MEMBANGUN SUPER ENKRIPSI DENGAN *VIGENERE CIPHER* DAN *BIFID CIPHER* MENGGUNAKAN PEMROGRAMAN *PYTHON* UNTUK MENGAMANKAN PESAN

SKRIPSI

OLEH
LAURA AGUSTINA
NIM.17610081

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2021

MEMBANGUN SUPER ENKRIPSI DENGAN *VIGENERE CIPHER* DAN *BIFID CIPHER* MENGGUNAKAN PEMROGRAMAN *PYTHON* UNTUK MENGAMANKAN PESAN

SKRIPSI

Diajukan Kepada
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Matematika (S.Mat)

Oleh
Laura Agustina
NIM.17610081

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2021

MEMBANGUN SUPER ENKRIPSI DENGAN *VIGENERE CIPHER* DAN *BIFID CIPHER* MENGGUNAKAN PEMROGRAMAN *PYTHON* UNTUK MENGAMANKAN PESAN

SKRIPSI

Oleh
Laura Agustina
NIM.17610081

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal, 15 November 2021

Pembimbing I,

Dr. H Imam Sujarwo, M.Pd
NIP. 19630502 198703 1 005

Pembimbing II,

Muhammad Khudzaifah, M.Si
NIDT. 19900611 20160801 1 057

Mengetahui,
Ketua Program Studi Matematika

Dr. Elly Susanti, M.Sc
NIP. 19741129 200012 2 005

MEMBANGUN SUPER ENKRIPSI DENGAN *VIGENERE CIPHER* DAN *BIFID CIPHER* MENGGUNAKAN PEMROGRAMAN *PYTHON* UNTUK MENGAMANKAN PESAN

SKRIPSI

Oleh
Laura Agustina
NIM.17610081

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Matematika (S.Mat)
Tanggal, 03 Desember 2021

Penguji Utama	: Mohammad Nafie Jauhari, M.Si	.....
Ketua Penguji	: Hisyam Fahmi, M.Kom	.....
Sekretaris Penguji	: Dr. H Imam Sujarwo, M.Pd	.....
Anggota Penguji	: Muhammad Khudzaifah, M.Si	.....

Mengetahui,
Ketua Program Studi Matematika

Dr. Elly Susanti, M.Sc
NIP. 19741129 200012 2 005

PERNYATAAN KEASLIAN TULISAN

Saya yang bertandatangan di bawah ini:

Nama : Laura Agustina

NIM : 17610081

Program Studi : Matematika

Fakultas : Sains dan Teknologi

Judul Skripsi : Membengun Super Enkripsi dengan Algoritma *Vigenere Cipher* dan *Bifid Cipher* Menggunakan Pemrograman *Python* untuk Mengamankan Pesan.

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya sendiri, bukan merupakan pengambilan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan dan pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perilaku tersebut.

Pasuruan, 15 Oktober 2021
Yang membuat pernyataan,

Laura Agustina
NIM. 17610081

MOTTO

“Hatiku tenang karena mengetahui bahwa apa yang melewatkanku tidak akan pernah menjadi takdirku, dan apa yang ditakdirkan untukku tidak akan pernah melewatkanku.”

(Umar bin Khattab)

“Ada yang tak seberuntung dirimu, tetapi rasa syukurnya melebihi dirimu.”

PERSEMBAHAN

Skripsi ini dipersembahkan oleh penulis kepada:

Orang Tua penulis, Bapak Mokhamad Agus Salim, Ibu Nurul Hidayati, yang tanpa lelah dan tidak ada henti-hentinya memberikan dukungan moral, spiritual, dan finansial, serta kasih sayangnya.

Adik Muhammad Alfan Agustian, yang memberikan doa, semangat, dan motivasi. Jangan lupa belajar dan semangat.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Puji syukur Kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan yang berjudul "*Membangun Super Enkripsi dengan Vigenere Cipher dan Bifid Cipher Menggunakan Pemrograman Python untuk Mengamankan Pesan*" dengan baik. Shalawat serta salam semoga senantiasa tercurahkan kepada junjungan kita Nabi Muhammad saw. yang telah membimbing kita dari zaman jahiliyah menuju zaman yang terang yakni agama Islam.

Dalam proses penyusunan skripsi ini, penulis banyak mendapatkan bimbingan dan arahan dari berbagai pihak. Dengan demikian, ucapan terima kasih yang sebesar-besarnya dan penghargaan yang setinggi-tingginya penulis sampaikan, terutama kepada:

1. Prof. Dr. H. M. Zainuddin, M.A, selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si, selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Elly Susanti, M.Sc, selaku ketua Program Studi Matematika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Dr. H. Imam Sujarwo, M.Pd. selaku dosen pembimbing I yang telah meluangkan waktunya untuk memberikan bimbingan, arahan, nasihat, serta berbagai pengalaman berharga kepada penulis.
5. Muhammad Khudzaifah, M.Si. selaku dosen pembimbing II yang banyak memberikan arahan, nasihat, dan berbagai ilmunya kepada penulis.
6. M. Nafie Jauhari, M.Si. sebagai dosen penguji Seminar Proposal dan Sidang Skripsi yang telah memberikan kritik dan saran yang membangun kepada penulis.
7. Hisyam Fahmi, M.Kom. sebagai dosen penguji Sidang Skripsi yang telah memberikan kritik dan saran yang bermanfaat bagi penulis.

8. Seluruh dosen Program Studi Matematika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang, beserta seluruh staff dan karyawan.
9. Orang Tua, Adik, dan seluruh keluarga yang memberikan doa, semangat, motivasi, kasih sayang, serta pengorbanan materi dan waktu yang begitu besar kepada penulis.
10. Semua teman-teman di Program Studi Matematika angkatan 2017, terutama Anis Putri Rahmadhani dan Olivia Karinina yang telah berjuang bersama dan saling mendukung satu sama lain.
11. Meydina Zulfa Atikah yang selalu mendengarkan keluh kesah penulis, serta memberikan motivasi yang tiada henti.
12. Aqilarik Nugra Rezkanintio yang membantu penulis dalam belajar bahasa pemrograman.
13. Semua teman-teman seperjuangan di bangku perkuliahan yang selalu memberikan doa, bantuan, semangat serta dukungan kepada penulis dalam menyelesaikan skripsi ini.

Penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat kekurangan. Dengan demikian, penulis sangat mengharapkan kritik dan saran yang membangun agar penulisan skripsi ini dapat lebih baik lagi. Semoga skripsi ini dapat bermanfaat dan menambah wawasan kepada para pembaca.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Pasuruan, Oktober 2021

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Agustina, Laura. 2021. **Membangun Super Enkripsi dengan *Vigenere Cipher* dan *Bifid Cipher* Menggunakan Pemrograman *Python* untuk Mengamankan Pesan**. Skripsi. Program Studi Matematika Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I)Dr. H. Imam Sujarwo, M.Pd. (II) Muhammad Khudzaifah, M.Si.

Kata Kunci: Enkripsi, Dekripsi, *Vigenere Cipher*, *Bifid Cipher*, Super Enkripsi

Masalah keamanan pesan atau suatu informasi sangatlah penting. Makna keamanan disini adalah suatu pesan bersifat rahasia yang akan disampaikan kepada penerima. Sebuah ilmu yang mempelajari tentang pengamanan kerahasiaan pesan/data dengan menggunakan sandi disebut kriptografi. Untuk lebih meningkatkan keamanan maka dilakukan penggabungan dua algoritma untuk mengamankan suatu pesan. Super enkripsi merupakan suatu konsep yang menggunakan kombinasi dari dua atau lebih teknik kriptografi substitusi dan permutasi (transposisi) untuk mendapatkan suatu algoritma yang lebih sulit untuk dipecahkan. Hal pertama yang dilakukan adalah melakukan enkripsi pesan dengan menggunakan teknik substitusi (*Cipher* Subtitusi), kemudian dienkripsi lagi dengan menggunakan teknik permutasi (*Cipher* Transposisi). Pada penelitian kali ini akan dilakukan penggabungan dua algoritma kriptografi untuk membangun super enkripsi menggunakan algoritma *Vigenere Cipher* dan *Bifid Cipher* dalam mengamankan pesan. Proses enkripsi pesan yaitu menggunakan algoritma *Vigenere Cipher* untuk proses enkripsi yang pertama, kemudian dilanjutkan menggunakan algoritma *Bifid Cipher* untuk proses enkripsi yang kedua. Untuk proses dekripsi dilakukan sebaliknya, dimulai dari urutan terbelakang proses enkripsi. Penggabungan dua algoritma ini menghasilkan keamanan pesan yang lebih terjaga.

ABSTRACT

Agustina, Laura. 2021. **Build Super Encryption by Vigenere Cipher and Bifid Cipher Using Python Programming to Secure Messages**. Thesis. Department of Mathematics, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University of Malang: (I)Dr. H. Imam Sujarwo, M.Pd. (II) Muhammad Khudzaifah, M.Si.

Keyword: *Encryption, Decryption, Vigenere Cipher, Bifid Cipher, Super Encryption.*

The issue of message security or an information is very important. A science that studies about securing the confidentiality of messages using passwords is called cryptography. To enhance security, two algorithms are combined to secure messages. Super encryption is a concept that uses a combination of two or more substitution and permutation (transposition) cryptography techniques to obtain an algorithm that is more difficult to crack. The first thing to do is to encrypt the message using a substitution technique (Cipher Substitution), then re-encrypt it using a permutation technique (Cipher Transposition). In this study, two cryptographic algorithms will be combined to build super encryption using the Vigenere Cipher and Bifid Cipher algorithms to secure messages. The message encryption process is using the Vigenere Cipher algorithm for the first encryption process, then continued using the Bifid Cipher algorithm for the second encryption process. The encryption process is done the other way around, starting from the back of the encryption process. The combination of these two algorithms results in more secure message security.

ملخص

أوجستينا ، لورا. ٢٠٢١. بناء التشفير الفائق باستخدام تشفير فيجينير و مشقوق الشفرات باستخدام برمجة فيطون لتأمين الرسائل. البحث العلمي. برنامج دراسة الرياضيات، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف: (١) الدكتور. الحاج. الإمام سوجاروا. الماجستير (٢) محمد حذيفة. الماجستير.

الكلمات الرئيسية: التشفير ، فك التشفير ، التشفير فيجينير (*Vigenere Cipher*)، التشفير بيفيض (*Bifid Cipher*)، التشفير الفائق(*Super enkripsi*).

تعتبر مسألة أمن الرسائل أو المعلومات في غاية الأهمية. معنى الأمن هنا هو رسالة سرية سيتم نقلها إلى المستلم. العلم الذي يدرس حول تأمين سرية الرسائل باستخدام كلمة مرور يسمى التشفير. لزيادة تحسين الأمان ، يتم دمج خوارزميتين لتأمين رسالة. التشفير الفائق هو مفهوم يستخدم مزيجًا من طريقتين أو أكثر من تقنيات تشفير الاستبدال والتبديل (التحويل) للحصول على خوارزمية يصعب فكها. أول شيء يجب القيام به هو تشفير الرسالة باستخدام تقنية الاستبدال/ التشفير سبتتوتيءون(*Cipher Substitusi*) .ثم يتم تشفيرها مرة أخرى باستخدام تقنية التقليب/التشفيرترانفتتيءون(*Cipher Transposisi*) في هذه الدراسة ، سيتم دمج خوارزميتين للتشفير لبناء تشفير فائق(*Super enkripsi*) باستخدام فيجينيرالسفرت(*Vigenere Cipher*) و بيفيض السفرت(*Bifid Cipher*) في تأمين الرسائل. تستخدم عملية تشفير الرسائل خوارزمية فيجينير التشفير(*Vigenere Cipher*) لعملية التشفير الأولى ، ثم تستمر في استخدام خوارزمية بيفيض التشفير(*Bifid Cipher*) لعملية التشفير الثانية. تتم عملية فك التشفير في الاتجاه المعاكس ، بدءًا من الجزء الخلفي من عملية التشفير. ينتج عن دمج هاتين الخوارزميتين في أمان أكثر أمانًا للرسائل.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pesan adalah sebuah pemberitahuan, kata, perintah, informasi, atau komunikasi dalam bentuk lisan maupun tulisan yang dibuat oleh pengirim kepada penerima pesan. Proses pengiriman pesan tentu dibutuhkan suatu media atau perantara agar pesan dapat tersampaikan dan diterima dengan baik oleh penerima pesan. Pada proses pengiriman pesan juga dibutuhkan sebuah pengamanan agar pesan dapat diterima seutuhnya dan tanpa melalui perantara pihak lain untuk menjaga kerahasiaan pesan dan menghindari berubahnya isi pesan tersebut.

Masalah keamanan pesan atau suatu informasi sangatlah penting. Makna keamanan disini adalah suatu pesan bersifat rahasia yang akan disampaikan kepada penerima. Tidak ada pihak-pihak lain yang boleh mengetahui isi informasi ataupun pesan kecuali yang memiliki kewenangan, seperti pengirim dan penerima pesan. Media pengiriman pesan adalah suatu media, sarana, atau alat yang digunakan untuk menyampaikan pesan. Semakin berkembangnya zaman, penggunaan media pengiriman pesan juga semakin berkembang. Saat ini media pengiriman pesan sudah menggunakan teknologi yang lebih canggih dan mudah dengan sistem komputer dan internet. Dalam bahasa komputer terdapat istilah algoritma yang merupakan langkah-langkah dalam membuat kode bahasa komputer yang pada umumnya merupakan proses atau aturan yang diikuti dengan perhitungan dan operasi pemecahan masalah menggunakan hitungan matematika.

Sebuah ilmu yang mempelajari tentang pengamanan kerahasiaan pesan/data dengan menggunakan sandi disebut kriptografi. Kriptografi berasal dari bahasa Yunani: dari dua suku kata *cryptos* dan *graphein*. *Cryptos* artinya rahasia, sedangkan *graphein* artinya tulisan. Jadi, arti kriptografi adalah tulisan rahasia. Sedangkan definisi kriptografi sendiri secara umum adalah ilmu dan seni yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, integritas data, serta otentifikasi data untuk menjaga kerahasiaan berita (Megantara dan Fauzi, 2019). Konsep dasar pada kriptografi disini meliputi enkripsi, dekripsi, dan *key*/kunci. Enkripsi merupakan sebuah algoritma yang memiliki 2 masukan yaitu teks asli dan kunci rahasia. Algoritma enkripsi melakukan transformasi terhadap teks asli sehingga menghasilkan teks sandi. Dekripsi merupakan sebuah algoritma yang memiliki 2 masukan yaitu teks sandi dan kunci rahasia. Algoritma dekripsi memulihkan kembali teks sandi menjadi teks asli bila kunci rahasia yang digunakan algoritma dekripsi sama dengan kunci rahasia yang digunakan oleh algoritma enkripsi. *Key*/kunci merupakan masukan bagi algoritma enkripsi yang merupakan nilai bebas terhadap teks asli dan menentukan hasil keluaran algoritma enkripsi, juga dilakukan untuk menentukan hasil keluaran algoritma dekripsi (Ariyus, 2006).

Ada berbagai macam jenis algoritma kriptografi yang telah ditemukan, seperti algoritma *Vigenere Cipher* dan algoritma *Bifid Cipher* yang akan dijelaskan pada penelitian kali ini. *Vigenere Cipher* merupakan algoritma kriptografi yang menggunakan teknik substitusi dimana proses enkripsi dan dekripsi dilakukan dengan menggeser posisi huruf sejauh kunci yang telah ditentukan. Kunci pada *Vigenere Cipher* berbentuk karakter huruf dengan penomoran karakter yang sudah

ditetapkan. *Bifid Cipher* menupakan algoritma kriptografi yang menggunakan teknik permutasi/transposisi dimana proses enkripsi dan dekripsi dilakukan dengan memindah koordinat posisi huruf satu dengan yang lainnya pada susunan huruf yang terdapat dalam tabel, dengan urutan penyusunan diawali dengan kunci kemudian dilanjutkan dengan huruf yang tersisa tanpa melakukan perulangan. Kunci pada *Bifid Cipher* berbentuk karakter huruf dengan menghapus atau menghilangkan kata berulang. Digunakan algoritma ini karena kedua algoritma tersebut memiliki karakter kunci yang sama, yaitu menggunakan karakter kata. Sehingga untuk melakukan enkripsi dan dekripsi hanya memerlukan satu kunci saja untuk dua algoritma gabungan.

Pada penelitian terdahulu telah dilakukan penerapan kriptografi dengan menggunakan berbagai macam algoritma, akan tetapi yang dilakukan tersebut hanyalah menggunakan satu algoritma saja. Untuk lebih meningkatkan keamanan maka dilakukan penggabungan dua algoritma untuk mengamankan suatu pesan. Super enkripsi yang merupakan suatu konsep kombinasi dari dua atau lebih algoritma kriptografi yaitu teknik substitusi dan permutasi *cipher* untuk mendapatkan suatu algoritma yang sulit dipecahkan (Ariyus, 2006). Algoritma super enkripsi dapat meningkatkan keamanan pesan karena pada proses enkripsi dilakukan lebih dari satu kali, setelah melakukan enkripsi pertama akan dilakukan enkripsi selanjutnya dengan algoritma kriptografi yang berbeda untuk mendapatkan teks sandi, begitu seterusnya jika ingin mendapatkan teks sandi yang sulit untuk dipecahkan. Maka keamanan pesan akan semakin terjaga.

Untuk implementasi proses enkripsi dan dekripsi dapat dilakukan dengan pemrograman *Python*. Karena *Python* merupakan bahasa pemrograman yang

mudah dipelajari serta memiliki kode-kode pemrograman yang lengkap, jelas, dan mudah dipahami. *Python* dapat digunakan untuk pembuatan aplikasi berbasis kecerdasan buatan (*artificial intelligence*). Distribusi aplikasi yang dibuat menggunakan *Python* bersifat *multi-platform*, yang artinya dapat digunakan dan dijalankan pada berbagai platform sistem operasi. Beberapa platform yang mendukung program *Python* meliputi: Linux/Unix, Windows, Mac OS X, Java Virtual Machine, OS/2, Amiga, Palm, dan Symbian (untuk produk-produk Nokia) (Enterprise, 2019).

Ilmu pengetahuan haruslah bermanfaat penggunaannya dan tidak menyebabkan kerugian bagi orang lain, sebagaimana yang sudah dijelaskan dalam Al-Quran surat Yunus ayat 24 :

إِنَّمَا مَثَلُ الْحَيٰوةِ الدُّنْيَا كَمَآءٍ أَنْزَلْنٰهُ مِنَ السَّمَآءِ فَاخْتَلَطَ بِهِ نَبَاتُ الأَرْضِ مِمَّا يَأْكُلُ النَّاسُ وَالأَنْعٰمُ حَتّٰٓى إِذَآ أَخَذَتِ الأَرْضُ زُخْرُفَهَا وازَّيَّنَتْ وَظَنَّ أَهْلُهَآ أَنَّهُمْ قٰدِرُوْنَ عَلَيْهَآ أَتٰهَآ أَمْرُنَا لَيْلًا أَوْ نَهَارًا فَجَعَلْنٰهَا حَصِيْدًا كَأَنْ لَّمْ تَغْنَ بِالأَمْسِ كَذٰلِكَ نُفَصِّلُ الْأٰيٰتِ لِقَوْمٍ يَتَفَكَّرُوْنَ (24)

"Sesungguhnya perumpamaan kehidupan duniawi itu adalah seperti air (hujan) yang Kami Turunkan dari langit, lalu tumbuhlah tanaman-tanaman bumi dengan subur (karena air itu), diantaranya ada yang dimakan manusia dan binatang ternak. Hingga apabila bumi itu telah sempurna keindahannya, dan berhias, dan pemiliknya mengira bahwa mereka pasti menguasainya (memetik hasilnya), datanglah kepadanya azab Kami pada waktu malam atau siang, lalu Kami Jadikan (tanaman-tanamannya) seperti tanaman yang sudah disabit, seakan-akan belum pernah tumbuh kemarin. Demikianlah Kami Menjelaskan tanda-tanda kekuasaan (Kami) kepada orang-orang yang berfikir". (Q.S Yunus/10 : 24).

Berdasarkan Al-Quran surat Yunus ayat 24 tersebut menjelaskan bahwa sebuah kehidupan di dunia yang diumpamakan sebagai air hujan. Segala sesuatu yang diberikan oleh Allah di dunia pastilah membawa manfaat bagi makhuknya, tetapi hal tersebut berlaku bagi makhluk yang menggunakannya dengan bijaksana. Apabila pemberian tersebut digunakan dengan serakah maka Allah akan

memberikan azab kepada mereka. Begitupun juga ilmu pengetahuan, Allah telah memberikan ilmu kepada manusia haruslah digunakan dan dikembangkan dengan sebaik-baiknya agar dapat bermanfaat dan tidak merugikan bagi orang lain.

Pada zaman sekarang ini, pengembangan suatu ilmu pengetahuan sangat penting untuk dilakukan, terutama pada pengamanan suatu informasi atau pesan. Oleh karena itu dilakukan suatu penelitian dengan penggabungan sistem kriptografi agar menjadi lebih aman. Peningkatan sistem keamanan pesan ini dilakukan agar pengiriman informasi/pesan dapat tersampaikan kepada penerima pesan tanpa diambil alih oleh orang lain. Pada penelitian kali ini akan dilakukan penggabungan dua algoritma kriptografi untuk membangun super enkripsi menggunakan algoritma *Vigenere Cipher* dan *Bifid Cipher* dalam mengamankan pesan, serta implementasi menggunakan bahasa pemrograman *Python*.

1.2 Rumusan Masalah

Adapun rumusan masalah yang akan dijelaskan pada penelitian ini adalah sebagai berikut:

1. Apakah penggabungan dua algoritma *Vigenere Cipher* dan *Bifid Cipher* membentuk algoritma super enkripsi?
2. Bagaimana implementasi algoritma super enkripsi menggunakan pemrograman *Python*?

1.3 Tujuan Penelitian

Adapun tujuan penelitian yang akan dijelaskan pada penelitian ini adalah sebagai berikut:

1. Untuk mengetahui penggabungan dua algoritma *Vigenere Cipher* dan *Bifid Cipher* membentuk algoritma super enkripsi.
2. Untuk mengetahui implementasi algoritma super enkripsi menggunakan pemrograman *Python*.

1.4 Manfaat Penelitian

Adapun manfaat penelitian pada penelitian ini adalah sebagai berikut:

1. Bagi Penulis

   Penelitian ini dapat digunakan sebagai tambahan informasi dan wawasan pengetahuan di bidang kriptografi tentang penggabungan dua algoritma untuk mendapatkan algoritma super enkripsi.

2. Bagi Lembaga

   Hasil dari penelitian ini dapat digunakan sebagai bahan referensi dan tambahan wawasan di program studi Matematika untuk mata kuliah kriptografi di bidang aljabar.

3. Bagi Pembaca

   Sebagai informasi atau tambahan wawasan bagi pembaca yang akan melakukan penelitian tentang algoritma super enkripsi.

4. Bagi Pengembang Ilmu

   Hasil penggabungan algoritma ini dapat dilakukan sebagai bahan pembuatan aplikasi super enkripsi bagi bidang komputasi.

1.5 Batasan Masalah

Untuk membatasi ruang lingkup pembahasan, maka batasan masalah penelitian ini adalah dengan menggunakan 26 karakter huruf yaitu huruf alfabet dengan penulisan kapital dalam proses enkripsi dan dekripsi pesan.

1.6 Metode Penelitian

Dalam penelitian ini akan membahas tentang super enkripsi, yaitu penggabungan dari *cipher* substitusi dan *cipher* transposisi. *Cipher* substitusi disini menggunakan algoritma *Vigenere Cipher* dan *cipher* transposisi menggunakan algoritma *Bifid Cipher*. Dilakukan penggabungan dua algoritma ini bertujuan untuk meningkatkan keamanan pesan.

Adapun langkah-langkah yang harus dilakukan untuk menyelesaikan super enkripsi tersebut adalah:

Proses Enkripsi

1. Membuat *plaintext*.
2. Menentukan kunci.
3. Melakukan enkripsi menggunakan algoritma *Vigenere Cipher*.
4. Memperoleh *ciphertext* dari hasil enkripsi menggunakan algoritma *Vigenere Cipher*.
5. *Ciphertext* tersebut selanjutnya menjadi *plaintext* untuk algoritma *Bifid Cipher*.
6. Melakukan enkripsi menggunakan algoritma *Bifid Cipher*.
7. Memperoleh *ciphertext* akhir dari hasil enkripsi menggunakan gabungan algoritma *Vigenere Cipher* dan *Bifid Cipher*.

Proses Dekripsi

1. Menentukan *ciphertext*.
2. Mendapatkan kunci.
3. Melakukan dekripsi menggunakan algoritma *Bifid Cipher*.
4. Memperoleh *plaintext* dari hasil dekripsi menggunakan algoritma *Bifid Cipher*.
5. *Plaintext* tersebut selanjutnya menjadi *ciphertext* untuk algoritma *Vigenere Cipher*.
6. Melakukan dekripsi menggunakan algoritma *Vigenere Cipher*.
7. Memperoleh *plaintext* akhir dari hasil dekripsi menggunakan gabungan algoritma *Vigenere Cipher* dan *Bifid Cipher*.

1.7 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini terbagi menjadi empat bab dan terdapat sub-bab di setiap babnya. Sistematika penulisan ini dibuat sebagai acuan bagi penulis agar lebih sistematis dan mudah dipahami. Adapun sistematika penulisan dalam penelitian ini adalah sebagai berikut:

BAB I Pendahuluan

Pada Bab Pendahuluan terdiri atas latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, metode penelitian, dan sistematika penulisan.

BAB II Kajian Pustaka

Kajian Pustaka berisi tentang teori-teori yang berkaitan dengan permasalahan yang akan diteliti. Teori tersebut meliputi keterbagian,

aritmatika modulo, kongruensi, fungsi , matriks, kriptografi, algoritma kriptografi, kriptografi klasik yang terdiri dari sandi substitusi dan sandi transposisi, super enkripsi, bahasa pemrograman *Python*, serta kajian keagamaan.

BAB III Pembahasan

Bab Pembahasan ini berisikan hasil dari penelitian tentang penggabungan algoritma kriptografi untuk membangun algoritma super enkripsi dengan dua algoritma, serta implementasi algoritma super enkripsi menggunakan pemrograman *Python*.

BAB IV Penutup

Bab ini berisi kesimpulan dari hasil penggabungan algoritma kriptografi untuk membangun algoritma super enkripsi dengan dua algoritma, juga saran untuk penelitian selanjutnya.

BAB II

KAJIAN PUSTAKA

2.1 Keterbagian

Suatu bilangan bulat b habis dibagi oleh suatu bilangan bulat $a \neq 0$, jika $\exists k \in \mathbb{Z} \ni b = ak$.

Dalam hal keterbagian, notasi yang umum digunakan adalah $a|b$ dibaca dengan "*a membagi b*" atau "*a adalah pembagi dari b ; a adalah faktor dari b ; b adalah kelipatan dari a*". Pada penjelasan selanjutnya tentang keterbagian $a|b$ dijelaskan $a \neq 0$ secara otomatis meskipun tidak dituliskan (Irawan, 2014).

2.2 Aritmatika Modulo

Misalkan a dan m adalah bilangan bulat dengan $m > 0$. Operasi $a \ mod \ m$ dibaca *"a modulo m",* memberikan sisa r jika a dibagi m.

Dalam aritmatika modulo, dapat dinotasikan sebagai:

$$a \ mod \ m = r \tag{2.1}$$

sedemikian sehingga $a = m \cdot q + r$, dengan $0 \leq r < m$. Bilangan m disini disebut $modulus$ atau $modulo$. Hasil aritmatika modulo m terletak dalam himpunan $\{0, 1, 2, \dots, m-1\}$ (Munir, 2019).

2.3 Kongruensi

Misalkan a, b, dan m adalah bilangan bulat dengan $m > 0$. Bilangan a disebut kongruen dengan b modulo m jika $m|(a-b)$ dan ditulis

$$a \equiv b(\ mod \ m) \tag{2.2}$$

Jika m membagi $a-b$, dengan kata lain $a-b=t\cdot m$, untuk suatu t elemen bilangan bulat.

Sehingga dari definisi diatas, $a\equiv b\,(mod\,m)$ juga dapat dinyatakan sebagai

$$a-b=t\cdot m \tag{2.3}$$

yaitu selisih $a-b$ adalah kelipatan dari m. Atau juga dapat dinyatakan

$$a=t\cdot m+b \tag{2.4}$$

yaitu a sama dengan b ditambah dengan kelipatan m (Irawan, 2014).

Teorema 2.1

Misalkan a, b, dan c merupakan bilangan bulat, dan m adalah bilangan asli. Maka berlaku sifat-sifat kongruensi:

1. Sifat Refleksi

   $$a\equiv a\,(mod\,m) \tag{2.5}$$

2. Sifat Simetris

   Jika

   $$a\equiv b\,(mod\,m) \tag{2.6}$$

   maka $b\equiv a(mod\,m)$ ekuivalen dengan $a-b\equiv 0(mod\,m)$.

3. Sifat Transitif

   Jika

   $$a\equiv b\,(mod\,m)\;dan\;b\equiv c\,(mod\,m) \tag{2.7}$$

   maka $a\equiv c(mod\,m)$

Bukti:

1. Apabila $m\neq 0$

   Maka $m|0$

   Dapat ditulis $m|a-a$

Berdasarkan definisi kongruensi, maka terbukti $a \equiv a(mod\ m)$, untuk a adalah bilangan bulat dan $m \neq 0$

2. $a \equiv b(mod\ m)$

$m\|a-b$	Definisi Kongruensi
$a-b=tm$	Definisi Kongruensi
$-(a-b)=-(tm)$	Kedua ruas dikalikan negatif
$-a+b=-tm$	Komutatif
$b-a=(-t)m$	Definisi Kongruensi

Berdasarkan definisi kongruensi,

$$b \equiv a(mod\ m)$$

Apabila $a \equiv b(mod\ m)$ menjadi $m|a-b$ yang berarti $a-b=tm$, dengan t adalah bilangan bulat.

Dan apabila $b \equiv a(mod\ m)$ menjadi $m|b-a$ yang berarti $b-a=(-t)m$, dengan t adalah bilangan bulat.

Dan untuk setiap $(a-b)-0=tm$, maka dapat ditulis $a-b \equiv 0(mod\ m)$.

Sehingga terbukti pernyataan tersebut ekuivalen.

3. $a \equiv b(mod\ m)$

$m\|a-b$	Definisi Kongruensi
$b \equiv c(mod\ m)$	
$m\|b-c$	Definisi Kongruensi

Menurut definisi kongruensi, terdapat bilangan bulat t_1 dan t_2, sehingga:

$m|a-b$ dinyatakan sebagai $a-b=t_1 \cdot m$

$m|b-c$ dinyatakan sebagai $b-c=t_2 \cdot m$

Apabila kedua persamaan diatas dijumlahkan, diperoleh:

$$(a-b)+(b-c\,)=(t_1\cdot m)+(t_2\cdot m)$$

$$a-b+b-c=t_1\cdot m+t_2\cdot m$$

$$a-c=(t_1+t_2)m$$

Berdasarkan definisi kongruensi, dengan t_1, t_2 adalah bilangan bulat

Maka sifat kongruensi tersebut terbukti dan dapat ditulis menjadi:

$$a\equiv c(mod\ m)$$

(Irawan, 2014).

2.4 Fungsi

Jika A dan B adalah himpunan tak kosong, maka Hasil Kali Kartesian $A\times B$ dari A dan B merupakan himpunan semua pasangan terurut (a,b) dengan $a\in A$ dan $b\in B$. Dapat ditulis,

$$A\times B=\{(a,b):a\in A,b\in B\} \tag{2.8}$$

Misalkan A dan B merupakan suatu himpunan. Maka fungsi dari A ke B adalah himpunan f dari pasangan terurut $A\times B$ sedemikian sehingga untuk setiap $a\in A$ terdapat $b\in B$ dengan $(a,b)\in f$. Dengan kata lain jika $(a,b)\in f$ dan $(a,b')\in f$, maka $b=b'$.

Himpunan A dari elemen pertama dari fungsi f disebut *domain* dari f dan dilambangkan dengan $D(f)$. Himpunan dari semua elemen kedua dalam f disebut *range* f dan dilambangkan dengan $R(f)$. Meskipun $D(f)=A$, kita hanya memiliki $R(f)\subseteq B$ (Bartle dan Sherbert, 2011).

2.4.1 Jenis-Jenis Fungsi

Misalkan $f:A\to B$ merupakan fungsi dari A ke B.

1) Fungsi Injektif

   Fungsi f dikatakan injektif atau disebut juga fungsi satu-satu jika untuk setiap $x_1 \neq x_2$, maka $f(x_1) \neq f(x_2)$. Jika f adalah fungsi injektif, kita juga dapat mengatakan f adalah injeksi.

   Bukti:

   Untuk membuktikan bahwa fungsi f adalah injektif, kita harus menetapkan bahwa:

   Untuk semua x_1, x_2 di A, jika $f(x_1) = f(x_2)$, maka $x_1 = x_2$.

   Untuk melakukan ini kita asumsikan bahwa $f(x_1) = f(x_2)$ dan tunjukkan bahwa $x_1 = x_2$.

2) Fungsi Surjektif

   Fungsi f dikatakan surjektif atau disebut juga fungsi kepada jika $f(A) = B$; yaitu jika daerah hasil $R(f) = B$. Jika f adalah fungsi surjektif, kita juga dapat mengatakan f adalah surjeksi.

   Bukti:

   Untuk membuktikan bahwa suatu fungsi f adalah surjektif, kita harus menunjukkan bahwa untuk sembarang $b \in B$ terdapat paling sedikit satu $x \in A$ sedemikian hingga $f(x) = B$.

3) Fungsi Bijektif

   Jika f memiliki kedua sifat diatas yaitu injektif dan surjektif, maka f dikatakan bijektif. Jika f adalah fungsi bijektif, kita juga dapat mengatakan f adalah bijeksi

(Bartle dan Sherbert, 2011).

2.4.2 Invers Fungsi

Jika f merupakan fungsi dari A ke B, maka f merupakan himpunan bagian khusus dari $A \times B$. Himpunan pasangan terurut di $B \times A$ diperoleh dengan menukar anggota pasangan terurut di f yang umumnya bukan fungsi. Namun apabila f adalah bijeksi, maka pemetaan menghasilkan suatu fungsi yang disebut fungsi invers dari f (Bartle dan Sherbert, 2011).

Jika $f : A \to B$ adalah bijeksi dari A ke B, maka

$$g = \{(b, a) \in B \times A : (a, b) \in f\} \tag{2.9}$$

2.4.3 Komposisi Fungsi

Sering terjadi kita ingin mengkomposisikan dua fungsi f dan g, terlebih dahulu dapatkan $f(x)$ dan kemudian terapkan fungsi g untuk mendapatkan $g\big(f(x)\big)$; namun, hal ini terjadi hanya jika $f(x)$ termasuk dalam *domain* g. Agar komposisi fungsi tersebut dapat dilakukan, maka untuk semua $f(x)$ kita asumsikan bahwa daerah hasil f berada dalam daerah asal g (Bartle dan Sherbert, 2011).

Jika $f : A \to B$ dan $g : B \to C$, dan jika $R(f) \subseteq D(g) = B$, maka fungsi komposisi $g \circ f$ adalah fungsi dari A ke C yang didefinisikan dengan

$$(g \circ f\,)(x) = g\big(f(x)\big) \text{ untuk semua } x \in A \tag{2.10}$$

Teorema 2.2

Misalkan $f : A \to B$ dan $g : B \to C$ adalah fungsi dan misalkan H merupakan himpunan bagian dari C, maka

$$(g \circ f)^{-1}(H) = f^{-1}\big(g^{-1}(H)\big) \tag{2.11}$$

Perhatikan penulisan pembalikan urutan fungsi.

(Bartle dan Sherbert, 2011).

Bukti:

Akan dibuktikan $(g \circ f)^{-1}(H) = f^{-1} \circ \left(g^{-1}(H)\right)$

$(g \circ f)^{-1} = f^{-1} \circ g^{-1}$

$(g \circ f)^{-1} = \left\{(c,a) \in C \times A \mid (a,c) \in g \circ f\right\}$

$= \left\{(c,a) \in C \times A \mid (\exists b \in B).(a,b) \in f \wedge (b,c) \in g\right\}$

$= \left\{(c,a) \in C \times A \mid (\exists b \in B).(b,a) \in f^{-1} \wedge (c,b) \in g^{-1}\right\}$

$= f^{-1} \circ g^{-1}$

Sehingga dari pembuktian Teorema 2.2 diatas benar bahwa

$$(g \circ f)^{-1}(H) = f^{-1}\left(g^{-1}(H)\right).$$

Teorema 2.3

Misalkan fungsi $f : A \to B$, $g : B \to C$, dan $h : C \to D$, maka

1. $h \circ (g \circ f) = (h \circ g) \circ f$, merupakan sifat asosiatif.
2. Jika f dan g adalah fungsi satu-satu (fungsi injektif), maka $g \circ f$ adalah fungsi injektif.
3. Jika f dan g adalah fungsi kepada (fungsi surjektif), maka $g \circ f$ adalah fungsi surjektif.
4. Jika f adalah fungsi bijektif, maka terdapat fungsi f^{-1} dari B ke A sedemikian sehingga $(f^{-1} \circ f)(a) = a$ untuk semua a di A dan $(f \circ f^{-1})(b) = b$ untuk semua b di B.

(Gallian, 2013).

Bukti:

1. $h \circ (g \circ f) = \{(a,d) \in A \times D \mid (\exists c \in C).\, (a,c) \in g \circ f \wedge (c,d) \in h\}$

$= \{(a,d) \in A \times D \mid (\exists c \in C).\, ((\exists b \in B)(a,b) \in f \wedge (b,c) \in g) \wedge (c,d) \in h\}$

$= \{(a,d) \in A \times D \mid (\exists c \in C)(\exists b \in B)((a,b) \in f \wedge (b,c) \in g \wedge (c,d) \in h)\}$

$= \{(a,d) \in A \times D \mid (\exists b \in B)(\exists c \in C)((a,b) \in f \wedge (b,c) \in g \wedge (c,d) \in h)\}$

$= \{(a,d) \in A \times D \mid (\exists b \in B).\, (a,b) \in f \wedge ((\exists c \in C)(b,c) \in g \wedge (c,d) \in h)\}$

$= \{(a,d) \in A \times D \mid (\exists b \in B).\, (a,b) \in f \wedge (b,d) \in h \circ g\}$

$= (h \circ g) \circ f$

Maka terbukti bahwa $h \circ (g \circ f) = (h \circ g) \circ f$ memiliki sifat asosiatif.

2. Karena g adalah fungsi satu-satu,

dapat ditulis $g(f(a_1)) = g(f(a_2))$,

dan menyatakan bahwa $f(a_1) = f(a_2)$.

Karena f adalah fungsi satu-satu,

sehingga $a_1 = a_2$.

Maka terbukti bahwa $g \circ f$ adalah fungsi injektif.

3. Misalkan $c \in C$,

   terdapat $b \in B$,

   sedemikian sehingga $g(b) = c$,

   dan $a \in A$,

   sedemikian sehingga $f(a) = b$,

   sehingga $(g \circ f)(a) = g\big(f(a)\big) = g(b) = c$.

   Maka terbukti bahwa $g \circ f$ adalah fungsi surjektif.

4. Karena f adalah fungsi bijektif (fungsi satu-satu dan fungsi kepada),

   dapat didefinisikan $f^{-1}(x) = y$,

   jika dan hanya jika $f(y) = x$,

   sehingga $f^{-1}\big(f(a)\big) = a$ dan $f\big(f^{-1}(b)\big) = b$.

   Maka untuk komposisi fungsi yang memiliki sifat bijektif, hasil dari invers fungsi adalah nilai awal dari anggota daerah himpunan sebelum pemetaan dilakukan. Begitu pula sebaliknya, pemetaan dari invers fungsi menghasilkan anggota himpunan yang sama.

2.5 Matriks

Matriks adalah susunan bilangan atau skalar yang disusun dalam bentuk bujur sangkar atau empat persegi (dalam baris dan kolom) yang dibatasi dengan tanda kurung. Bilangan-bilangan atau skalar-skalar tersebut dinamakan ***entri*** atau ***elemen*** dalam matriks (Kusumawati, 2014). Sebuah matriks dinotasikan dengan huruf kapital seperti $A, B, C,$ dsb. dan untuk elemen matriks dinotasikan dalam bentuk a_{ij} pada baris ke-i dan kolom ke-j. Sehingga untuk A matriks berordo $m \times n$, yaitu matriks A dengan m baris dan n kolom. Atau biasanya matriks secara

sederhana dapat ditulis sebagai $A = [a_{ij}]$, maka bentuk umum dari matriks $A_{m\times n}$ adalah:

$$A_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{2.12}$$

Baris-baris dari matriks A di atas adalah m deret horizontal yang terdiri dari skalar-skalar:

$$[a_{11}, a_{12}, \dots, a_{1n}], [a_{21}, a_{22}, \dots, a_{2n}], \dots, [a_{m1}, a_{m2}, \dots, a_{mn}]$$

dan kolom-kolom dari matriks A di atas adalah n deret vertikal yang terdiri dari skalar-skalar:

$$\begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix}, \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix}, \dots, \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix}$$

(Kusumawati, 2014).

Dua buah matriks A dan B dikatakan sama jika dan hanya jika ordo matriks A sama dengan ordo matriks B, dan $a_{ij} = b_{ij}$, untuk setiap i dan j.

2.5.1 Operasi Matriks

a) Penjumlahan Matriks

Jika $A = [a_{ij}]$ dan $B = [b_{ij}]$ adalah dua matriks yang berukuran sama, misalnya $m \times n$, maka jumlah matriks A dan matriks B adalah matriks yang diperoleh dengan menjumlahkan unsur-unsur yang bersesuaian pada kedua matriks tersebut secara bersamaan. Matriks dengan ukuran berbeda tidak dapat dijumlahkan (Kusumawati, 2014).

$$A_{m\times n} + B_{m\times n} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix} \quad (2.13)$$

b) Pengurangan/Selisih Matriks

Jika $A = [a_{ij}]$ dan $B = [b_{ij}]$ adalah dua buah matriks yang berukuran sama, misalnya $m \times n$, maka matriks $A - B$ didefinisikan sebagai jumlah dari $A + (-B) = A + (-1)B$, atau unsur-unsur yang bersesuaian pada kedua matriks tersebut dikurangkan secara bersamaan. Matriks dengan ukuran berbeda tidak dapat dikurangkan (Kusumawati, 2014).

$$A_{m\times n} - B_{m\times n} = \begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \cdots & a_{1n}-b_{1n} \\ a_{21}-b_{21} & a_{22}-b_{22} & \cdots & a_{2n}-b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}-b_{m1} & a_{m2}-b_{m2} & \cdots & a_{mn}-b_{mn} \end{bmatrix} \quad (2.14)$$

c) Penggandaan Matriks dengan Bilangan (Skalar)

Jika A adalah matriks dan k adalah skalar, maka hasil kali $k.A$ adalah matriks yang diperoleh dengan mengalikan setiap entri dari A dengan k (Kusumawati, 2014).

$$k.A = \begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix} \quad (2.15)$$

d) Penggandaan Vektor Kolom (n Kolom) dengan Vektor Baris (n Baris)

Jika A dan B adalah sebarang dua matriks, hasil kali AB dari matriks baris $A = [a_i]$ dapat ditulis sebagai $A = [a_1, a_2, \dots, a_n]$ dan matriks kolom $B = [b_j]$ atau ditulis sebagai $B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ yang didefinisikan sebagai matriks skalar (atau

1×1). Diperoleh dengan mengalikan entri yang sesuai dan kemudian menjumlahkannya (Kusumawati, 2014).

$$AB = [a_1, a_2, \dots, a_n] \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \sum_{k=1}^{n} a_k b_k \tag{2.16}$$

e) Penggandaan Matriks

Jika $A = [a_{ik}]$ dan $B = [b_{kj}]$ matriks, jumlah kolom matriks A sama dengan jumlah baris matriks B; dengan asumsi bahwa A adalah matriks berukuran $m \times r$ dan B adalah matriks berukuran $r \times n$, maka hasil kali dari AB adalah matriks berukuran $m \times n$ dan entri (ij) diperoleh dengan cara berikut:

Untuk mencari entri pada baris ke-i dan kolom ke-j dari AB, pilih baris ke-i dari matriks A dan kolom ke-j dari matriks B. Kalikan entri yang sesuai pada baris ke-i dari matriks A dan kolom ke-j dari matriks B, dan kemudian jumlahkan hasil yang diperoleh dengan mengalikan entri tersebut.

$$\begin{bmatrix} a_{11} & \cdots & a_{1r} \\ \cdot & \cdots & \cdot \\ a_{i1} & \cdots & a_{ir} \\ \cdot & \cdots & \cdot \\ a_{m1} & \dots & a_{mr} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1r} & \cdots & b_{1n} \\ \cdot & \cdots & \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot & \cdots & \cdot \\ b_{r1} & \dots & b_{rj} & \dots & b_{rn} \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ c_{m1} & \dots & c_{mn} \end{bmatrix} \tag{2.17}$$

di mana:

$$c_{ij} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{ir} b_{rj} = \sum_{k=1}^{r} a_{ik}\, b_{kj}.$$

Hasil kali AB tidak dapat didefinisikan apabila A adalah matriks dengan ukuran $m \times r$ dan B adalah matriks dengan ukuran $s \times n$, dimana $r \neq s$ (Kusumawati, 2014).

2.5.2 Jenis-Jenis Matriks

1) Matriks Persegi (*Square Matrix of order n*)

Sebuah matriks dengan jumlah baris (n baris) sama dengan jumlah kolom (n kolom), entri $a_{11}, a_{22}, \dots, a_{nn}$, dikatakan berada pada diagonal utama. Jumlah semua entri diagonal utama disebut *trace* (disingkat Tr) (Kusumawati, 2014).

Contoh

$$A = \begin{bmatrix} 3 & 1 & 4 \\ 0 & 9 & -2 \\ 5 & 7 & 15 \end{bmatrix}$$

maka : $Tr(A) = 3 + 9 + 15 = 27.$

2) Matriks Nol (*Zero Matrix*)

Sebuah matriks dengan semua elemennya sama dengan 0, biasanya dilambangkan dengan O (Kusumawati, 2014).

Contoh

$$O_{2\times 2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};\ O_{2\times 3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

3) Matriks Segitiga Atas (*Upper Triangular*)

Sebuah matriks persegi dimana entrinya $a_{ij} = 0$ untuk $i > j$ atau entri di bawah diagonal utama adalah nol (Kusumawati, 2014).

Contoh

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix} = \begin{bmatrix} 2 & 8 & 1 \\ 0 & 6 & 7 \\ 0 & 0 & 9 \end{bmatrix}.$$

4) Matriks Segitiga Bawah (*Lower Triangular*)

Sebuah matriks persegi yang entrinya $a_{ij} = 0$ untuk $i < j$ atau entri di atas diagonal utama adalah nol (Kusumawati, 2014).

Contoh

$$B = \begin{bmatrix} b_{11} & 0 & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 8 & 10 & 0 \\ 1 & 7 & 8 \end{bmatrix}.$$

5) Matriks Diagonal

Sebuah matriks persegi yang semua entrinya nol kecuali entri pada diagonal utama (bilangan bulat), biasanya dinyatakan sebagai matriks *D* (Kusumawati, 2014).

Contoh

$$D = \begin{bmatrix} -5 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 2 \end{bmatrix}.$$

6) Matriks Satuan (*Identity Matrix*)

Matriks kuadrat dengan elemennya adalah 1 pada diagonal utama dan elemen 0 di luar diagonal utama, dilambangn dengan I_n (Kusumawati, 2014).

Contoh

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix};\ I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

7) Matriks Skalar

Sebuah matriks diagonal, dimana $a_{11} = a_{22} = \cdots = a_{nn} = k$ (k skalar = konstan) atau matriks dengan diagonal utama adalah elemen yang sama, tetapi tidak 1 (Kusumawati, 2014).

Contoh

$$K = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}.$$

8) Matriks Tranpose

Jika A sebarang matriks $m \times n$, maka transpose A dinotasikan A^t dan didefinisikan oleh matriks $n \times m$, dimana kolom pertama adalah baris pertama matriks A, dan kolom kedua adalah baris kedua matriks A, dan seterusnya (Kusumawati, 2014).

Contoh

$$A = \begin{bmatrix} 5 & -2 & 1 \\ 3 & 6 & 4 \end{bmatrix}, \text{ maka } A^t = \begin{bmatrix} 5 & 3 \\ -2 & 6 \\ 1 & 4 \end{bmatrix}.$$

9) Matriks Simetris

Matriks persegi yang hasil transposenya sama dengan matriks aslinya $(A^t = A)$, atau matriks persegi $A = (a_{ij})$ dikatakan simetris jika $a_{ij} = a_{ji}$ untuk semua nilai i dan j (entrinya adalah simetris dengan diagonal utama) (Kusumawati, 2014).

Contoh

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 2 & -3 & 0 \\ 4 & 0 & 9 \end{bmatrix} \text{ merupakan matriks simetris,}$$

karena

$$A^t = \begin{bmatrix} 1 & 2 & 4 \\ 2 & -3 & 0 \\ 4 & 0 & 9 \end{bmatrix} = A.$$

10) Matriks Skew-Simetris

Matriks persegi dengan sifat $A^t = -A$. Atau matriks persegi $A = (a_{ij})$ jika $a_{ij} = -a_{ji}$ untuk semua nilai i dan j dan elemen diagonal utamanya adalah nol, maka matriks persegi tersebut dinamakan matriks skew-simetris (Kusumawati, 2014).

Contoh

$$A = \begin{bmatrix} 0 & 2 & 1 \\ -2 & 0 & 3 \\ -1 & -3 & 0 \end{bmatrix}$$ merupakan matriks skew-simetris,

karena

$$A^t = \begin{bmatrix} 0 & -2 & -1 \\ 2 & 0 & -3 \\ 1 & 3 & 0 \end{bmatrix} = -\begin{bmatrix} 0 & 2 & 1 \\ -2 & 0 & 3 \\ -1 & -3 & 0 \end{bmatrix} = -A.$$

2.6 Kriptografi

Kriptografi merupakan sebuah ilmu yang mempelajari tentang pengamanan kerahasiaan pesan/data dengan menggunakan sandi. Pada awalnya kriptografi diartikan sebagai ilmu yang mempelajari tentang penyembunyian pesan. Tetapi sekarang kriptografi dapat diartikan sebagai ilmu yang berdasar pada teknik perhitungan matematika untuk keamanan suatu informasi (Sahara, 2018). Kriptografi digunakan untuk menjaga pesan rahasia dengan cara merubah ke dalam susunan pesan yang tidak dimengerti menggunakan sebuah sandi sehingga pesan tersebut terlihat seperti susunan huruf acak. Fungsi kriptografi bukan hanya sekedar kerahasiaan data *(privacy*) saja, tetapi juga bertujuan untuk menjaga integritas data (*data integrity*), keaslian data (*authentication*) dan anti penyangkalan (*nonrepudation*) (Manullang, 2018).

Kriptografi memiliki sejarah yang sangat panjang dan menarik. Kriptografi digunakan 4000 tahun yang lalu. Diperkenalkan oleh orang Mesir untuk mengirim informasi atau pesan kepada tentara di medan perang, sehingga bahkan jika utusan yang membawa informasi atau pesan tersebut tertangkap, musuh tidak dapat membaca informasi tersebut (Ariyus, 2006). Kriptografi pertama kali dikenal pada zaman Romawi Kuno. Suatu ketika seorang kaisar Romawi yang bernama Julius

Caesar ingin mengirim pesan rahasia kepada jendral di medan perang. Pesan tersebut bersifat sangat rahasia, dan perlu melalui perantara kurir dalam mengirimkannya. Karena pesan tersebut tidak ingin diketahui oleh orang lain bahkan kurir itu sendiri, maka Julius Caesar melakukan pengacakan huruf-huruf alfabet dengan cara mensubstitusi setiap huruf dengan huruf pengganti. Sehingga yang mengerti isi dari pesan itu hanya Julius Caesar dan jendral-jendralnya (Ariyus, 2008). Sehingga dari kejadian tersebut ditemukan suatu sistem kriptografi *Caesar Cipher* yang diambil dari nama Kaisar Romawi Kuno yaitu Julius Caesar.

2.6.1 Algoritma Kriptografi

Dari asal kata, algoritma memiliki sejarah yang sangat menarik. Kata tersebut muncul dalam kamus Webster, dan pada akhir tahun 1957 kata algoritma ditemukan dengan *algorism* dalam bahasa Arab yang berarti proses perhitungan. Algoritma ini berasal dari penulis buku berbahasa Arab yang terkenal, yaitu Abu Ja'far Muhammad ibnu Musa al-Khuwarizmi (al-Khuwarizmi dianggap sebagai *algorism* oleh orang Barat). Kata *algorism* secara bertahap menjadi algorithm (Ariyus, 2006).

Pengertian dari istilah algoritma adalah serangkaian langkah-langkah logis yang disusun secara sistematis untuk memecahkan suatu masalah. Algoritma enkripsi adalah langkah logis bagaimana menyembunyikan pesan dari orang yang tidak berhak mendapatkannya. Algoritma kriptografi terdiri dari tiga fungsi dasar yaitu enkripsi, dekripsi, dan kunci (Ariyus, 2006). Pada dasarnya kriptografi terdiri dari beberapa komponen, seperti (Ariyus, 2008):

1. *Enkripsi* : merupakan proses mengubah teks pesan asli (*plaintext*) kedalam bentuk teks pesan rahasia (*ciphertext*) dengan menggunakan algoritma kriptografi.
2. *Dekripsi* : merupakan kebalikan dari enkripsi, yaitu proses mengubah teks pesan rahasia (*ciphertext*) menjadi bentuk teks pesan asli (*plaintext*) dengan menggunakan algoritma kriptografi.
3. *Kunci* : merupakan kunci/kode yang digunakan untuk proses enkripsi dan dekripsi. Kunci dibagi menjadi dua bagian, yaitu kunci privat dan kunci publik.
4. *Ciphertext* : merupakan suatu pesan yang telah dienkripsi. *Ciphertext* tersusun secara acak dan tidak dapat dibaca karena berupa karakter-karakter yang tidak memiliki makna (arti). *Ciphertext* juga biasa disebut teks kode/pesan rahasia.
5. *Plaintext* : pesan asli atau teks biasa adalah pesan yang tertulis dan memiliki makna (arti). *Plaintext* tersebut akan diproses menggunakan algoritma enkripsi dan menjadi teks pesan rahasia (*ciphertext*).
6. *Pesan* : dapat berupa kumpulan teks, informasi, data yang akan dikirim (melalui kurir, media komunikasi, dll).
7. *Cryptanalysis* : merupakan analisis kode atau ilmu yang dapat memperoleh pesan asli (*plaintext*) tanpa mengetahui kuncinya. Jika sebuah teks rahasia (*ciphertext*) berhasil diubah menjadi pesan asli tanpa menggunakan kunci yang valid, hal itu disebut proses pemecahan kode. Orang yang melakukan ini disebut *Cryptanalys*. Analisis kode juga dapat digunakan untuk menemukan kelemahan dari suatu algoritma kriptografi dan akhirnya dapat

menemukan kunci atau pesan asli dari pesan rahasia yang dienkripsi dengan algoritma tertentu.

Enkripsi bertujuan untuk menjaga informasi rahasia/tersembunyi dari seseorang yang tidak berhak mengetahui informasi rahasia tersebut, serta menjamin kerahasiaan terhadap informasi yang telah dienkripsi. Sedangkan dekripsi merupakan kebalikan dari proses enkripsi, yaitu transformasi dari informasi rahasia yang telah melalui proses enkripsi yang menghasilkan *ciphertext* (pesan rahasia) dan kembali ke bentuk semula *plaintext* (pesan asli) (Pratama, 2014). Algoritma enkripsi merupakan fungsi yang digunakan untuk melakukan proses enkripsi dan dekripsi. Algoritma ini dibuktikan menggunakan basis matematika (Aprilia, 2005).

2.6.2 Kriptografi Klasik

Klasifikasi pada kriptografi dibedakan menjadi dua, yaitu kriptografi klasik dan kriptografi modern. Algoritma kriptografi klasik adalah teknik yang digunakan pada awal mula sejarah kriptografi yang pada dasarnya dilakukan dengan menggunakan skema yang sederhana, dan diimplementasikan menggunakan tangan atau mesin sederhana. Sedangkan algoritma kriptografi modern adalah teknik yang beroperasi dalam *bit* atau *byte*. Pada kriptografi modern juga digunakan metode-metode pada kriptografi klasik, tetapi dengan bantuan komputer sehingga algoritma dapat dibuat menjadi lebih rumit.

Algoritma dalam kriptografi klasik dibagi menjadi dua kategori yaitu:

• Sandi Subtitusi : Teknik penyandian yang dilakukan dengan mengganti setiap karakter dengan karakter lain dalam urutan huruf alfabet.

• Sandi Transposisi : Teknik penyandian yang dilakukan pada karakter *plaintext* melalui metode permutasi (Ariyus, 2008).

2.6.2.1 Teknik Substitusi

Sistem-sistem kriptografi klasik dapat dilihat dari operasi yang digunakan dalam enkripsi dan dekripsi. Salah satu operasi yang digunakan untuk enkripsi dan dekripsi adalah substitusi. Prinsip utama substitusi adalah mengganti kemunculan sebuah simbol dengan simbol lainnya. Sistem kriptografi yang menggunakan operasi substitusi disebut dengan sistem kriptografi berbasis substitusi (Sadikin, 2012). Berikut adalah beberapa sistem kriptografi berbasis substitusi:

1) Sandi *Caesar*

Sandi substitusi yang pertama dalam dunia penyandian pada waktu pemerintahan Yulius Caesar yang dikenal dengan *Caesar Cipher*, dengan mengganti posisi huruf awal dari alfabet (Ariyus, 2006). *Caesar Cipher* juga dikenal sebagai sandi geser dan merupakan algoritma enkripsi klasik yang paling sederhana. Cara kerja dari *Caesar Cipher* ini adalah dengan mengganti pesan asli (*plaintext*) dengan huruf baru yang sudah digeser posisinya sejauh kunci yang telah ditentukan. Misalkan dengan menggunakan kunci pergeseran 3, maka huruf A diganti dengan huruf D, huruf B diganti dengan huruf E, dan seterusnya (Sadikin, 2012).

Enkripsi *Caesar Cipher* juga dapat direpresentasikan menggunakan kongruensi modulo. Dengan langkah awal yaitu mengubah huruf menjadi angka menurut skema, $A = 0, B = 1, \dots, Z = 25$. Enkripsi pesan asli

(*plaintext*) dengan pergeseran n dapat dijelaskan secara matematis sebagai (Quist, 2013):

$$C : E(x) = (x + n) \; mod \; 26 \tag{2.18}$$

dekripsi dilakukan dengan cara yang sama, yaitu:

$$P : D(x) = (x - n) \; mod \; 26 \tag{2.19}$$

dengan

C : Karakter *ciphertext*

P : Karakter *plaintext*

E : Proses enkripsi

D : Proses dekripsi

x : karakter huruf (dalam angka)

n : kunci pergeseran

Untuk mempermudah penomoran karakter, dapat dilihat Gambar 2.1 berikut:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13

O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25

Gambar 2.1 Penomoran Karakter Alfabet

Contoh

Dengan menggunakan sandi Caesar dengan kunci $K = 7$ tentukan pesan rahasia berikut ini:

CAESAR PERGI KE GAUL

Petakan terlebih dahulu karakter alfabet menjadi bilangan pada $\mathbb{Z}_{26}$.

C	A	E	S	A	R	P	E	R	G	I	K	E	G	A	U	L
2	0	4	18	0	17	15	4	17	6	8	10	4	6	0	20	11

Gambar 2.2 Penomoran Karakter *Plaintext*

Setelah di dapatkan rangkaian bilangan, kemudian tambahkan dengan kunci K untuk tiap bilangan dalam aritmatika modular $\mathbb{Z}_{26}$. Sehingga hasilnya adalah:

9 7 11 25 7 24 22 11 24 13 15 17 11 13 7 1 1

Langkah terakhir adalah memetakan kembali dari bilangan di $\mathbb{Z}_{26}$ ke karakter alfabet. Sehingga *ciphertext* untuk pesan "CAESAR PERGI KE GAUL" dengan sandi Caesar dan $K = 7$ adalah: JHLZHY WLYNP RL NHBS

2) Sandi *Affine*

Sistem sandi *Affine* merupakan sandi dengan teknik substitusi menggunakan fungsi linier (Sadikin, 2012). Untuk proses enkripsi dapat dihitung secara matematis sebagai berikut:

$$C = aP + b \ (mod\ 26) \tag{2. 20}$$

dan untuk proses dekripsi dilakukan dengan:

$$P = a^{-1}(C - b) \ (mod\ 26) \tag{2. 21}$$

dengan

C : Karakter *ciphertext*

P : Karakter *plaintext*

a : kunci

b : kunci

Jadi kunci untuk proses enkripsi sandi *Affine* terdiri dari dua parameter yaitu a dan b. Agar a mempunyai *inverse* (a^{-1}), maka a harus memenuhi $\gcd(a, 26) = 1$. Untuk karakter penomoran alfabet masih tetap sama seperti *Caesar Cipher* seperti pada Gambar 2.1.

Contoh

Misalkan terdapat teks pesan asli “BIMA SAKTI” dan sepasang kunci $a = 15$ dan $b = 7$. Akan dilakukan penyandian pesan sebagai berikut:
Akan dilakukan penghitungan secara matematis sesuai yang telah dijelaskan diatas.

$$C = aP + b\ (mod\ 26)$$

$$C = 15(1) + 7\ mod\ 26 = 22 : W$$

$$C = 15(8) + 7\ mod\ 26 = 23 : X$$

$$C = 15(12) + 7\ mod\ 26 = 5 : F$$

$$C = 15(0) + 7\ mod\ 26 = 7 : H$$

$$C = 15(18) + 7\ mod\ 26 = 17 : R$$

$$C = 15(0) + 7\ mod\ 26 = 7 : H$$

$$C = 15(10) + 7\ mod\ 26 = 1 : B$$

$$C = 15(19) + 7\ mod\ 26 = 6 : G$$

$$C = 15(8) + 7\ mod\ 26 = 23 : X$$

Maka dari proses perhitungan diperoleh *ciphertext*:

“WXFH RHBGX”.

3) Sandi *Playfair*

Sandi *Playfair* ditemukan oleh sir Charles Wheatstone dan Baron Lyon Playfair pada tahun1854 dan pertama kali digunakan pada awal abad ke-20 untuk mengirim pesan antar pangkalan di Inggris selama Perang Dunia Pertama. Kunci *Playfair* menggunakan matriks 5×5 (input terdiri dari 25 karakter dan menghapus huruf J dari alfabet), sehingga kunci yang digunakan adalah 25 huruf alfabet (Ariyus, 2006). Dalam beberapa versi, yang

dihilangkan adalah huruf Q, sedang dalam versi lain huruf I dan J ditulis sebagai I / J (Munir, 2019).

S	T	A	N	D
E	R	C	H	B
K	F	G	I/J	L
M	O	P	Q	U
V	W	X	Y	Z

Gambar 2.3 Contoh Papan Kunci 5 × 5

Contoh

Misalkan diketahui *plaintext* : Didalam jiwa yang sehat terdapat akal yang sehat. Untuk melakukan enkripsi dengan menggunakan *Playfair Cipher* mempunyai beberapa aturan seperti di bawah ini:

1. Aturan enkripsi dan dekripsi mengikuti aturan segiempat, dan karakter yang ada terlebih dahulu dibagi menjadi dua karakter di setiap bagian. Jika dua huruf (karakter) tidak berada dalam satu baris atau kolom, maka pergeseran karakter dimulai dari huruf kedua secara vertikal ke karakter *ciphertext* pertama. Dengan asumsi bahwa huruf kedua *plaintext* "di" adalah "i", dan kemudian bergerak secara vertikal dari "i" dalam matriks untuk menemukan huruf yang satu baris dengan "d", maka akan ditemukan karakter "n"(sebagai *ciphertext*). Untuk karakter kedua "d", cari sisi lain seperti cara karakter "i", lalu cari karakter "l" sehingga *ciphertext* "di" adalah "nl".

S	T	A	N	D
E	R	C	H	B
K	F	G	I	L
M	O	P	Q	U
V	W	X	Y	Z

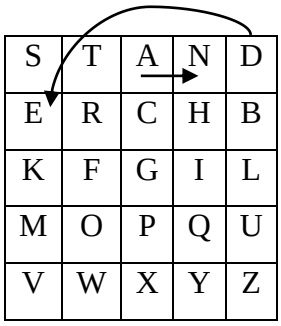
Gambar 2.4 Contoh Aturan *Encipher Right, Decipher Left*

2. Jika karakter yang dienkripsi atau didekripsi ada di kolom atau baris yang sama dan saling berdekatan, maka digunakan prinsip enkripsi atau dekripsi ke bawah atau ke samping. Contoh untuk mengenkripsi "an" maka karakter di samping "n" adalah "d" dan karakter di samping "a" adalah "n" sehingga *ciphertext* nya menjadi "dn".

S	T	A	N	D
E	R	C	H	B
K	F	G	I	L
M	O	P	Q	U
V	W	X	Y	Z

Gambar 2.5 Contoh 1 Aturan *Encipher Below, Decipher Above*

3. Jika karakter yang dienkripsi berada di akhir baris, ikuti aturan nomor 2 di atas, tetapi dalam kasus baris terakhir, karakter yang diambil dalam *ciphertext* adalah karakter di sebelahnya (yakni baris pertama setelah baris tersebut).



Gambar 2.6 Contoh 2 Aturan *Encipher Below, Decipher Above*

4. Jika ada karakter ganda untuk menggunakan sandi *Playfair*, masukkan karakter di antara karakter, misalnya "aa", "ii" menjadi "aza", "izi" tergantung pada kesepakatan karakter yang digunakan apabila terdapat kasus seperti ini.
5. Untuk menganalisis sandi *Playfair* aturan satu, aturan dua, dan aturan tiga diberi singkatan, aturan satu ERDL (*encipher right, decipher left*), sedangkan aturan dua dan tiga EBDA (*encipher below, decipher above*).

   Jadi, dari proses enkripsi diperoleh *ciphertext* pada Gambar 2.7 di bawah ini:

Plaintext	di	da	la	mj	iw	ay	an	gs	eh	at
Ciphertext	nl	ne	gd	kq	fy	nx	dn	ak	br	na

Plaintext	te	rd	ap	ta	ka	ly	an	gs	eh	at
Ciphertext	rs	tb	xc	na	gs	iz	dn	ak	br	na

Gambar 2.7 Hasil *Ciphertext Playfair Cipher*

Ciphertext : nlnegdkqfynxdnakbrnarstbxcnagsizdnakbrna.

Playfair Cipher adalah metode enkripsi klasik, dan sulit untuk melakukan kriptanalisis secara manual. Namun, sandi *Playfair* dapat diselesaikan dengan menggunakan informasi tentang frekuensi kemunculan bigram. Bagian penting dari metode *Playfair* adalah tabel kunci yang

digunakan untuk melakukan enkripsi dan dekripsi. Tabel bawaan yang diperkenalkan oleh *Playfair* adalah tabel matriks 5×5 yang berisi huruf kapital $A - Z$, dengan huruf J dihilangkan dan diganti dengan huruf I. Berikut adalah beberapa aturan yang digunakan dalam metode sandi *Playfair* (Susanti, 2020):

1. Jika ada huruf J, ganti dengan huruf I.
2. Tulis pesan dalam pasangan huruf, yaitu memisahkan dua huruf.
3. Tidak boleh ada huruf yang sama, jadi sisipkan huruf atau karakter yang jarang digunakan di tengah huruf yang sama (ganda).
4. Jika jumlah huruf ganjil, tambahkan huruf atau karakter yang jarang digunakan di akhir dari tabel dibentuk.

Playfair adalah *digraphs cipher*, yang berarti bahwa setiap proses enkripsi dilakukan pada setiap dua huruf. Misalnya *plaintext*nya adalah "KRIPTOLOGI", maka menjadi "KR|IP|TO|LO|GI". Kunci yang digunakan berupa kata-kata, tanpa mengulang huruf yang sama. Jika kuncinya adalah "SARAPAN", maka kunci yang digunakan adalah "SARPN". Kemudian masukkan kunci ke dalam tabel. Entri pertama adalah kunci, dan kemudian tuliskan huruf-huruf berikutnya dari baris pertama. Jika huruf-huruf ini sudah muncul, tidak akan ditulis kembali pada tabel. Berikut penggunaan kunci "SARPN" pada tabel 5×5 (Susanti, 2020):

S	A	R	P	N
B	C	D	E	F
G	H	I	K	L
M	O	Q	T	U
V	W	X	Y	Z

Gambar 2.8 Contoh *Input* Alfabet Pada Papan Kunci 5 × 5

4) Sandi *Vigenere*

Sandi *Vigenere* merupakan algoritma substitusi yang diperoleh dari modifikasi Sandi *Caesar*. Teknik ini diberi nama sesuai dengan penemunya, yaitu *Blaise de Vigenere* dari Istana Henry III Prancis pada abad ke-16, dan dianggap tidak dapat dipecahkan selama 300 tahun (Quist, 2013). Teknik sandi *Vigenere* dapat dilakukan dengan dua cara, yakni : Angka dan Huruf.

a) Angka

Sandi *Vigenere* menggunakan angka untuk mengganti karakter huruf dengan angka (dapat dilihat pada Gambar 2.1 penomoran karakter alfabet pada Sandi *Caesar*). Sandi *Vigenere* merupakan metode enkripsi huruf dengan menggunakan serangkaian enkripsi Sandi *Caesar*, yang menjadi pembeda adalah dari kata kuncinya. Kunci yang digunakan pada Sandi *Vigenere* berupa huruf yang disusun dan membentuk suatu kata. Dengan cara ini, setiap huruf dari *plaintext* yang sama dapat memiliki huruf *ciphertext* yang berbeda, tergantung pada kunci huruf yang digunakan. Jadi dengan menggunakan *vigenere password*, kita dapat mencegah frekuensi huruf pada *ciphertext* muncul sama dengan frekuensi huruf pada *plaintext* (Munir, 2019). Sama seperti *Caesar Cipher*, penjumlahan dilakukan dengan $modulo\ 26$ karena karakter huruf hanya berhenti hingga angka 25. Enkripsi

Sandi *Vigenere* teks pesan asli (*plaintext*) dengan kunci k dapat dijelaskan secara matematis sebagai (Quist, 2013):

$$E: C_i = (P_i + K_i)\ mod\ 26 \quad (2.22)$$

dekripsi dilakukan dengan cara yang sama, yaitu:

$$D: P_i = (C_i - K_i)\ mod\ 26 \quad (2.23)$$

dimana

E : Proses enkripsi

D : Proses eekripsi

P_i : Karakter *plaintext*

C_i : Karakter *ciphertext*

K_i : Karakter kunci

i : urutan karakter

Keamanan sandi *Vigenere Cipher* bergantung dengan jumlah kunci yang digunakan. Semakin panjang kunci yang digunakan maka penyelesaiannya semakin sulit dipecahkan. Jika jumlah kunci adalah n maka ruang kunci sandi *Vigenere Cipher* adalah M^n dengan M adalah jumlah alfabet yang digunakan pada sistem *Vigenere Cipher*. Misalkan, jika jumlah alfabet adalah 26 dan sandi *Vigenere Cipher* adalah 5 maka ruang kuncinya adalah $26^5 \approx 10^7$(Sadikin, 2012)

b) Huruf

Teknik substitusi *Vigenere* dengan menggunakan huruf adalah dengan cara memperluas dan mengatur ulang tabel *Vigenere*, oleh karena itu membuatnya jauh lebih kompleks daripada yang sudah ada. Kumpulan karakter yang lebih besar memungkinkan lebih banyak jenis pesan untuk

dienkripsi. Hal ini dapat meningkatkan keamanan pesan (Quist, 2013). Berikut adalah tabel karakter pada sandi *Vigenere* dengan menggunakan teknik huruf:

Tabel 2.1 Karakter Huruf Pada Sandi *Vigenere*

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Cara menentukan *ciphertext* sistem ini, pada tabel dapat dilihat pada posisi horizontal merupakan *plaintext* dan pada posisi vertikal merupakan kunci. Jika *plaintext* adalah huruf *K* maka lihat posisi letak huruf *K* pada *plaintext* dan posisi huruf *K* pada kunci dalam tabel. Apabila sudah menemukan, tarik garis lurus ke bawah dari *plaintext* dan garis lurus ke samping dari kunci,

kita akan menemukan huruf U. Sehinngga huruf U yang akan menjadi *ciphertext* dan begitu seterusnya (Ariyus, 2006).

Contoh

Tentukan *ciphertext* untuk teks pesan asli "BULANPURNAMA" dengan menggunakan sandi *vigenere* dengan himpunan kunci $\{7, 10, 21\}$.

Dengan menggunakan himpunan kunci secara berulang dapat diperoleh *ciphertext* dengan hitungan matematis (menggunakan cara angka) sebagai berikut:

$$c[0] = 1 + 7 \; mod \; 26 = 8 : I$$

$$c[1] = 20 + 10 \; mod \; 26 = 4 : E$$

$$c[2] = 11 + 21 \; mod \; 26 = 6 : G$$

$$c[3] = 0 + 7 \; mod \; 26 = 7 : H$$

$$c[4] = 13 + 10 \; mod \; 26 = 23 : X$$

$$c[5] = 15 + 21 \; mod \; 26 = 10 : K$$

$$c[6] = 20 + 7 \; mod \; 26 = 1 : B$$

$$c[7] = 17 + 10 \; mod \; 26 = 1 : B$$

$$c[8] = 13 + 21 \; mod \; 26 = 8 : I$$

$$c[9] = 0 + 7 \; mod \; 26 = 7 : H$$

$$c[10] = 12 + 10 \; mod \; 26 = 22 : W$$

$$c[11] = 0 + 21 \; mod \; 26 = 21 : V$$

Oleh karena itu setelah transformasi dari angka ke huruf didapatkan teks sandi :

"IEGHXKBBIHWV"

5) Sandi *Hill*

Sandi *Hill* adalah sandi *polyalphabet* yang menggunakan metode alternatif perhitungan perkalian matriks. Kunci pada sandi *Hill* adalah sebuah matriks K berukuran $n \times n$ yang digunakan untuk mengganti n alfabet sekaligus. Matriks K harus mempunyai *inverse*, dan nilai determinan dari matriks K juga harus mempunyai matriks *inverse* perkalian pada $\mathbb{Z}_{26}$ (karena jumlah alfabet yang digunakan berjumlah 26) (Sadikin, 2012). Maksud dari polialfabetik disini adalah untuk setiap karakter alfabet bisa dipetakan ke lebih dari satu macam karakter alfabet. *Cipher* ini ditemukan pada tahun 1929 oleh Lester S. Hill. Misalkan m adalah bilangan bulat positif, dan $P = C = (Z_{26})^m$. Ide dari *Hill cipher* adalah untuk mengambil m kombinasi linier dari m karakter alfabet dalam elemen *plaintext*, sehingga diperoleh m karakter alfabet dalam elemen *plaintext* (Ariyus, 2006).

Diasumsikan $m = 2$, maka kita dapat menulis elemen *plaintext* sebagai $x = (x_1, x_2)$ dan elemen *ciphertext* sebagai $y = (y_1, y_2)$. Di sini, y_1, y_2 adalah kombinasi linier dari x_1 dan x_2. Katakanlah

$$y_1 = 11x_1 + 3x_2$$

$$y_2 = 8x_1 + 7x_2.$$

Kita juga dapat menuliskannya dalam notasi matriks sebagai berikut:

$$(y_1, y_2) = (x_1, x_2)\left[\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}\right].$$

Secara umum, kita akan menggunakan matriks K berukuran $m \times m$ sebagai kunci kita. Jika elemen pada baris i dan kolom j dari matriks K adalah k_{ij}, maka kita tuliskan $K = (k_{ij})$. Untuk $x = (x_1, \dots, x_m)\hat{I}P$ dan $K\hat{I}K$, kita menghitung $y = e_K(x) = (y_1, \dots, y_m)$ sebagai berikut:

$$(y_1, y_2, \dots, y_m) = (x_1, x_2, \dots, x_m)\begin{pmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,m} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ k_{m,1} & k_{m,2} & \cdots & k_{m,m} \end{pmatrix},$$

atau dapat ditulis, $y = xK$.

Dikatakan bahwa *ciphertext* diperoleh dari *plaintext* dengan transformsi linier. Untuk dekripsi, kita gunakan matriks *inverse* K^{-1}. Jadi dekripsi dilakukan dengan rumus $x = yK^{-1}$.

- Matriks memiliki sifat asosiatif pada perkalian, yakni $(AB)C = A(BC)$.
- Matriks identitas $m \times m$ ditulis I_m, merupakan matriks yang elemennya adalah 1 pada diagonal utama dan pada elemen lainnya berisi 0. Contohnya:

  $$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

  I_m dikatakan sebagai matriks identitas, karena $AI_m = A$ untuk matriks $1 \times m$ dan $I_m B = B$ untuk matriks $m \times n$.
- Matriks *inverse* dari A (jika ada) adalah A^{-1} dimana $AA^{-1} = A^{-1}A = I_m$.

  Dengan menggunakan sifat-sifat matriks di atas, maka:

  $$y = xK$$

  $$yK^{-1} = (xK^{-1}) = x(KK^{-1}) = xI_m = x$$

  (Ariyus, 2006).

Contoh

Mencari *ciphertext* apabila diberikan teks pesan asli "MATAHARI" dengan menggunakan sandi *Hill* dan kunci matriks

$$K = \begin{pmatrix} 10 & 7 \\ 5 & 21 \end{pmatrix}.$$

Karena K matriks dengan ukuran 2×2, maka enkripsi teks dengan 2 karakter dilakukan sekali waktu (sekaligus) menggunakan sandi *Hill*. Ubah terlebih dahulu teks asli "MATAHARI" dalam urutan angka yakni $\{12, 0, 19, 0, 7, 0, 17, 8\}$. Kemudian dilakukan enkripsi pesan dengan membentuk vektor ukuran 2×1 secara berurutan dari 2 karakter pada teks pesan asli. Untuk karakter ke-1 dan ke-2 dapat dihitung sebagai berikut:

$$\begin{pmatrix} C[0] \\ C[1] \end{pmatrix} = \begin{pmatrix} 10 & 7 \\ 5 & 21 \end{pmatrix} \begin{pmatrix} 12 \\ 0 \end{pmatrix} = \begin{pmatrix} 16 \\ 8 \end{pmatrix}.$$

Oleh karena itu nilai karakter pertama dan kedua pada *ciphertext* diatas adalah "Q" dan "T". Hal tersebut dilakukan secara berulang dengan pasangan 2 karakter. Sehingga diperoleh teks sandi "QIIRSJST".

Untuk proses dekripsi menggunakan sandi *Hill* dapat dilakukan dengan cara mengkalikan matriks *inverse* kunci dengan 2 karakter berurutan dari *ciphertext*.

2.6.2.2 Teknik Transposisi

Pada bagian sebelumnya sudah dijelaskan tentang teknik dari substitusi *cipher*. Pada bagian ini, kita akan membahas teknik permutasi (transposisi *cipher*). Teknik ini menggunakan permutasi karakter, dan pesan asli tidak dapat dibaca menggunakan teknik ini kecuali memiliki kunci untuk mengembalikan pesan ke bentuk aslinya.

Contoh

Ada 6 kunci yang digunakan untuk melakukan permutasi *cipher*, pertama buat indeks sepanjang 6, kemudian tentukan kunci dengan mengacak penomoran indeks.

Indeks	1	2	3	4	5	6
Kunci	3	5	1	6	4	2

Gambar 2.9 Indeks Dan Kunci Pada Permutasi *Cipher*

Dan 6 kunci untuk *inverse* dari permutasi tersebut adalah dengan mengubah kunci menjadi urutan indeks. Kemudian kunci diperoleh dari pasangan indeks dan kunci pada urutan sebelumnya.

Indeks	1	2	3	4	5	6
Kunci	3	6	1	5	2	4

Gambar 2.10 *Inverse* Kunci Permutasi *Cipher*

Misalkan kita ingin melakukan enkripsi terhadap kalimat berikut:

"SAYA SEDANG BELAJAR KEAMANAN KOMPUTER"

Pertama, kalimat dibagi menjadi 6 blok. Jika blok tidak mencukupi, huruf pilihan dapat ditambahkan. Dalam contoh ini, huruf X ditambahkan, yang membantu memperumit kriptanalisis. Maka menjadi:

SAYASE DANGBE LAJARK EAMANA NKOMPU TERXXX

Setelah dibagi menjadi 6 blok, dengan menggunakan kunci blok di atas, perubahan pada setiap blok adalah sebagai berikut:

YSSEAA NBDEGA JRLKAA MNEAAA OPNUMK RXTXXE

Jadi, proses enkripsi menghasilkan *ciphertext*:

"YSSEAANBDEGAJRLKAAMNEAAAOPNUMKRXTXXE"

Untuk mengembalikan pesan rahasia ke *plaintext*, *ciphertext* dibalik sesuai dengan kunci nomor dua diatas. Banyak teknik permutasi lainnya, seperti zig-zag, segitiga, spiral, diagonal.

Zig-zag : Menyusun *plaintext* menggunakan pola zig-zag, *ciphertext* dari teknik ini diperoleh dengan membaca dari baris atas ke baris bawah (dari kiri ke kanan), misalkan pada contoh berikut:

			A						G						A						A						M					X
		Y		S				N		B				J		R				M		N				O		P				R	
	A				E		A				E		A				K		A				A		K				U		E		
S						D						L						E						N					T				

Gambar 2.11 Teknik Permutasi Zig-zag

*Cipherteks*nya adalah:

"AGAAMXYSNBJRMNOPRAEAEAKAAKUESDLENTX"

Segitiga : Menyusun *plaintext* dengan pola segitiga, dan *ciphertext* diperoleh dengan menyusun karakter dari atas ke bawah, misalkan pada contoh berikut:

					S					
				A	Y	A				
			B	E	L	A	J			
		R	K	E	A	M	A	N		
	A	N	K	O	M	P	U	T	E	
R	X	X	X	X	X	X	X	X	X	X

Gambar 2.12 Teknik Permutasi Segitiga

*Ciphertext*nya adalah:

"RAXRNXBKKXAEEOXSYLAMXAAMPXJAUXNTXEXX"

Spiral : Menyusun *plaintext* dengan pola spiral, *ciphertext* diperoleh dengan menyusun karakter dari atas ke bawah, misalkan pada contoh berikut:

S	A	Y	A	S	E
A	M	A	N	A	D
E	E	R	X	N	A
K	T	X	X	K	N
R	U	P	M	O	G
A	J	A	L	E	B

Gambar 2.13 Teknik Permutasi Spiral

*Ciphertext*nya adalah:

"SAEKRAAMETUJYARXPAANXXMLSANKOEEDANGB"

Diagonal : Menyusun *plaintext* dengan pola dari atas ke bawah, *ciphertext* diperoleh dengan cara membaca dari kiri ke kanan, misalkan pada contoh berikut:

S	D	L	E	N	T
A	A	A	A	K	E
Y	N	J	M	O	R
A	G	A	A	M	X
S	B	R	N	P	X
E	E	K	A	U	X

Gambar 2.14 Teknik Permutasi Diagonal

*Ciphertext*nya adalah:

"SDLENTAAAAKEYNJMORAGAAMXSBRNPXEEKAUX"

Dari teknik transposisi (permutasi) dengan berbagai pola, pesan dapat disembunyikan dari orang yang tidak berwenang. Kombinasi ini membentuk dasar dari algoritma kriptografi (modern) yang kita kenal sekarang (Ariyus, 2006). Berikut adalah beberapa sistem kriptografi berbasis transposisi:

1) Sandi Transposisi *Columnar*

Sandi transposisi yang paling sederhana, salah satunya adalah sandi transposisi *Columnar*. Cara kerja sandi *Columnar* adalah dengan menuliskan karakter teks pesan asli (*plaintext*) dengan panjang karakter yang sama pada arah baris, kemudian menulis ulang teks pesan rahasia pada arah kolom. Urutan kolom disepakati terlebih dahulu untuk memperumit kriptanalisis (Sadikin, 2012).

Contoh

Misalnya ada teks asli “MUNDURSAMPAIBATASKOTA”, kemudian menuliskannya dalam tabel yang terdiri dari 6 kolom dan baris, kita peroleh:

Tabel 2.2 *Input* Kunci Dan Teks Asli Sandi Transposisi *Columnar*

Kunci	4	2	1	6	3	5
Teks asli	M	U	N	D	U	R
	S	A	M	P	A	I
	B	A	T	A	S	K

String XYZ di baris terakhir digunakan untuk mengisi sel kosong dalam tabel. Setelah tabel terbentuk, *ciphertext* ditulis dalam urutan kunci pada arah kolom. Jadi kita mendapatkan *ciphertext*:

NMTAUAATUASYMSBORIKZDPAX

Dekripsi sandi *Columnar* hampir sama dengan enkripsi, tetapi saat mendekripsi, jumlah baris dapat dihitung dengan membagi panjang *ciphertext* dengan panjang kunci. Kemudian isilah kolom pertama sesuai panjang kunci dengan teks sandi sampai baris terakhir.

Besar ruang kunci sandi *Columnar* bergantung dengan jumlah kolom yang digunakan yaitu $1! + 2! + 3! + \cdots + N!$ dengan N adalah jumlah kolom. Namun karena sandi transposisi tidak menghilangkan informasi statistik teks pesan asli maka pola-pola yang muncul pada teks sandi (*ciphertext*) dapat digunakan untuk menerka panjang kunci dan kemudian dapat digunakan untuk memecahkan kunci. Untuk meningkatkan keamanan enkripsi sandi transposisi *Columnar* dapat dilakukan dua kali (disebut dengan *double columnar*) (Sadikin, 2012).

2) Sandi Transposisi *Rail Fence*

Rail Fence Cipher adalah algoritma *cipher* transposisi yang mengacak urutan alfabet pada pesan. Algoritma ini menuliskan *plaintext* satu per satu, baris atas dan baris bawah, dan *ciphertext* membaca huruf baris demi baris. Algoritma ini mengatur naik dan turun sesuai dengan ukuran (panjang) kolom dan baris yang ditentukan oleh kunci, dan mengatur *plaintext* secara 'zig-zag'. *Ciphertext* diperoleh dengan membaca susunan alfabet secara horizontal. Sandi ini digunakan selama Perang Saudara Amerika, ketika digunakan untuk menyembunyikan informasi militer Union dan mata-mata Konfederasi (Girsang dkk, 2019).

Contoh

Sebagai contoh misal $n = 5$ dan *plaintext* = "TETAP SEMANGAT"

T								M					
	E						E		A				
		T				S				N			
			A								G		T
				P								A	

Gambar 2.15 Teknik Transposisi *Rail Fence Cipher*

Plaintext yang ditulis dibagi menjadi blok-blok dengan panjang 5 karakter. Jika masih ada kotak yang belum diisi karakter secara 'zig-zag', dan semua huruf sudah dibuat blok, dapat ditambahkan karakter *dummy* di akhir karakter. Maka dihasilkan *ciphertext* : TMEEATSNAGTPA

3) Sandi Transposisi *Route*

Sandi *Route* adalah sandi transposisi, ditulis dalam *plaintext* berdasarkan kunci yang telah ditentukan. *Ciphertext* dibaca sesuai dengan pola yang ditentukan oleh kunci. *Route Cipher* ini adalah sandi militer yang

diperkenalkan oleh Kolonel Parker Hitt pada tahun 1916 untuk tentara dan warga sipil (Girsang dkk, 2019).

Contoh

Sebagai contoh misal $n = 7$, spiral ke dalam jarum jam kanan atas, *plaintext* = "TETAP SEMANGAT"

T	T	P	E	A	G	T
E	A	S	M	N	A	X

Gambar 2.16 Teknik Transposisi *Route Cipher*

Algoritma sandi *Route* dapat dikatakan proses enkripsi yang kompleks. Hal ini dikarenakan kunci yang digunakan membuat proses enkripsi dan dekripsi menjadi fleksibel. Jika panjang karakter tidak habis dibagi dengan panjang karakter, kita dapat menambahan karakter *dummy* selama proses enkripsi.

Hasil *ciphertext* : TXANMSAETTPEAG

4) Sandi *Bifid*

Bifid Cipher adalah berbentuk matriks, atau termasuk kedalam *cipher* transposisi. *Bifid Cipher* berbentuk matriks dengan ordo 5×5. Algoritma *Bifid Cipher* tergolong pada algoritma kriptografi klasik yang ditemukan oleh Felix Delestelle. Algoritma ini menggunakan persegi polibius dan pemecahan karakter untuk memperoleh efek difusi pada *ciphertext*nya (Manullang, 2018). Sampai saat ini *Bifid Cipher* masih merupakan algoritma penyandian sederhana, biasanya hanya menggunakan pensil dan kertas yang sangat aman digunakan untuk menyelesaikan.

Bifid Cipher adalah algoritma yang menggunakan metode *Playfair Cipher*, dimana teks pesan asli (*plaintext*) diacak menjadi tabel persegi 5×5 dengan mengacak posisi baris dan kolom huruf pada pesan yang sesuai

dengan papan kunci (bentuk matriks), seperti penerapan pada *Playfair Cipher*. Setiap baris dan kolom pada susunan papan kunci diisi dengan huruf alfabet yang disusun secara acak (Wulandari, 2019). Sama seperti pada *Playfair Cipher*, dalam memasukkan huruf pada papan kunci akan menghilangkan huruf J terlebih dahulu dan menggantiya dengan huruf I. Untuk papan kunci dalam bentuk matriks berordo 5×5 ini dapat ditulis sebagai berikut:

$$\begin{pmatrix} A & B & C & D & E \\ F & G & H & I & K \\ L & M & N & O & P \\ Q & R & S & T & U \\ V & W & X & Y & Z \end{pmatrix}.$$

Contoh

Misalkan *plaintext* adalah "SELAMAT MALAM" dan kunci yang digunakan adalah "salam". Maka dengan menghilangkan huruf-huruf berulang, kalimat pada kunci menjadi "salm". Dengan memasukkan huruf-huruf tersebut ke dalam tabel, dan mengisi kolom-kolom kosong yang tidak terisi dengan huruf yang belum pernah ditulis kecuali huruf J, maka akan diperoleh tabel kunci yang digunakan untuk penyandian dengan *Bifid cipher* seperti di bawah ini:

S	A	L	M	B
C	D	E	F	G
H	I	K	N	O
P	Q	R	T	U
V	W	X	Y	Z

Gambar 2.17 Contoh *Input* Alfabet Pada Papan Kunci 5×5 Sandi *Bifid*

Proses enkripsi dilakukan dengan mengidentifikasi posisi setiap huruf berdasarkan baris dan kolom setiap huruf.

Plainteks : S E L A M A T M A L A M

Baris : 1 2 1 1 1 1 4 1 1 1 1 1

Kolom : 1 3 3 2 4 2 4 4 2 3 2 4

Kemudian *plaintext* tersebut dienkripsi dengan mengacak koordinat posisi dan menjadikannya koordinat posisi baru dengan aturan tertentu. Dalam hal ini, aturan yang digunakan adalah mengubah koordinat baris dan kolom menjadi deretan karakter numerik, dan kemudian mengubahnya menjadi bigram (dipisahkan dua baris) sebagai koordinat baru, dan isi sel adalah hasil enkripsi. Proses enkripsi akan ditampilkan secara sistematis sebagai berikut:

Koordinat baris dan kolom dibuat menjadi satu baris karakter numerik.

1 2 1 1 1 1 4 1 1 1 1 1 1 3 3 2 4 2 4 4 2 3 2 4

Kemudian pasangkan angka bersebelahan yang akan membentuk koordinat baru baris dan kolom.

12 11 11 41 11 11 13 32 42 44 23 24

Kemudian *ciphertext* adalah isi sel dengan koordinat yang baru terbentuk.

Baris : 1 1 1 4 1 1 1 3 4 4 2 2

Kolom : 2 1 1 1 1 1 3 2 2 4 3 4

Ciphertext : A S S P S S L I Q T E F

Maka hasil dari penggunaan metode *Bifid Cipher* untuk melakukan enkripsi adalah sebagai berikut, *ciphertext* : ASSPSSLIQTEF

2.7 Super Enkripsi

Penggabungan dari 2 metode Sandi Subtitusi dan Sandi Transposisi dapat dikatakan sebagai Super enkripsi. Super enkripsi adalah sebuah konsep yang menggunakan dua atau lebih teknik kriptografi substitusi dan permutasi (transposisi) untuk mendapatkan algoritma yang lebih sulit untuk dipecahkan. Hal pertama yang harus dilakukan adalah menggunakan teknik substitusi (*Cipher* Subtitusi) untuk mengenkripsi pesan, dan kemudian menggunakan teknik permutasi (*Cipher* Transposisi) untuk mengenkripsi kempali *ciphertext* yang diperoleh (Ariyus, 2008).

Contoh

Misal diketahui *plaintext*:

"KENAIKAN HARGA BBM MEMBUAT RAKYAT KECIL MENDERITA"

1. Gunakan teknik sandi substitusi menggunakan algoritma *Caesar Cipher* dan kuncinya adalah 6. *ciphertext* yang didapat:

   "QKTGOQGTNGXMGHHSSKSHAGZQKIORSKTJKXOZG"

2. Gunakan teknik sandi transposisi menggunakan teknik diagonal permutasi dan kuncinya adalah 4. Seperti pada Gambar 2.18 berikut:

Q	K	T	G
O	Q	G	T
N	G	X	M
G	H	H	S
S	K	S	H
A	G	Z	Q
K	I	O	R
S	K	T	J
K	X	O	Z
G	X	X	X

Gambar 2.18 Teknik Tansposisi Diagonal Permutasi Untuk Super Enkripsi

Hasil akhir *ciphetext* adalah sebagai berikut:

"QONGSAKSKGKQGHKGIKXXTGXHSZOTOXGTMSHQRJZX".

Teknik dari super enkripsi sangat penting dan banyak dari algoritma enkripsi modern menggunakan teknik ini sebagai dasar pembuatan suatu algoritma (Ariyus, 2006).

2.8 *Python*

Python pertama kali dikembangkan oleh programmer kelahiran Belanda, Guido van Rossum pada tahin 1991 di CWI, Amsterdam. Guido sangat suka pada acara televisi Monty Python's Flying Circus, oleh karena itu ia memberi nama *Python* pada bahasa pemrograman ciptaannya tersebut. *Python Software Foundation* bersama dengan Guido mengkoordinir pengembangan *Python* yang saat ini masih terus dilakukan oleh sekelompok programmer. Hingga kini distribusi *Python* sudah mencapai 3.7 (Enterprise, 2019).

Berikut beberapa fitur dan kelebihan pada bahasa pemrograman *Python*:

1) Sudah tersedia modul pemrograman yang tersebar di internet dan dapat diakses dengan mudah.
2) Struktur bahasa yang sederhana, jelas dan mudah dimengerti.
3) Sistem pengelolaan memori otomatis (*garbage collection*) seperti Java.
4) Mudah dikembangkan dengan menciptakan modul-modul baru.

2.9 Kajian Keagamaan

Allah SWT menurunkan berbagai ilmu pengetahuan kepada makhluknya. Salah satunya yaitu ilmu matematika yang merupakan ilmu tentang pengukuran dan

perhitungan. Sebagaimana juga yang dijelaskan pada Al-Quran surat Al-Jin ayat 28:

لِّيَعْلَمَ أَنْ قَدْ أَبْلَغُوا۟ رِسَٰلَٰتِ رَبِّهِمْ وَأَحَاطَ بِمَا لَدَيْهِمْ وَأَحْصَىٰ كُلَّ شَىْءٍ عَدَدًا۟ ﴿28﴾

"Supaya Dia Mengetahui, bahwa sesungguhnya rasul-rasul itu telah menyampaikan risalah-risalah Tuhannya, sedang (sebenarnya) ilmu-Nya meliputi apa yang ada pada mereka, dan Dia menghitung segala sesuatu satu persatu". (Q.S Al-Jin/72 : 28).

Menurut buku Tafsir Al Misbah, ayat sebelumnya (Al-Jin ayat 27) menjelaskan pemeliharaan dan perlindungan yang gaib serta wahyu kepada para rasul yang menerimanya. Ayat di atas (Al-Jin ayat 28) menjelaskan bahwa Allah berhati-hati agar Dia mengetahui kebenaran setelah sebelumnya Dia mengetahuinya dengan ilmu-Nya yang kekal (abadi), sesungguhnya rasul telah menyampaikan risalah Tuhannya, dan sesungguhnya dia membawa ilmu dan kekuasaan-Nya, termasuk menspesifikasikan semua yang ada didalamnya tidak hanya terkait dengan penyampaian risalah, tetapi Dia (satu-persatu) menghitung semua yang ada, meski hanya sebutir pasir. Tidak ada yang bisa lepas dari pengetahuan-Nya.

Ayat diatas menunjukkan bahwa wahyu Allah senantiasa terpelihara karena diterima dari malaikat oleh rasul sebagai perantara, yang meyakinkan rasul bahwa tidak ada sedikitpun perubahan, tidak dilupakan, ditambah, atau bahkan dikurangi, menandakan bahwa tidak ada keterlibatan pihak lain dalam mengubah arti atau pengucapan. Hal ini karena tujuan penjagaan dan pemeliharaan Allah harus diwujudkan dalam kenyataan dimana wahyu tersebut mencapai tujuannya, yaitu manusia. Jika rasul tidak terpelihara dengan cara apapun, sehingga wahyu Allah tidak akan mencapai umat manusia, atau sampai tetapi terdapat perubahan, maka perlindungan wahyu akan gagal. Ini adalah peristiwa yang mustahil, karena yang

menyampaikan wahyu adalah para malaikat yang ditugaskan oleh Allah SWT. (Shihab, 2002).

Berdasarkan Al-Quran surat Al-Jin ayat 28 tersebut menjelaskan bahwa para rasul menyampaikan pesan dari Allah kepada umatnya berupa wahyu dan ilmu. Allah SWT menghitung segala sesuatu, termasuk wahyu yang disampaikan melalui malaikat jibril. Penjagaan keamanan wahyu yang disampaikan tersebut sudah terjamin keamanannya tanpa ada campur tangan pihak lain yang menyebabkan perubahan arti atau makna. Serta wahyu nang disampaikan tidak akan berubah, bertambah, ataupun dikurang.

Hal tersebut juga terjadi pada pengiriman pesan rahasia dari pengirim kepada penerima pesan menggunakan algoritma kriptografi dengan perhitungan matematika. Algoritma kriptografi bertujuan untuk menjaga keamanan pesan asli dengan mengubah susunan huruf menjadi urutan acak, sehingga tidak ada seorang pun yang bisa mengetahui pesan tersebut. Dan penerima dapat mendapatkan pesan tanpa ada perubahan isi pesan.

BAB III

PEMBAHASAN

Pada bab ini akan menjelaskan penggabungan dua algoritma beserta pembuktiannya untuk proses enkripsi dan dekripsi, selain itu akan dilakukan penghitungan secara matematis dan implementasi enkripsi dan dekripsi menggunakan bahasa pemrograman *Python.* Urutan algoritma yang digunakan pada proses enkripsi adalah algoritma *Vigenere Cipher* kemudian dilanjutkan dengan algoritma *Bifid Cipher*. Dan urutan algoritma yang digunakan pada proses dekripsi adalah kebalikan dari proses enkripsi yaitu algoritma *Bifid Cipher* kemudian dilanjutkan dengan algoritma *Vigenere Cipher*. Untuk algoritma *Bifid Cipher* disini akan dilakukan proses enkripsi dan dekripsi menggunakan papan kunci berukuran 6×6, hal ini bertujuan untuk mempermudah dalam pembuatan program enkripsi dan dekripsi menggunakan *Python*. Dengan papan kunci berukuran 6×6, penulisan huruf I dan J pada tabel dapat diletakkan pada posisi yang berbeda tempat. Sebagai pelengkap agar dapat menjadi papan kunci berbentuk persegi, dilakukan penambahan angka 0 sampai dengan 9.

3.1 Penggabungan Algotitma *Vigenere Cipher* dan Algoritma *Bifid Cipher* Membentuk Algoritma Super Enkripsi

3.1.1 Menggabungkan Dua Algoritma *Vigenere Cipher* dan *Bifid Cipher*

1) Proses Enkripsi

1. Hal pertama yang dilakukan adalah membuat *plaintext* (teks asli) yang akan dikirim kepada penerima pesan. Kemudian menentukan kunci

yang digunakan untuk proses enkripsi pesan. Selanjutnya akan dilakukan proses enkripsi pertama menggunakan algoritma *Vigenere Cipher* dengan langkah-langkah sebagai berikut:

2. Menyusun *plaintext* dan kunci kedalam tabel.

   Hal tersebut dilakukan agar memudahkan urutan *plaintext* yang berpasangan dengan urutan kunci tersusun secara berurutan. Tabel yang digunakan adalah sebagai berikut:

Tabel 3.1 Urutan Susunan Karakter *Plaintext* Dan Kunci Algoritma *Vigenere Cipher*

i	*Plaintext*	Kunci
1	P_1	K_1
2	P_2	K_2
3	P_3	K_3
4	P_4	⋮
5	P_5	K_n
6	P_6	K_1
7	P_7	K_2
8	P_8	K_3
9	P_9	⋮
10	P_{10}	K_n
11	P_{11}	K_1
12	P_n	K_2

3. Menghitung hasil *ciphertext* secara matematis menggunakan rumus enkripsi pada persamaan (2.22).

   Setelah *plaintext* dan kunci diletakkan secara berurutan pada tabel kemudian dilakukan penghitungan menggunakan rumus enkripsi untuk memperoleh *ciphertext* yang dibutuhkan.

4. Memperoleh *ciphertext* dari hasil enkripsi menggunakan algoritma *Vigenere Cipher*.

Setelah enkripsi pertama menggunakan algoritma *Vigenere Cipher* selesai dan diperoleh *ciphertext* dari proses tersebut, selanjutnya dilakukan proses enkripsi yang kedua menggunakan algoritma *Bifid Cipher* dengan langkah-langkah sebagai berikut:

5. *Ciphertext* tersebut selanjutnya menjadi *plaintext* untuk algoritma *Bifid Cipher*.

   Karena *Bifid Cipher* merupakan algoritma untuk proses enkripsi yang kedua, maka untuk *plaintext* (teks asli) diperoleh dari *ciphertext* hasil enkripsi algoritma pertama yaitu *Vigenere Cipher*.

6. Membuat Tabel 6×6 (6 baris dan 6 kolom). Yang merupakan papan kunci untuk algoritma *Bifid Cipher*. Tabel yang digunakan adalah sebagai berikut:

Tabel 3.2 Papan Kunci 6×6 *Bifid Cipher*

	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

7. Menginput huruf-huruf alfabet kedalam tabel.

Tabel 3.3 *Input* Alfabet Pada Papan Kunci 6×6 Untuk Proses Enkripsi Algoritma *Bifid Cipher*

	1	2	3	4	5	6
1	A	B	C	D	E	F
2	G	H	I	J	K	L
3	M	N	O	P	Q	R
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

Dalam pengisian tabel diawali dengan menginput kunci dengan menghilangkan huruf yang berulang, kemudian selanjutnya menginput huruf-huruf alfabet yang belum termasuk pada kunci.

8. Mengidentifikasi posisi tiap huruf *plaintext* pada tabel (koordinat baris dan kolom) dan membuatnya menjadi bigram (dipisahkan dua baris).

   Untuk memudahkan urutan posisi huruf maka digunakan tabel sebagai berikut:

Tabel 3.4 Koordinat Posisi Huruf *Plaintext* Untuk Proses Enkripsi

Plaintext	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_n
Baris	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_n
Kolom	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_n

9. Menyusun urutan koordinat tabel menjadi 1 baris.

   Untuk urutannya diawali dengan koordinat baris dari karakter pertama sampai terakhir kemudian dilanjutkan dengan koordinat kolom dari karakter pertama sampai terakhir.

10. Membuat pasangan koordinat tabel yang baru.

    Pembuatan pasangan koordinat baru dengan cara memasangkan angka yang berdekatan masing-masing satu pasang (dua angka).

11. Mencari huruf-huruf dari tabel dengan urutan koordinat yang baru.

    Untuk mempermudah mendapatkan urutan posisi huruf maka digunakan tabel sebagai berikut:

Tabel 3.5 Koordinat Posisi Huruf *Ciphertext* Untuk Proses Enkripsi

Baris	r_1	r_3	r_5	r_7	r_9	r_n	s_2	s_4	s_6	s_8	s_{10}
Kolom	r_2	r_4	r_6	r_8	r_{10}	s_1	s_3	s_5	s_7	s_9	s_n
Ciphertext	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_n

12. Memperoleh *ciphertext* akhir dari hasil enkripsi menggunakan gabungan algoritma *Vigenere Cipher* dan *Bifid Cipher*.

2) Proses Dekripsi

1. Pertama dapatkan *cipertext* (teks kode) yang didapatkan dari pengirim pesan. Kemudian mengetahui kunci yang digunakan untuk proses dekripsi pesan yang juga diperoleh dari pengirim pesan. Selanjutnya akan dilakukan proses dekripsi pertama menggunakan algoritma *Bifid Cipher* dengan langkah-langkah sebagai berikut:
2. Membuat Tabel 6×6 (6 baris dan 6 kolom). Yang merupakan papan kunci untuk algoritma *Bifid Cipher* seperti pada Tabel 3.2.

3. Menginput huruf-huruf alfabet kedalam tabel.

Tabel 3.6 *Input* Alfabet Pada Papan Kunci 6 × 6 Untuk Proses Dekripsi Algoritma *Bifid Cipher*

	1	2	3	4	5	6
1	A	B	C	D	E	F
2	G	H	I	J	K	L
3	M	N	O	P	Q	R
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

Dalam pengisian tabel dilakukan dengan cara yang sama, yaitu diawali dengan menginput kunci dengan menghilangkan huruf yang berulang, kemudian selanjutnya menginput huruf-huruf alfabet yang belum termasuk pada kunci. Tabel yang digunakan untuk mengenkripsi pesan harus sama dengan tabel yang digunakan untuk mendekripsi pesan.

4. Mengidentifikasi posisi tiap huruf c*iphertext* pada tabel (koordinat baris dan kolom) dan membuatnya menjadi bigram (dipisahkan dua baris). Untuk memudahkan urutan posisi huruf maka digunakan tabel sebagai berikut:

Tabel 3.7 Koordinat Posisi Huruf *Ciphertext* Untuk Proses Dekripsi

Ciphertext	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_n
Baris	r_1	r_3	r_5	r_7	r_9	r_n	s_2	s_4	s_6	s_8	s_{10}
Kolom	r_2	r_4	r_6	r_8	r_{10}	s_1	s_3	s_5	s_7	s_9	s_n

5. Menyusun urutan pasangan koordinat tabel menjadi 1 baris.

Untuk urutannya berbeda dengan urutan pada proses enkripsi. Pada proses dekripsi susunan 1 baris ini dilakukan dengan urutan pasangan

baris dan kolom untuk setiap karakter huruf. (Misal: baris-kolom-baris-kolom-dst)

6. Mengubah urutan pasangan koordinat 1 baris menjadi 2 bagian (2 baris sama panjang).

   Dalam mengubah urutan angka dari 1 baris menjadi 2 bagian sama panjang ini bertujuan untuk mengubah koordinat posisi setiap karakter huruf *ciphertext*.

7. Membuat pasangan koordinat tabel yang baru.

   Pembuatan pasangan koordinat baru dengan cara memasangkan angka yang posisinya sejajar atas bawah.

8. Mencari huruf-huruf dari tabel dengan urutan koordinat yang baru.

   Untuk mempermudah mendapatkan urutan posisi huruf maka digunakan tabel sebagai berikut:

Tabel 3.8 Koordinat Posisi Huruf *Plaintext* Untuk Proses Dekripsi

Baris	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_n
Kolom	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_n
Plaintext	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_n

9. Memperoleh *plaintext* dari hasil dekripsi menggunakan algoritma *Bifid Cipher*.

Setelah dekripsi pertama menggunakan algoritma *Bifid Cipher* selesai dan diperoleh *plaintext* dari proses tersebut, selanjutnya dilakukan proses dekripsi yang kedua menggunakan algoritma *Vigenere Cipher* dengan langkah-langkah sebagai berikut:

10. *Plaintext* tersebut selanjutnya menjadi c*iphertext* untuk algoritma *Vigenere Cipher*.

    Karena *Vigenere Cipher* merupakan algoritma untuk proses dekripsi yang kedua, maka untuk *ciphertext* (teks kode) diperoleh dari *plaintext* hasil dekripsi algoritma pertama yaitu *Bifid Cipher*.

11. Menyusun c*iphertext* dan kunci kedalam tabel.

    Hal tersebut dilakukan agar memudahkan urutan *ciphertext* yang berpasangan dengan urutan kunci tersusun secara berurutan. Kunci yang digunakan sama dengan proses enkripsi pada algoritma sebelumnya, tetapi untuk penghapusan pengulangan huruf tidak dilakukan. Tabel yang digunakan adalah sebagai berikut:

Tabel 3.9 Urutan Susunan Karakter *Ciphertext* Dan Kunci Algoritma *Vigenere Cipher*

i	*Ciphertext*	Kunci
1	C_1	K_1
2	C_2	K_2
3	C_3	K_3
4	C_4	⋮
5	C_5	K_n
6	C_6	K_1
7	C_7	K_2
8	C_8	K_3
9	C_9	⋮
10	C_{10}	K_n
11	C_{11}	K_1
12	C_n	K_2

12. Menghitung hasil *plaintext* secara matematis menggunakan rumus dekripsi pada persamaan (2.23).

Setelah *ciphertext* dan kunci diletakkan secara berurutan pada tabel kemudian dilakukan penghitungan menggunakan rumus dekripsi untuk memperoleh *plaintext* yang dibutuhkan.

13. Memperoleh *plaintext* dari hasil dekripsi menggunakan gabungan algoritma *Vigenere Cipher* dan *Bifid Cipher*.

3.1.2 Membuktikan Penggabungan Dua Algoritma *Vigenere Cipher* dan *Bifid Cipher* Memenuhi Algoritma Enkripsi

1) Proses Enkripsi

Algoritma *Vigenere Cipher*

$C_i = (P_i + K_i)\ mod\ 26$	Rumus enkripsi
$(P_i + K_i) \equiv C_i\ mod\ 26$	Definisi kongruensi
$26\ \vert (P_i + K_i) - \ C_i$	Definisi kongruensi
$(P_i + K_i) - C_i = 26 \cdot t$	Definisi kongruensi
$-C_i + (K_i + P_i) = 26 \cdot t$	Sifat komutatif
$(-C_i + K_i) + P_i = 26 \cdot t$	Sifat asosiatif
$(C_i - K_i) - P_i = 26 \cdot (-t)$	Kedua ruas dikalikan negatif
$(\ C_i - K_i) \equiv P_i\ mod\ 26$	Definisi kongruensi
$P_i = (\ C_i - K_i)\ mod\ 26$	Aritmatika modulo

Algoritma *Bifid Cipher*

Bentuk umum *plaintext* pada algoritma *Bifid Cipher*:

$$P_i(r_i, s_i) \tag{3.1}$$

Identifikasi posisi tiap huruf *plaintext* seperti pada Tabel 3.4.

Menyusun karakter baris dan kolom dalam 1 baris:

$r_1\ r_2\ r_3\ r_4\ r_5\ r_6\ r_7\ r_8\ r_9\ r_{10}\ r_n\ s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ s_7\ s_8\ s_9\ s_{10}\ s_n$

Memasangkan tiap karakter baris dan kolom yang berdekatan:

$r_1r_2 \;\; r_3r_4 \;\; r_5r_6 \;\; r_7r_8 \;\; r_9r_{10} \;\; r_ns_1 \;\; s_2s_3 \;\; s_4s_5 \;\; s_6s_7 \;\; s_8s_9 \;\; s_{10}s_n$

Identifikasi posisi tiap huruf yang akan menjadi *ciphertext* seperti pada Tabel 3.5.

Diperoleh *ciphertext*:

$$C_1(r_1, r_2), C_2(r_3, r_4), C_3(r_5, r_6), C_4(r_7, r_8), C_5(r_9, r_{10}), C_6(r_n, s_1),$$
$$C_7(s_2, s_3), C_8(s_4, s_5),, C_9(s_6, s_7), C_{10}(s_8, s_9), C_n(s_{10}, s_n) \quad (3.2)$$

2) Proses Dekripsi

Algoritma *Bifid Cipher*

Kita gunakan bentuk *ciphertext* pada pembuktian proses enkripsi pada persamaan (3.2).

Identifikasi posisi tiap huruf *ciphertext* seperti pada Tabel 3.7.

Menyusun karakter baris dan kolom dalam 1 baris (dengan urutan baris-kolom-baris-kolom-dst):

$r_1 \;\; r_2 \;\; r_3 \;\; r_4 \;\; r_5 \;\; r_6 \;\; r_7 \;\; r_8 \;\; r_9 \;\; r_{10} \;\; r_n \;\; s_1 \;\; s_2 \;\; s_3 \;\; s_4 \;\; s_5 \;\; s_6 \;\; s_7 \;\; s_8 \;\; s_9 \;\; s_{10} \;\; s_n$

Mengubah urutan pasangan koordinat 1 baris menjadi 2 bagian (dibagi menjadi 2 baris sama panjang).

$r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5 \quad r_6 \quad r_7 \quad r_8 \quad r_9 \quad r_{10} \quad r_n$

$s_1 \quad s_2 \quad s_3 \quad s_4 \quad s_5 \quad s_6 \quad s_7 \quad s_8 \quad s_9 \quad s_{10} \quad s_n$

Identifikasi posisi tiap huruf yang akan menjadi *plaintext* seperti pada Tabel 3.8.

Diperoleh *plaintext*:

$$P_1(r_1, s_1), P_2(r_2, s_2), P_3(r_3, s_3), P_4(r_4, s_4), P_5(r_5, s_5), P_6(r_6, s_6),$$
$$P_7(r_7, s_7), P_8(r_8, s_8), P_9(r_9, s_9), P_{10}(r_{10}, s_{10}), P_n(r_n, s_n) \quad (3.3)$$

Dari *plaintext* di atas didapatkan persamaan (3.3), sehingga bentuk umum *plainteks* pada algoritma *Bifid Cipher* sama seperti pada persamaan (3.1).

Algoritma *Vigenere Cipher*

$P_i = (C_i - K_i)\ mod\ 26$	Rumus dekripsi
$(C_i - K_i)\ \equiv P_i\ mod\ 26$	Definisi kongruensi
$26 \mid (C_i - K_i) - P_i$	Definisi kongruensi
$(C_i - K_i) - P_i = 26 \cdot t$	Definisi kongruensi
$-P_i + (-K_i + C_i) = 26 \cdot t$	Sifat komutatif
$\left(-P_i + (-K_i)\right) + C_i = 26 \cdot t$	Sifat asosiatif
$(P_i + K_i) -\ C_i = 26 \cdot (-t)$	Kedua ruas dikalikan negatif
$P_i + K_i \equiv C_i\ mod\ 26$	Definisi kongruensi
$C_i = (P_i + K_i)\ mod\ 26$	Aritmatika modulo

Singkatnya, untuk proses enkripsi dan dekripsi kita dapat membuktikannya secara langsung menggunakan kajian sifat-sifat dan jenis-jenis fungsi. Kedua algoritma *Vigenere Cipher* dan *Bifid Cipher* memiliki sifat bijektif, dimana fungsi bijektif memuat fungsi injektif dan surjektif yang juga harus terpenuhi. Seperti yang kita ketahui dari beberapa kajian dan literatur penelitian, kedua algoritma tersebut telah dinyatakan serta diakui sebagai algoritma kriptografi dalam proses enkripsi dan dekripsi. Fungsi injektif menggambarkan bahwa untuk setiap karakter *plaintext* sudah dipastikan mendapatkan satu hasil karkter *ciphertext*. Dan untuk fungsi surjektif menggambarkan bahwa tidak ada karakter *ciphertext* yang dihasilkan tanpa adanya *plaintext*.

Untuk penggabungan 2 algoritma yang disebut dengan super enkripsi dapat dibuktikan menggunakan komposisi dua fungsi. Jika kedua fungsi bersifat bijektif, misalkan algoritma *Vigenere Cipher* merupakan fungsi f dan algoritma *Bifid Cipher* merupakan fungsi g, maka komposisi kedua fungsi tersebut dapat ditulis $(g \circ f\,)(x) = g\big(f(x)\big)$. Yaitu dengan mengoperasikan fungsi yang pertama (fungsi f) terlebih dahulu yakni melakukan enkripsi algoritma *Vigenere Cipher*, kemudian dilanjutkan pengoperasian fungsi yang kedua (fungsi g) dengan melakukan enkripsi pada hasil *ciphertext* menggunakan algoritma *Bifid Cipher*. Sehingga hasil penggabungan dua algoritma tersebut juga bersifat bijektif, seperti yang dijelaskan Teorema 2.3 pada bab sebelumnya.

3.1.3 Menghitung Proses Enkripsi Dekripsi Menggunakan Cara Matematis

1) Proses Enkripsi

Plaintext : SEMANGAT SKRIPSI

Kunci : BELAJAR

Agar urutan karakter *plaintext* dan kunci tersusun berurutan, maka digunakan tabel sebagai berikut:

Tabel 3.10 Contoh Urutan Susunan Karakter *Plaintext* Dan Kunci Algoritma *Vigenere Cipher*

i	*Plaintext*	Kunci
1	S	B
2	E	E
3	M	L
4	A	A
5	N	J

6	G	A
7	A	R
8	T	B
9	S	E
10	K	L
11	R	A
12	I	J
13	P	A
14	S	R
15	I	B

Jika dihitung secara matematis menggunakan rumus pada persamaan (2.22):

maka,

$$C_1 = (P_1 + K_1)\ mod\ 26$$
$$= (18 + 1)\ mod\ 26$$
$$= 19 : T$$
$$C_2 = (P_2 + K_2)\ mod\ 26$$
$$= (4 + 4)\ mod\ 26$$
$$= 8 : I$$
$$C_3 = (P_3 + K_3)\ mod\ 26$$
$$= (12 + 11)\ mod\ 26$$
$$= 23 : X$$
$$C_4 = (P_4 + K_4)\ mod\ 26$$
$$= (0 + 0)\ mod\ 26$$
$$= 0 : A$$

$$C_5 = (P_5 + K_5)\ mod\ 26$$
$$= (13 + 9)\ mod\ 26$$
$$= 22 : W$$

$$C_6 = (P_6 + K_6)\ mod\ 26$$
$$= (6 + 0)\ mod\ 26$$
$$= 6 : G$$

$$C_7 = (P_7 + K_7)\ mod\ 26$$
$$= (0 + 17)\ mod\ 26$$
$$= 17 : R$$

$$C_8 = (P_8 + K_8)\ mod\ 26$$
$$= (19 + 1)\ mod\ 26$$
$$= 20 : U$$

$$C_9 = (P_9 + K_9)\ mod\ 26$$
$$= (18 + 4)\ mod\ 26$$
$$= 22 : W$$

$$C_{10} = (P_{10} + K_{10})\ mod\ 26$$
$$= (10 + 11)\ mod\ 26$$
$$= 21 : V$$

$$C_{11} = (P_{11} + K_{11})\ mod\ 26$$
$$= (17 + 0)\ mod\ 26$$
$$= 17 : R$$

$$C_{12} = (P_{12} + K_{12})\ mod\ 26$$
$$= (8 + 9)\ mod\ 26$$
$$= 17 : R$$

$$C_{13} = (P_{13} + K_{13})\ mod\ 26$$
$$= (15 + 0)\ mod\ 26$$
$$= 15 : P$$
$$C_{14} = (P_{14} + K_{14})\ mod\ 26$$
$$= (18 + 17)\ mod\ 26$$
$$= 9 : J$$
$$C_{15} = (P_{15} + K_{15})\ mod\ 26$$
$$= (8 + 1)\ mod\ 26$$
$$= 9 : J$$

Diperoleh *ciphertext* : TIXAWGRUWVRRPJJ

Selanjutnya dilakukan enkripsi menggunakan *Bifid Cipher*, yang merupakan proses enkripsi algoritma kedua setelah *Vigenere Cipher*, sehingga *plaintext* diperoleh dari *ciphertext* hasil enkripsi *Vigenere Cipher*.

Plaintext : TIXAWGRUWVRRPJJ

Kunci : BELAJAR, maka dengan menghilangkan huruf yang berulang diperoleh “BELAJR”

Kemudian kita susun kunci pada papan kunci, dan mengisi kotak kosong dengan huruf alfabet yang belum termasuk dalam kunci. Seperti pada Tabel 3.11 berikut.

Tabel 3.11 Contoh Papan Kunci 6 × 6 *Bifid Cipher*

	1	2	3	4	5	6
1	B	E	L	A	J	R
2	C	D	F	G	H	I
3	K	M	N	O	P	Q
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

Maka diperoleh identifikasi:

Tabel 3.12 Contoh Koordinat Posisi Huruf *Plaintext* Untuk Proses Enkripsi

Plaintext	T	I	X	A	W	G	R	U	W	V	R	R	P	J	J
Baris	4	2	4	1	4	2	1	4	4	4	1	1	3	1	1
Kolom	2	6	6	4	5	4	6	3	5	4	6	6	5	5	5

Selanjutnya menyusun koordinat tabel menjadi 1 baris dengan urutan baris kemudian kolom, diperoleh:

4 2 4 1 4 2 1 4 4 4 1 1 3 1 1 2 6 6 4 5 4 6 3 5 4 6 6 5 5 5

Setelah itu pasangkan 2 angka sehingga menjadi koordinat tabel yang baru, menjadi:

42 41 42 14 44 11 31 12 66 45 46 35 46 65 55

Dari pasangan koordinat tabel diatas, diperoleh hasil identifikasi urutan posisi huruf yang baru sebagai berikut:

Tabel 3.13 Contoh Koordinat Posisi Huruf *Ciphertext* Untuk Proses Enkripsi

Baris	4	4	4	1	4	1	3	1	6	4	4	3	4	6	5
Kolom	2	1	2	4	4	1	1	2	6	5	6	5	6	5	5
Ciphertext	T	S	T	A	V	B	K	E	9	W	X	P	X	8	2

Jadi, dari enkripsi penggabungan algoritma *Vigenere Cipher* kemudian dilanjutkan dengan algoritma *Bifid Cipher* diperoleh

Ciphertext akhir : TSTAVBKE9WXPX82

2) Proses Dekripsi

Ciphertext : TSTAVBKE9WXPX82

Kunci : BELAJAR, maka dengan menghilangkan huruf yang berulang diperoleh "BELAJR"

Kemudian kita susun kunci pada papan kunci, dan mengisi kotak kosong dengan huruf alfabet yang belum termasuk dalam kunci :

Untuk papan kunci dekripsi sama dengan papan kunci enkripsi, karena kata/kalimat kuncinya sama.

Tabel 3.14 Contoh *Input* Alfabet Pada Papan Kunci 6 × 6 Untuk Proses Dekripsi Algoritma *Bifid Cipher*

	1	2	3	4	5	6
1	B	E	L	A	J	R
2	C	D	F	G	H	I
3	K	M	N	O	P	Q
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

Maka diperoleh identifikasi:

Tabel 3.15 Contoh Koordinat Posisi Huruf *Ciphertext* Untuk Proses Dekripsi

Ciphertext	T	S	T	A	V	B	K	E	9	W	X	P	X	8	2
Baris	4	4	4	1	4	1	3	1	6	4	4	3	4	6	5
Kolom	2	1	2	4	4	1	1	2	6	5	6	5	6	5	5

Selanjutnya menyusun pasangan koordinat karakter tabel menjadi 1 baris, diperoleh:

42 41 42 14 44 11 31 12 66 45 46 35 46 65 55

Setelah itu mengubah urutan pasangan koordinat diatas menjadi 2 bagian

4 2 4 1 4 2 1 4 4 4 1 1 3 1 1

2 6 6 4 5 4 6 3 5 4 6 6 5 5 5

Dari pasangan koordinat tabel diatas, diperoleh hasil identifikasi sebagai berikut:

Tabel 3.16 Contoh Koordinat Posisi Huruf *Plaintext* Untuk Proses Dekripsi

Baris	4	2	4	1	4	2	1	4	4	4	1	1	3	1	1
Kolom	2	6	6	4	5	4	6	3	5	4	6	6	5	5	5
Plaintext	T	I	X	A	W	G	R	U	W	V	R	R	P	J	J

Diperoleh *plaintext* : TIXAWGRUWVRRPJJ

Proses dekripsi *Vigenere Cipher* ini adalah proses dekripsi algoritma kedua setelah *Bifid Cipher*, sehingga *ciphertext* diperoleh dari *plaintext* hasil dekripsi *Bifid Cipher*.

Cipherteks : TIXAWGRUWVRRPJJ

Kunci : BELAJAR

Agar urutan karakter *ciphertext* dan kunci tersusun berurutan, maka digunakan Tabel 3.17 sebagai berikut:

Tabel 3.17 Contoh Urutan Susunan Karakter *Ciphertext* Dan Kunci Algoritma *Vigenere Cipher*

i	*Ciphertext*	Kunci
1	T	B
2	I	E
3	X	L
4	A	A
5	W	J
6	G	A
7	R	R
8	U	B
9	W	E
10	V	L
11	R	A
12	R	J
13	P	A
14	J	R
15	J	B

Jika dihitung secara matematis menggunakan rumus pada persamaan (2.23):

maka,

$$P_1 = (C_1 - K_1)\ mod\ 26$$
$$= (19 - 1)\ mod\ 26$$
$$= 18 : S$$
$$P_2 = (C_2 - K_2)\ mod\ 26$$
$$= (8 - 4)\ mod\ 26$$
$$= 4 : E$$
$$P_3 = (C_3 - K_3)\ mod\ 26$$
$$= (23 - 11)\ mod\ 26$$

$$= 12 : M$$

$$P_4 = (C_4 - K_4)\ mod\ 26$$

$$= (0 - 0)\ mod\ 26$$

$$= 0 : A$$

$$P_5 = (C_5 - K_5)\ mod\ 26$$

$$= (22 - 9)\ mod\ 26$$

$$= 13 : N$$

$$P_6 = (C_6 - K_6)\ mod\ 26$$

$$= (6 - 0)\ mod\ 26$$

$$= 6 : G$$

$$P_7 = (C_7 - K_7)\ mod\ 26$$

$$= (17 - 17)\ mod\ 26$$

$$= 0 : A$$

$$P_8 = (C_8 - K_8)\ mod\ 26$$

$$= (20 - 1)\ mod\ 26$$

$$= 19 : T$$

$$P_9 = (C_9 - K_9)\ mod\ 26$$

$$= (22 - 4)\ mod\ 26$$

$$= 18 : S$$

$$P_{10} = (C_{10} - K_{10})\ mod\ 26$$

$$= (21 - 11)\ mod\ 26$$

$$= 10 : K$$

$$P_{11} = (C_{11} - K_{11})\ mod\ 26$$

$$= (17 - 0)\ mod\ 26$$

$$= 17 : R$$

$$P_{12} = (C_{12} - K_{12}) \; mod \; 26$$

$$= (17 - 9) \; mod \; 26$$

$$= 8 : I$$

$$P_{13} = (C_{13} - K_{13}) \; mod \; 26$$

$$= (15 - 0) \; mod \; 26$$

$$= 15 : P$$

$$P_{14} = (C_{14} - K_{14}) \; mod \; 26$$

$$= (9 - 17) \; mod \; 26$$

$$= 18 : S$$

$$P_{15} = (C_{15} - K_{15}) \; mod \; 26$$

$$= (9 - 1) \; mod \; 26$$

$$= 8 : I$$

Jadi, dari dekripsi penggabungan algoritma *Bifid Cipher* kemudian dilanjutkan dengan algoritma *Vigenere Cipher* diperoleh

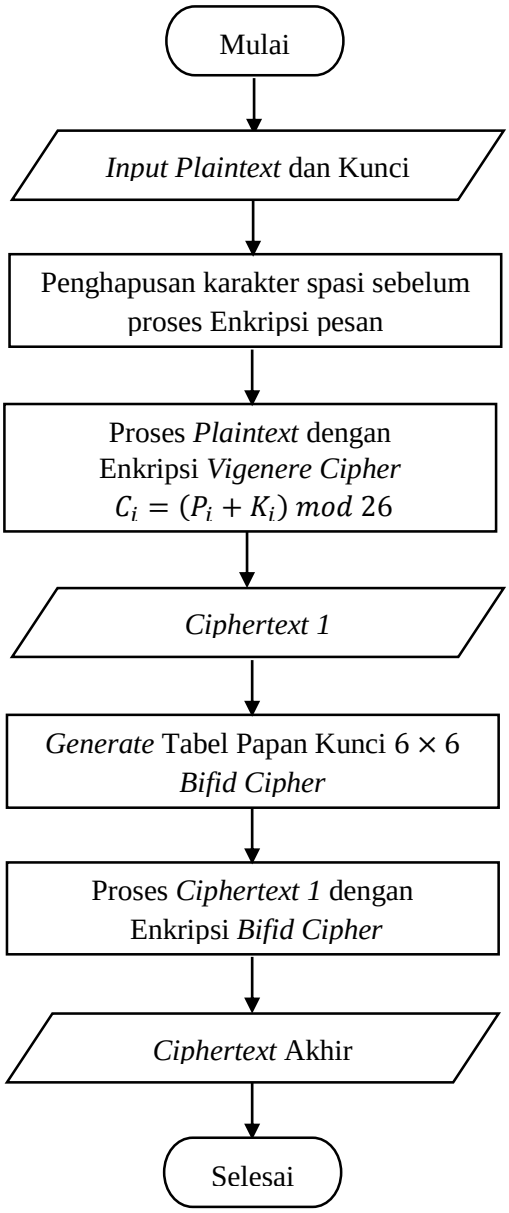
Plaintext akhir : SEMANGAT SKRIPSI

3.2 Implementasi Algoritma Super Enkripsi Menggunakan Pemrograman *Python*

Setelah menggabungkan dua algoritma dan melakukan penghitungan secara matematis, pada bagian ini akan dilakukan implementasi algoritma super enkripsi menggunakan *Python*. Algoritma yang berisi langkah-langkah dan hitungan matematis diubah kedalam bahasa pemrograman *Python* sehingga dapat menjalankan perintah enkripsi pesan secara otomatis. Untuk mempermudah dalam

pembuatan program super enkripsi maka langka-langkah tersebut disajikan dengan bentuk diagram alir (*flowchart)* terlebih dahulu. Berikut adalah penyajian *flowchart* super enkripsi menggunakan algoritma *Vigenere Cipher* dan *Bifid Cipher* untuk proses enkripsi dan dekripsi pesan.

1) Proses Enkripsi



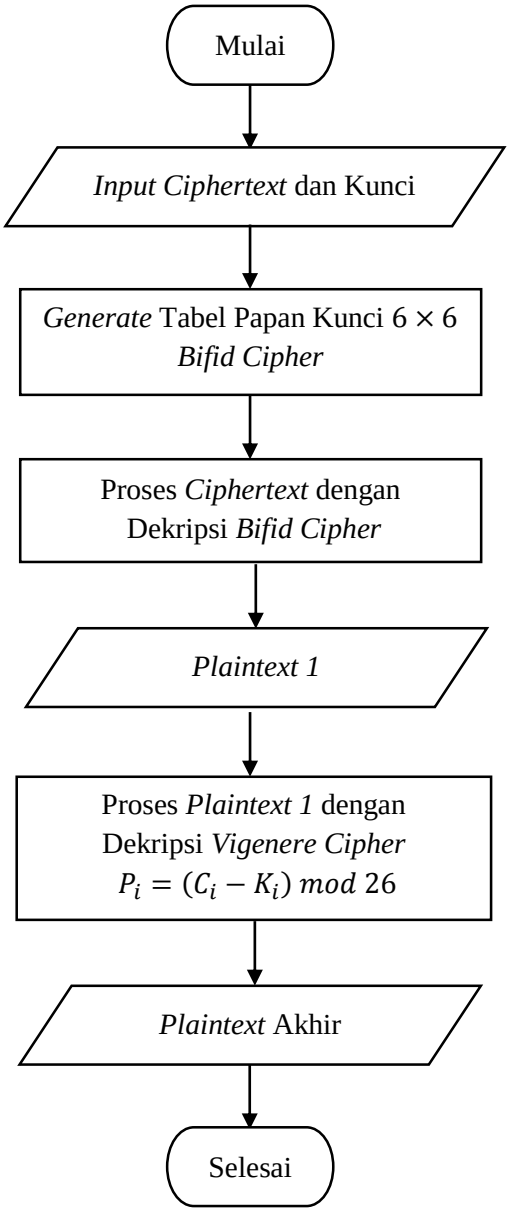
Gambar 3.1 Diagram Alir Proses Enkripsi Super Enkripsi

Pada Gambar 3.1 di atas menggambarkan alur proses enkripsi dari super enkripsi menggunakan algoritma *Vigenere Cipher* kemudian dilanjutkan dengan algoritma *Bifid Cipher*. Proses tersebut diawali dengan memasukkan teks awal (*plaintext*) dan kunci, sebelum melakukan proses enkripsi dilakukan penghapusan karakter spasi pada pesan teks awal (*plaintext*), hal ini dilakukan karena karakter spasi tidak termasuk ke dalam karakter pesan yang akan melalui proses enkripsi ataupun dekripsi. Kemudian *plaintext* diproses menggunakan algoritma pertama yaitu *Vigenere Cipher* menghasilkan *Ciphertext 1*. Selanjutnya membangun tabel papan kunci 6×6 dengan menginput huruf-huruf kunci, dilanjutkan dengan huruf alfabet yang belum termasuk dalam kunci untuk algoritma *Bifid Cipher*. Untuk proses selanjutnya melakukan enkripsi kedua pada *Ciphertext 1* menggunakan algoritma *Bifid Cipher* sehingga menghasilkan *Ciphertext* Akhir.

2) Proses Dekripsi

Pada Gambar 3.2 menggambarkan alur proses dekripsi dari super enkripsi menggunakan algoritma *Bifid Cipher* kemudian dilanjutkan dengan algoritma *Vigenere Cipher*. Proses tersebut diawali dengan memasukkan pesan rahasia (*ciphertext*) dan kunci. Karena untuk proses dekripsi adalah kebalikan dari proses enkripsi, maka algoritma pertama yang digunakan adalah *Bifid Cipher*. Setelah memasukkan kunci, maka selanjutnya membangun tabel papan kunci 6×6 dengan menginput huruf-huruf kunci, dilanjutkan dengan huruf alfabet yang belum termasuk dalam kunci untuk algoritma *Bifid Cipher*. Kemudian melakukan proses dekripsi yang pertama untuk *ciphertext* menggunakan *Bifid Cipher* menghasilkan *Plaintext 1*. Langkah selanjutnya melakukan proses dekripsi kedua

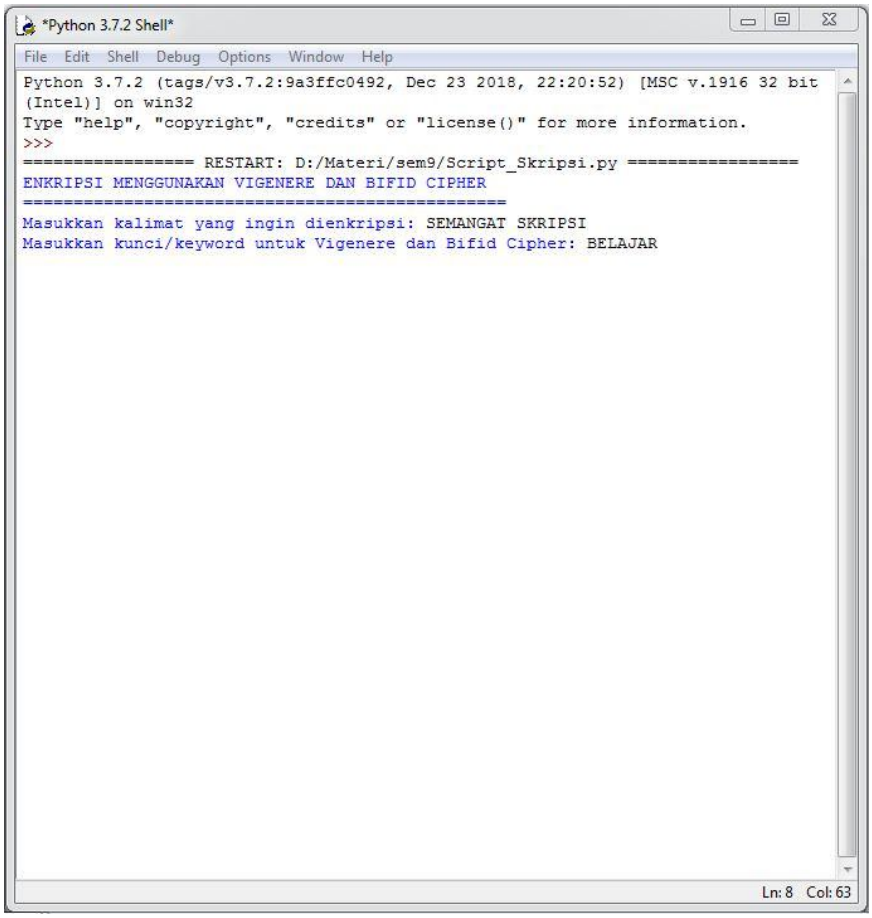
pada *Plaintext 1* menggunakan algoritma *Vigenere Cipher* sehingga menghasilkan *Plaintext* Akhir.



Gambar 3.2 Diagram Alir Proses Dekripsi Super Enkripsi

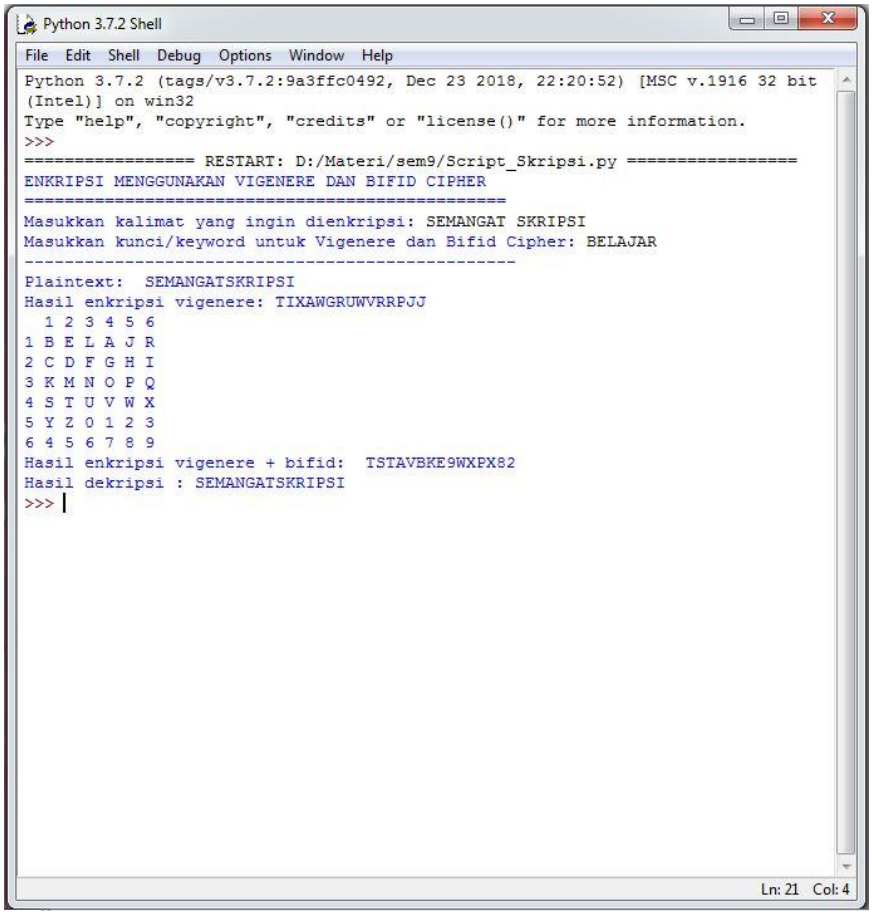
Kemudian mengimplementasikan rangkaian *flowchart* tersebut kedalam bahasa pemrograman *Python* (*script*). Pada pembuatan program disini dilakukan dengan mendefinisikan fungsi-fungsi kemudian pada langkah terakhir dilakukan

pemanggilan fungsi-fungsi *Python* tersebut. Sehingga ketika menjalankan program (*running*) diperoleh tampilan seperti Gambar 3.3.



Gambar 3.3 Hasil *Running* Program *Python*

Setelah menjalankan program *Python* akan muncul tampilan seperti Gambar 3.3. Pengguna dapat menginput pesan yang ingin dienkripsi, serta kunci yang digunakan untuk proses enkripsi dan dekripsi. Untuk kunci hanya menggunakan 1 susunan huruf/kata saja, karena dengan 1 jenis kunci dapat digunakan untuk algoritma *Vigenere Cipher* dan *Bifid Cipher*.



Gambar 3.4 Hasil Enkripsi dan Dekripsi Program *Python*

Untuk *output* yang dihasilkan setelah memasukkan kalimat yang ingin dienkripsi dan kunci, dapat dilihat dari Gambar 3.4 diatas menampilkan *plaintext* (pesan asli), dan hasil enkripsi algoritma pertama yaitu algoritma *Vigenere Cipher*. Pada *output* tersebut juga menampilkan papan kunci berbentuk persegi dengan ukuran 6×6 untuk algoritma *Bifid Cipher*. Kemudian pada bagian terakhir ditampilkan hasil enkripsi dari dua algoritma *Vigenere Cipher* dan *Bifid Cipher*. Untuk membuktikan bahwa penggabungan 2 algoritma tersebut memenuhi proses enkripsi, maka ditampilkan hasil dari proses dekripsi, dimana hasil dekripsi harus sama dengan kalimat yang diinput sebelum proses enkripsi.

Tabel 3.18 Uji Coba Proses Enkripsi Dan Dekripsi Menggunakan Pemrograman *Python*

No.	***Plaintext*** **(Teks asli)**	**Kunci**	***Ciphertext***	***Plaintext*** **(Hasil dekripsi dari *ciphertext*)**
1.	SEMANGAT SKRIPSI	BELAJAR	TSTAVBKE9WXPX82	SEMANGATSKRIPSI
2.	IJIN TIDAK MASUK	SAKIT	KIJCASCCILEISJ	IJINTIDAKMASUK
3.	JALAN JALAN DI SORE HARI	SENANG	SYNNSMKSRF5AWGM7SDP6	JALANJALANDISOREHARI
4.	MATA KULIAH KRIPTOGRAFI	CIPHER	KCMMPKNHKPWBWF62A5SAX	MATAKULIAHKRIPTOGRAFI
5.	JANGAN LUPA MAKAN	SEHAT	SREKKSQJFWP9ZDV	JANGANLUPAMAKAN
6.	MATEMATIKA	HITUNG	HOYDAKQFHV	MATEMATIKA
7.	HITUNG	MATEMATIKA	MIZPMG	HITUNG

Pada percobaan dalam Tabel 3.18, dengan berbagai macam masukan untuk pesan yang ingin dienkripsi (*plaintext*) dan juga kunci, dapat dihasilkan pesan rahasia (*ciphertext*) yang apabila dilakukan proses dekripsi dapat kembali ke bentuk pesan awal semula. Hal ini membuktikan bahwa proses enkripsi dan dekripsi pesan menggunakan bahasa pemrograman *Python* berhasil dilakukan.

BAB IV

PENUTUP

4.1 Kesimpulan

Berdasarkan rumusan masalah dan pembahasan diatas, dapat disimpulkan bahwa:

1. Berdasarkan pembuktian dan percobaan dengan contoh, penggabungan dua algoritma *Vigenere Cipher* dan *Bifid Cipher* memenuhi algoritma enkripsi sehingga dapat membentuk algoritma super enkripsi. Karena setelah melakukan proses enkripsi diperoleh *ciphertext* dari proses tersebut kemudian dilakukan proses dekripsi menggunakan *ciphertext* yang sama maka diperoleh *plaintext* (teks asli) yang sama dengan *plaintext* semula sebelum melalui proses enkripsi pesan. Proses enkripsi pesan yaitu menggunakan algoritma *Vigenere Cipher*, kemudian dilanjutkan menggunakan algoritma *Bifid Cipher* untuk proses enkripsi yang kedua. Dan pada proses dekripsi pesan yaitu menggunakan algoritma *Bifid Cipher*, kemudian dilanjutkan dengan algoritma *Vigenere Cipher* untuk proses dekripsi yang kedua. Penggabungan dua algorita ini menghasilkan keamanan pesan yang lebih terjaga.
2. Implementasi algoritma super enkripsi menggunakan pemrograman *Python* dapat dilakukan dengan menyusun proses penghitungan secara matematis kedalam bentuk susunan *flowchart*, kemudian diubah kedalam bahasa pemrograman *Python* sesuai dengan prosedur dan langkah-langkah proses enkripsi dan dekripsi pesan. Proses enkripsi dan dekripsi pada *Python* dilakukan dengan memanggil fungsi-fungsi yang memiliki perintah tertentu.

Sehingga penghitungan secara manual memiliki hasil yang sama dengan penghitungan menggunakan program *Python*.

4.2 Saran

Pada penelitian ini membahas tentang penggabungan dua algoritma *Vigenere Cipher* dan *Bifid Cipher* untuk membangun super enkripsi dengan proses enkripsi dan dekripsi, serta implementasi menggunakan bahasa pemrograman *Python*. Maka untuk penelitian selanjutnya disarankan untuk membangun algoritma super enkripsi dengan menggabungkan tiga, empat atau beberapa algoritma yang berbeda selain menggunakan *Vigenere Cipher* dan *Bifid Cipher*. Selain itu untuk implementasi dapat menggunakan bahasa pemrograman lain.

DAFTAR PUSTAKA

Al-Qur'an dan Terjemah.2007.*Departemen Agama RI*. Bandung: CV Penerbit *Dipenonegoro*

Ariyus, Doni .2006. *Kriptografi Keamanan Data dan Komunikasi*. Yogyakarta:Graha Ilmu

Ariyus, Dony.2008.*Pengantar Ilmu Kriptografi Teori Analisis dan Implementasi*.Yogyakarta:C.V Andi Offset.

Bartle, Robert G, dan Sherbert Donald R.2011.*Introduction to Real Analysis*.4th.USA:John Willey and Sons.

Enterprise, Jubilee.2019.*Python untuk Programmer Pemula*.Jakarta:PT.Elex Media Komputindo

Gallian, Joseph A.2013.*Contemporary Abstract Algebra*.8th.USA:Brooks/Cole Cengange Learning.

Girsang, Nardianti Dewi, dkk.2019.Kombinasi Algoritma Kriptografi Transposisi *Rail Fence Cipher* dan *Route Cipher*.*Prosiding Seminar Nasional Teknologi Informatika*.No.1.Vol.2.

Irawan, Wahyu Henky, dkk.2014.*Pengantar Teori Bilangan*.Malang:UIN-Maliki Press

Kusumawati, Ririen.2014.*Aljabar Linear dan Matriks*.Malang:UIN-Maliki Press

Manullang, Dewi Intan.2018. Perancangan Aplikasi Penyandian File Teks Dengan Algoritma Bifid Cipher.*Jurnal Pelita Informatika*.No.3.Vol.6.

Munir, Rinaldi.2019.*Kriptografi*.Bandung:Informatika Bandug

Pratama, Reditya Kiko Pandu, dan Fitri Latifah.2014. Implementasi Enkripsi Dekripsi Pesan Teks Menggunakan Model Julis Caesar Berbasis Object Oriented Programme.*Jurnal Tecno Nusa Mandiri*.No.1.Vol.11.

Quist, Aphetsi Kester.2013. *A Hybrid Cryptosystem Based on Vigenere Cipher and Columnar Transposition Cipher*. *International Journal of Advanced Tecnology & Engineering Research (IJATER)*.Vol.3.Issue.1.

Sadikin, Rifki.2012.*Kriptografi untuk Keamanan Jaringan*.Yogyakarta:C.V Andi Offset

Sahara, Helmi.2018.Implementasi Pengamanan Pesan Chatting Menggunakan Metode Vigenere Cipher dan Cipher Block Chainning.*Media Informasi Analisa dan Sistem(MEANS)*.No.2.Vol.3.

Shihab, M. Quraish.2002.*Tafsir Al-Misbah Pesan, Kesan, dan Keserasian Al-Qur'an*.Jakarta:Lentera Hati.

Susanti, Dian.2020.Analisis Modifikasi Metode Playfair Cipher Dalam Pengamanan Data Teks.*Indonesian Journal of Data and Science*.No.1.Vol.1.

Wulandari, Siska.2019.Pengamanan Pesan Teks E-Mail Menggunakan Metode Algoritma Bifid Dan Feedback Cipher.*Jurnal Riset Komputer (JURKOM)*.No.5.Vol.6.

LAMPIRAN

```
"""
   GENERATE TABEL POLYBIUS UNTUK BIFID CIPHER
"""
from typing import Text

import random

def rand(min, max):
   return int((max - min) * random.random() + min)

def listToString(s):
   str1 = ""
      for ele in s:
            str1 += ele
      return str1

def generate_table(key):
   alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'

   key = key.upper() + alphabet

   key_list = []
      for k in key:
         if k not in key_list:
                        key_list.append(k.upper())

   keyword = listToString(key_list)

   table = [[0] * 6 for row in range(6)]

   counter = 0
      for x in range(6):
            for y in range(6):
         table[x][y] = keyword[counter]
                  alphabet = alphabet.replace(table[x][y], '')
                  counter += 1
      return table
 def getStr(x, format='%02s'):
   return ''.join(format % i for i in x)
```

```
# Untuk mengubah indeks huruf dalam tabel menjadi huruf
def getStr(x, format='%02s'):
    return ''.join(format % i for i in x)

# Mencetak tabel polybius
def print_table(table):
    print('  ' + getStr(range(1, 7)))
        for row in range(0, len(table)):
        print(str(row + 1) + getStr(table[row]))

 def encrypt_polybius(table, words):
    cipher = ''
        for ch in words.upper():
        for row in range(len(table)):
                        if ch in table[row]:
                                x = str((table[row].index(ch) + 1))
                                y = str(row + 1)
                                cipher += y + x
        return cipher

 def decrypt_polybius(table, numbers):
    text = ''
        for index in range(0, len(numbers), 2):
                y = int(numbers[index]) - 1
                x = int(numbers[index + 1]) - 1
                text += table[y][x]
        return text

"""
    Vigenere Chiper
"""
def generateKey(string, key):
    key = list(key)
        if len(string) == len(key):
        return(key)
        else:
                        for i in range(len(string) - len(key)):
                                key.append(key[i % len(key)])
        return("" . join(key))

def encrypt_vigenere(string, key):
    cipher_text = []
        for i in range(len(string)):
```

```
            x = (ord(string[i]) + ord(key[i])) % 26
            x += ord('A')
            cipher_text.append(chr(x))
        return("" . join(cipher_text))

def decrypt_vigenere(cipher_text, key):
    orig_text = []
        for i in range(len(cipher_text)):
            x = (ord(cipher_text[i]) - ord(key[i])) % 26
            x += ord('A')
            orig_text.append(chr(x))
        return("" . join(orig_text))



"""
Bifid Cipher
"""
def encrypt_bifid(table, words):
        cipher_row, cipher_col = '', ''

        for ch in words.upper():
            for row in range(len(table)):
                if ch in table[row]:
                cipher_row += str(row + 1)
                cipher_col += str((table[row].index(ch) + 1))
        return decrypt_polybius(table, cipher_row + cipher_col)

def decrypt_bifid(table, text):
    numbers = ''
    text = encrypt_polybius(table, text)
    a, b = text[:len(text) // 2], text[len(text) // 2:]
        numbers = ''.join(a[i] + b[i] for i in range(len(a)))
        return decrypt_polybius(table, numbers)

 if __name__ == '__main__':
            print("ENKRIPSI MENGGUNAKAN VIGENERE DAN BIFID
            CIPHER")
            print("==============================================
            ")     user_input = input("Masukkan kalimat yang ingin dienkripsi: ")
            import re
            user_input = re.sub(r"\s+", "", user_input, flags=re.UNICODE)
```

```
cipher_keyword = input("Masukkan kunci/keyword untuk Vigenere dan Bifid Cipher: ")
                key = generateKey(user_input, cipher_keyword)
print("-------------------------------------------------")     print("Ciphertext: ", user_input)

vigenere_result = encrypt_vigenere(user_input, key)     print("Hasil enkripsi vigenere:", vigenere_result)

table = generate_table(cipher_keyword)
print_table(table)

bifid_result = encrypt_bifid(table, vigenere_result)     print("Hasil enkripsi vigenere + bifid: ", bifid_result)

final_result = decrypt_vigenere(decrypt_bifid(table, bifid_result), key)
print("Hasil dekripsi :", final_result)
```

RIWAYAT HIDUP

Laura Agustina, lahir di kabupaten Pasuruan pada tanggal 28 Juni 1998. Biasa dipanggil Laura. Tinggal di dusun Gondang RT 008 RW 002, desa Kepulungan, kecamatan Gempol, kabupaten Pasuruan. Anak pertama dari dua bersaudara dari pasangan Bapak Agus Salim dan Ibu Nurul Hidayati.

Pendidikan pertama dimulai di TK An-Nur, kemudian pada tahun 2005 menempuh sekolah dasar di SDN Kepulungan I dan lulus pada tahun 2011. Dilanjutkan menempuh pendidikan menengah pertama di SMPN 2 Gempol yang lulus pada tahun 2014. Kemudian melanjutkan pendidikan menengah atas di SMA Negeri 1 Pandaan yang lulus pada tahun 2017. Selanjutnya, pada tahun 2017 melanjutkan pendidikannya di Universitas Islam Negeri Maulana Malik Ibrahim Malang pada program studi Matematika.

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No. 50 Dinoyo Malang Telp/Fax.(0341)558933

BUKTI KONSULTASI SKRIPSI

Nama : Laura Agustina
NIM : 17610081
Fakultas/ Program Studi : Sains dan Teknologi/ Matematika
Judul Skripsi : Membangun Super Enkripsi dengan *Vigenere Cipher* dan *Bifid Cipher* Menggunakan Pemrograman *Python* untuk Mengamankan Pesan.
Pembimbing I : Dr. H. Imam Sujarwo, M.Pd
Pembimbing II : Muhammad Khudzaifah, M.Si

No	Tanggal	Hal	Tanda Tangan
1.	19 Maret 2021	Konsultasi BAB I & II	1.
2.	22 Maret 2021	Konsultasi Kajian Keagamaan BAB I	2.
3.	25 Maret 2021	Revisi BAB I & II	3.
4.	8 April 2021	Konsultasi BAB I, II & III	4.
5.	16 April 2021	Revisi BAB I, II & III	5.
6.	28 April 2021	Konsultasi Kajian Keagamaan BAB II	6.
7.	07 Mei 2021	ACC untuk diseminarkan	7.
8.	07 Mei 2021	ACC untuk diseminarkan	8.
9.	10 Juni 2021	Konsultasi BAB III	9.
10.	15 Juni 2021	Konsultasi BAB III	10.
11.	22 Juni 2021	Revisi BAB III & konsultasi BAB IV	11.
12.	27 September 2021	Konsultasi BAB IV, Abstrak, dan *Flowchart*	12.
13.	04 Oktober 2021	ACC keseluruhan untuk disidangkan	13.
14.	06 Oktober 2021	ACC keseluruhan untuk disidangkan	14.

Malang, 26 November 2021
Mengetahui,
Ketua Program Studi Matematika

Dr. Elly Susanti, M.Sc.
NIP. 19741129 200012 2 005