungkin juga menemukan kelemahan, kemacetan, dan ketidakjelasan lain di dalamnya. Ada berbagai jenis diagram alur yang masing-masing memiliki *repertoire* kotak sendiri dan ketentuan notasinya. Dua jenis yang paling umum dari kotak di *flow chart* adalah :

1. Langkah pengolahan, biasanya disebut aktivitas, dan dilambangkan sebagai persegi panjang.
2. Keputusan, biasanya dilambangkan sebagai belah ketupat.

Sebuah *flowchart* digambarkan sebagai lintas-fungsional saat halaman dibagi

menjadi *swimlanes* yang berbeda yang menggambarkan control dari unit organisasi yang berbeda. Sebuah simbol muncul dalam jalur khusus berada dalam kontrol dari unit organisasi. Teknik ini memungkinkan penulis untuk mencari tanggung jawab untuk melakukan tindakan atau membuat keputusan dengan benar, menunjukkan tanggungjawab masing-masing unit organisasi untuk bagian yang berbeda dari sebuah proses tunggal.

## **2.15. Pengenalan UML (*Unified Modeling Language*)**

### **2.15.1. Pengertian UML**

*Unified Modeling Language* (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem (Whitten, 2007). Terdapat empat alat bantu yang digunakan dalam perancangan berorientasi objek berdasarkan UML yaitu:

1. *Use Case Diagram*
2. *Class Diagram*
3. *Sequence Diagram*
4. *Activity Diagram*

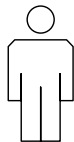
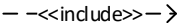


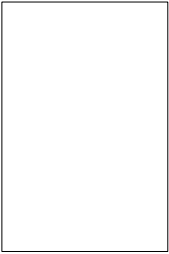
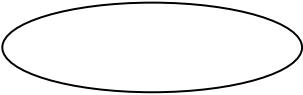
### **2.15.2. Use Case Diagram**

*Use case* adalah abstraksi dari interaksi antara sistem dan aktor. *use case* diagram memfasilitasi komunikasi di antara analis dan pengguna serta antara analis dan *client* (Hardinata, 2014).

Menurut (Whitten, 2007) *Use case diagram* merupakan pemodelan untuk

kelakuan (*behavior*) sistem informasi yang akan dibuat. Use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang ada pada *use case diagram* dapat dilihat pada tabel 2.2.





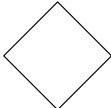
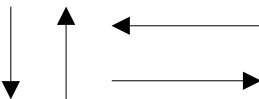
Tabel 2.3 Simbol *Use Case Diagram* (Setiawan, 2015)

NO	GAMBAR	NAMA	KETERANGAN
1		Actor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		Include	Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit.
3		Extend	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
4		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
5		System	Menspesifikasikan paket yang menampilkan system secara terbatas.
6		Use Case	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.

### 2.15.3. Activity Diagram

*Activity diagram* ini menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi (Hardinata, 2014). Simbol-simbol yang ada pada *Activity diagram* dapat dilihat pada tabel 2.3 dibawah ini:

Tabel 2.4 Simbol *Activity diagram* (Setiawan, 2015)

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antarmuka berinteraksi satu sama lain.
2		<i>Action</i>	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali.
4		<i>Activity Final Node</i>	Bagaimana objek dibentuk dan diakhiri.
5		<i>Decision</i>	Digunakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu.
6		<i>Line Connector</i>	Digunakan untuk menghubungkan satu symbol dengan symbol lainnya.

### 2.16. Pengujian Software

Pengujian *software* sangat diperlukan untuk memastikan *software* aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji *software* harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya (Shi, 2010).

### 2.17. Metode **BlackBox**

Metode uji coba *blackbox* memfokuskan pada keperluan fungsional dari *software* (Rakasiwi, 2013). Karena itu uji coba *blackbox* memungkinkan pengembang perangkat lunak untuk membuat himpunan kondisi masukan yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *blackbox* bukan merupakan alternatif dari uji coba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox*.

Uji coba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya:

- a. Fungsi-fungsi yang salah atau hilang.
- b. Kesalahan *interface*.
- c. Kesalahan dalam struktur data atau akses *database eksternal*.
- d. Kesalahan performa.
- e. Kesalahan inisialisasi dan terminasi.

Dengan mengaplikasikan uji coba *blackbox*, diharapkan dapat menghasilkan sekumpulan kasus uji yang memenuhi kriteria berikut:

- a. Kasus uji yang berkurang, jika jumlahnya lebih dari 1, maka jumlah dari uji kasus tambahan harus didesain untuk mencapai uji coba yang cukup beralasan.
- b. Kasus uji yang memberitahukan sesuatu tentang keberadaan atau tidaknya suatu jenis kesalahan, dari pada kesalahan yang terhubung hanya dengan suatu uji coba yang spesifik.

## DAFTAR PUSTAKA

- Affandi, A. (2013). Prototype Aplikasi M-Ticketing Pada Penjualan Tiket Pesawat Menggunakan Teknologi Mobile. *Skripsi*.
- Andi Juansyah. (2015). Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System ( A-GPS ) Dengan Platform Android. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA)*, 1(1), 1–8.
- Aribowo, E. (2014). Implementasi Algoritma Kriptografi Kunci Publik Elgamal Untuk Proses Enkripsi Dan Dekripsi Guna Pengamanan File Data. *Jurnal Informatika*, 1–10.
- Eko Hari Rachmawanto. (2016). Kriptografi Vernam Cipher Untuk Mencegah Pencurian Data Pada Semua Ekstensi File. *Prosiding Seminar Nasional Multi Disiplin Ilmu & Call For Papers Unisbank (Sendi\_U) Ke-2 Tahun 2016*, 46–51.
- Gusmaliza, D. (2019). Perangkat Lunak Bantu Administrasi Keuangan Sekolah Tinggi Teknologi Pagar Alam Dengan PHP Dan MySQL. *Jurnal Ilmiah Betrik*, 10(1).
- Halim, A. (2013). Implementasi Algoritma Hill Cipher Dalam Penyandian Data. IV, 115-122.
- Hardinata, N. (2014). Pembuatan Aplikasi Pocket Grammar Berbasis Android.
- Irawan, H. (2014). Perancangan Sistem Informasi Administrasi Kesiswaan pada SMP Negeri 4 Muntok dengan Berorientasi Objek. *Jurnal Sifom*.
- Jumeidi, M. (2016). Implementasi Algoritma Kriptografi Vernam Cipher Berbasis FPGA. *Jurnal Coding, Sistem Komputer UNTAN*, 04(1), 21–32.
- Natsir, M. (2016). Pengembangan Prototype Sistem Kriptografi Untuk Enkripsi Dan Dekripsi Data Office Menggunakan Metode Blowfish Dengan Bahasa Pemrograman Java. *Jurnal Format*, 6(1), 87–105.
- Nugroho, B. K. (2010). Aplikasi Enkripsi SMS pada Telepon Selular Berbasis J2ME dengan Metoda Vigenere Cipher. 1–2.
- Nurhidayat, M. (2018). Meningkatkan Keamanan Data Sms Dengan Menggunakan Metode Vigenere Cipher Berbasis Android. *Skripsi*, 21–27.
- Pressman, S. R. (2012). *Rekayasa Perangkat Lunak – Pendekatan Praktisi Edisi 7- Buku 1*. <https://doi.org/10.35793/jti.9.1.2016.14138>
- Puspita, K. (2015). Analisis Kombinasi Metode Caesar Cipher , Vernam Cipher , Dan Hill Cipher Dalam Proses Kriptografi. *Seminar Nasional Teknologi Informasi Dan*

- Multimedia* 2015, (Februari), 43–48.
- Rakasiwi, S. (2013). Perangkat Lunak Bantu Sistem Penentuan Prestasi Karyawan Pt. Telkom Divre Iv Semarang. *Ebisnis*, 6(2), 11–18.
- Renaldy, M. (2015). *Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan Menggunakan Metode Aes-128 (Advanced Encryption Standard-128) Dan Sha-1 (Secure Hash Algorithm-1)*. 128.
- San Pratama, F. (2016). Sistem Pendukung Keputusan Penerimaan Siswa Baru Menggunakan Metode Saw (Studi Kasus: Smk Ipiems Surabaya). *Jurnal Manajemen Informatika*, 5(2).
- Sari, C. A. (2016). Penyembunyian Data Untuk Seluruh Ekstensi File Menggunakan Kriptografi Vernam Cipher dan Bit Shifting. *Journal of Applied Intelligent System*, 1(3), 179–190.
- Setiawan, A. (2015). *Rancang Bangun Edugame The World Of Word Berbasis Unity 3D Dengan Implementasi Speech Recognition*. Retrieved from <http://bluewarrior.wordpress.com>
- Shi, M. (2010). Software Functional Testing from the Perspective of Business Practice. *Computer and Information Science*, 3(4), 49–52. <https://doi.org/10.5539/cis.v3n4p49>
- Sumandri. (2017). Studi Model Algoritma Kriptografi Klasik dan Modern. *Seminar Matematika Dan Pendidikan Matematika UNY*, 265–272.
- Wahyuningsih, Y. (2016). *Penyembunyian Pesan Terenkripsi Hill Cipher Pada File Audio Dengan Metode Least Significant Bit (LSB)*.
- Whitten, J. L. (2007). Systems Analysis and Design Methods. In B. Gordon (Ed.), *Human Factors in Land Use Planning and Urban Design* (7th ed.).
- Wirdasari, D. (2008). Prinsip Kerja Kriptografi dalam Mengamankan Informasi. *Saintikom*, 5(2), 174–184.
- Zelviana, A. (2012). Perancangan Aplikasi Pembelajaran Kriptografi Kunci Publik ElGamal Untuk Mahasiswa. *Jurnal Dunia Teknologi Informasi*, 1(1), 56–62.