PENGKODEAN TEKS MENGGUNAKAN ALGORITMA *CIPHER FEEDBACK* (CFB) *BASE64* BERDASARKAN MODIFIKASI DERET LUCAS

SKRIPSI

Oleh

Ahmad Farhan Nafisena
221810101023

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS JEMBER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
JURUSAN MATEMATIKA
JEMBER
2026

PENGKODEAN TEKS MENGGUNAKAN ALGORITMA *CIPHER FEEDBACK* (CFB) *BASE64* BERDASARKAN MODIFIKASI DERET LUCAS

diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana pada Program Studi Matematika

SKRIPSI

Oleh

Ahmad Farhan Nafisena
221810101023

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS JEMBER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
JURUSAN MATEMATIKA
JEMBER
2026

PERSEMBAHAN

Dengan menyebut nama Allah Yang Maha Pengasih lagi Maha Penyayang serta memanjatkan puji dan syukur kehadirat Allah SWT atas segala rahmat, nikmat, dan karunia-Nya, penulis mempersembahkan skripsi ini kepada:

1. Kedua orang tua tercinta, yang selalu memberikan kasih sayang, doa, dukungan, semangat, serta pengorbanan tanpa henti untuk setiap langkah penulis. Terima kasih atas segala perjuangan, nasihat, dan ketulusan yang telah diberikan hingga penulis mampu menyelesaikan pendidikan ini. Semoga Allah SWT senantiasa melimpahkan kesehatan, kebahagiaan, keberkahan, serta membalas segala kebaikan Bapak dan Ibu dengan pahala yang berlipat ganda.
2. Keluarga tersayang, yang selalu memberikan semangat dan motivasi kepada penulis selama menempuh pendidikan hingga menyelesaikan skripsi ini.
3. Almamater tercinta, yaitu Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) Universitas Jember, MAN 1 Jember, MTsN 1 Jember, dan SDN Lengkong 01 yang telah menjadi tempat bagi penulis untuk menuntut ilmu, memperoleh pengalaman, serta menjadi bagian penting dalam perjalanan pendidikan penulis hingga saat ini.

MOTTO

"Fortis Fortuna Adiuvat"

"Keberuntungan berpihak pada mereka yang berani"

(Terence)

PERNYATAAN ORSINALITAS

Saya yang bertanda tangan di bawah ini :

Nama : Ahmad Farhan Nafisena

NIM : 221810101023

Menyatakan dengan sesungguhnya bahwa skripsi yang berjudul: *Pengkodean Teks Menggunakan Algoritma Cipher Feedback (CFB) Base64 Berdasarkan Modifikasi Deret Lucas* adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, dan belum pernah diajukan pada institusi manapun, serta bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 18 Juni 2026

Yang menyatakan,

Ahmad Farhan Nafisena

NIM 221810101023

HALAMAN PERSETUJUAN

Skripsi berjudul Pengkodean Teks Menggunakan Algoritma *Cipher Feedback (CFB) Base64* Berdasarkan Modifikasi Deret Lucas telah diuji dan disetujui pada

Hari :

Tanggal :

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Jember

Tim Penguji

Ketua,

Prof. Dr. Kiswara Agung Santoso, S.Si., M.Kom.
NIP 197209071998031003

Anggota I,

Dr. Agustina Pradjaningsih, S.Si., M.Si..
NIP 197108022000032009

Anggota II,

Ikhsanul Halikin, S.Pd., M.Si.
NIP 198610142014041001

ABSTRACT

Cryptography is a technique used to maintain data confidentiality and security. One widely used modern cryptographic method is the Cipher Feedback (CFB) mode, but the ciphertext produced by standard CFB still has limitations in randomness and security complexity. Therefore, this study modifies the CFB algorithm based on Base64 using a modified Lucas sequence to generate a more dynamic keystream and improve encryption security. The modification process is carried out by generating the keystream using vowel and non-vowel rules in the keyword, where non-vowel characers include consonants, digits, and selected punctuation marks. In contrast, the Initialization Vector (IV) is generated through sequential XOR operations on the keyword. Security analysis was conducted using the Character Change Level (TPK) by comparing the modified CFB method with the standard CFB method. The results show that the modified method produces more random ciphertext and lower TPK values. For plaintext "AKU" and keyword "OK", the standard CFB produced ciphertext "OOU" with a TPK value of 0.4286, while the modified method produced ciphertext "JCT" with a TPK value of 0.3333. This indicates that the relationship between plaintext and ciphertext becomes smaller, so the proposed modification improves encryption security compared to the standard CFB method.

Keywords: Cipher Feedback, Base64, Lucas sequence, cryptography, Character Change Level (TPK).

RINGKASAN

Pengkodean Teks Menggunakan Algoritma Cipher Feedback (CFB) Base64 Berdasarkan Modifikasi Deret Lucas: Ahmad Farhan Nafisena; 221810101023; 2026; 38 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Kriptografi merupakan salah satu teknik yang digunakan untuk menjaga kerahasiaan data dari pihak yang tidak berwenang. Salah satu metode kriptografi modern yang banyak digunakan adalah *Cipher Feedback* (CFB) karena mampu mengubah *block cipher* menjadi *stream cipher* melalui mekanisme umpan balik *ciphertext*. Namun, pembentukan *keystream* pada algoritma CFB standar masih menghasilkan pola *ciphertext* yang kurang bervariasi sehingga diperlukan modifikasi untuk meningkatkan tingkat keamanan pengkodean teks. Oleh karena itu, penelitian ini memodifikasi algoritma CFB berbasis *Base64* menggunakan deret Lucas pada proses pembentukan *keystream*.

Modifikasi dilakukan dengan membentuk keystream menggunakan aturan karakter vokal dan nonvokal pada *keyword* berdasarkan modifikasi deret Lucas, serta menentukan *Initialization Vector* (IV) menggunakan operasi XOR berurutan *keyword*. Proses enkripsi dan dekripsi dilakukan menggunakan karakter Base64 dan diimplementasikan melalui program MATLAB berbasis *Graphical User Interface* (GUI). Selain itu, penelitian ini juga melakukan pengujian keamanan menggunakan Tingkat Perubahan Karakter (TPK) dengan membandingkan metode CFB standar dan metode CFB berbasis modifikasi deret Lucas.

Hasil penelitian menunjukkan bahwa metode modifikasi menghasilkan *ciphertext* yang lebih bervariasi dan sulit diprediksi dibandingkan metode CFB standar. Pada *plaintext* "AKU" dengan *keyword* "OK", metode CFB standar menghasilkan *ciphertext* "OOU" dengan nilai TPK sebesar 0,4286, sedangkan metode modifikasi menghasilkan *ciphertext* "JCT" dengan nilai TPK sebesar 0,3333. Nilai TPK yang lebih kecil menunjukkan bahwa hubungan antara *plaintext* dan *ciphertext* menjadi lebih kecil atau tidak signifikan, sehingga tingkat keamanan algoritma meningkat dibandingkan metode CFB standar.

PRAKATA

Puji Syukur Alhamdulillah penulis panjatkan atas kehadirat Allah SWT atas rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul *Pengkodean Teks Menggunakan Algoritma Cipher Feedback (CFB) Base64 Berdasarkan Modifikasi Deret Lucas*. Skripsi ini diajukan untuk memenuhi salah satu persyaratan untuk memperoleh gelar Sarjana pada Program Studi Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan serta bimbingan berbagai pihak, oleh karena itu penulis mengucapkan terima kasih kepada:

1. Prof. Dr. Kiswara Agung Santoso, S.Si., M.Kom., selaku dosen pembimbing utama yang telah membimbing dan memberi arahan hingga skripsi ini terselesaikan.
2. Dr. Agustina Pradjaningsih, S.Si., M.Si., selaku dosen penguji I dan Ihsanul Halikin, S.Pd., M.Si., selaku dosen penguji II yang telah memberikan kritik, masukan, dan saran untuk penelitian ini.
3. Drs. Moh. Hasan, M.Sc., Ph.D., selaku dosen pembimbing akademik yang telah membimbing dan memberikan arahan sejak awal hingga akhir masa studi.
4. Sahabat-sahabat penulis, Ahmad Bayu Rifai, Ahmad Jabar Husaini, Mariyanto, Achmad Teguh Prasetya Wicaksono, Nugroho Adi Cahyo S. M. yang telah menemani dan memberikan dukungan.
5. Teman-teman ALGEBRA Program Studi Matematika angkatan 2022 atas bantuan dan kerjasama selama masa studi.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi perbaikan selanjutnya. Semoga skripsi ini dapat bermanfaat bagi pembaca dan pihak yang ingin mengembangkan.

Jember, Juni 2026

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1. PENDAHULUAN

Bab 1 membahas landasan awal penelitian yang berkaitan dengan penerapan algoritma *Cipher Feedback* (CFB) Base64 berdasarkan modifikasi deret Lucas untuk pengkodean teks. Uraian dalam bab ini meliputi latar belakang penelitian, rumusan masalah, tujuan penelitian, dan manfaat penelitian sebagai dasar pembahasan penelitian.

1.1 Latar Belakang

Kemajuan teknologi informasi membuat informasi dan data telah bertransformasi menjadi aset yang sangat bernilai. Hampir seluruh aspek kehidupan modern bergantung pada teknologi informasi dan komunikasi (Dola dkk, 2024). Hal ini juga disertai dengan meningkatnya intensitas serangan dan maraknya insiden kebocoran data yang berpotensi mengancam keamanan serta kerahasiaan informasi di berbagai sektor. Berdasarkan laporan Lanskap Keamanan Siber Indonesia 2024 yang diterbitkan oleh Badan Siber dan Sandi Negara (BSSN) sepanjang tahun 2024 tercatat 330.527.636 trafik anomali serangan siber di Indonesia, disertai 241 dugaan insiden kebocoran data yang berdampak pada berbagai sektor strategis. Kondisi ini menegaskan bahwa perlunya mekanisme pengamanan data yang kuat untuk menjaga kerahasiaan, integritas, dan keaslian informasi dari ancaman kebocoran maupun manipulasi. Penerapan teknik kriptografi dan pengkodean data menjadi salah satu opsi untuk menjaga kerahasiaan dan integritas informasi dari ancaman serangan siber dan kebocoran data.

Permasalahan kebocoran data dan keamanan informasi dapat diatasi menggunakan berbagai metode kriptografi, baik kriptografi klasik dan kriptografi modern. Metode kriptografi klasik seperti Caesar Cipher, Vigenere Cipher, dan Hill Cipher banyak digunakan sebagai dasar pengamanan data karena mampu menyandikan pesan menjadi bentuk yang sulit dipahami oleh pihak yang tidak berwenang. Gusti dkk. (2020) memodifikasi algoritma *Vigenere Cipher* menggunakan pola *flip* dan *shift row* pada *plaintext*, sehingga keamanan teks

meningkat ditunjukkan dengan nilai kolerasi yang semakin mendekati nilai 0. Rohim dkk. (2022) memodifikasi algoritma *Hill Cipher* menjadi *Hill Cipher Chain* dengan menggunakan kunci berbeda pada setiap proses enkripsi berdasarkan *ciphertext* sebelumnya sehingga proses dekripsi menjadi lebih kompleks dan keamanan data meningkat dibandingkan *Hill Cipher* biasa. Penelitian oleh Makhomah dkk. (2021) mengkombinasikan algoritma *Hill Cipher* dengan operasi XOR, dimana *plaintext* terlebih dahulu di-XOR dengan jumlah setiap kolom pada matriks kunci sehingga menghasilkan *plaintext* baru yang kemudian digunakan dalam proses enkripsi dan dekripsi *Hill Cipher*. Astuti dkk. (2019) memodifikasi kunci pada algoritma *Affine Cipher* menggunakan aturan barisan *Fibonacci* sehingga menghasilkan keamanan yang lebih baik dibandingkan *Affine Cipher* standar.

Perkembangan teknologi kemudian mendorong munculnya kriptografi modern seperti *Advanced Encryption Standard*, *Blowfish*, serta berbagai mode operasi *block cipher* seperti *Electronic Code Book* (ECB), *Cipher Feedback* (CFB) dan yang lainnya. Kriptografi modern dinilai lebih aman dan efisien dalam proses enkripsi data digital (Stallings, 2017). Meko (2018) membandingkan algoritma *Advanced Encryption Standard*, DES, IDEA, dan *Blowfish* dalam proses enkripsi serta dekripsi data, dimana hasil penelitian menunjukkan bahwa algoritma AES memiliki kecepatan enkripsi dan dekripsi paling tinggi dibandingkan algoritma lainnya. Santoso dkk. (2024) memodifikasi metode *Electronic Code Book* (ECB) menggunakan pola *Seven Segment Display* pada pembentukan bit kunci sehingga menghasilkan *ciphertext* yang lebih acak dan dapat dikembalikan menjadi *plaintext* secara utuh. Namun, penelitian tersebut masih terbatas pada pola angka 0–9 dan belum disertai analisis keamanan yang mendalam.

Beberapa penelitian juga secara khusus meneliti mode operasi CFB seperti yang dilakukan oleh Dalimuthe (2019), dalam penelitiannya menunjukkan bahwa metode *Cipher Feedback* (CFB) dapat digunakan untuk mengamankan file dokumen teks melalui proses enkripsi dan dekripsi. Namun, penelitian ini belum dilengkapi pengujian performa dan analisis keamanan yang mendalam. Sementara

itu, Tsypyschev & Morgasov (2022)cmengkaji aspek keamanan CFB secara teoritik dengan memperkenalkan varian CFB yang dilengkapi mekanisme *external re-keying*. Pembahasan yang dilakukan masih bersifat teoritik sehingga implementasi langsung terhadap proses enkripsi data nyata belum dijelaskan secara rinci. Penelitian tersebut menegaskan bahwa CFB masih relevan untuk dikembangkan, khususnya dalam hal pengelolaan kunci dan peningkatan sifat keacakan *ciphertext*. Zhu dkk. (2025) menganalisis keamanan varian CFB dalam skema enkripsi terautentikasi dan menunjukkan bahwa mekanisme CFB tertentu masih memiliki kelemahan pada aspek autentikasi dan pengelolaan *keystream*, sehingga diperlukan pengembangan lebih lanjut untuk memperkuat keamanan mode operasi CFB dalam sistem pengkodean data atau teks.

Faktor penting dalam menentukan tingkat keamanan sistem pengkodean teks adalah pembangkitan kunci atau transformasi data. Salah satu pendekatan matematis yang dapat digunakan yaitu deret Lucas karena memiliki sifat rekursif dan pertumbuhan eksponensial yang berpotensi meningkatkan penyebaran *plaintext* serta kerandoman *ciphertext*. (Jayanti dkk., 2022). (Duman & Duman, 2021) memanfaatkan deret Lucas dalam proses enkripsi dan dekripsi dengan mengubah karakter teks menjadi nilai deret Lucas untuk membentuk *ciphertext* dan mengembalikannya kembali menjadi *plaintext* melalui proses pembalikan operasi. Namun, pola deret Lucas yang bersifat deterministik menyebabkan nilai berikutnya dapat diprediksi apabila aturan pembentukannya diketahui, sehingga diperlukan modifikasi agar *keystream* yang dihasilkan lebih bervariasi dan tidak mudah ditebak oleh pihak luar.

Berdasarkan penelitian-penelitian tersebut, algoritma *Cipher Feedback* (CFB) terbukti memiliki kemampuan yang baik dalam proses pengkodean teks, namun tingkat keacakan *ciphertext* yang dihasilkan masih dapat dikembangkan untuk meningkatkan keamanan algoritma. Selain itu, pengembangan metode pembangkitan *keystream* pada algoritma CFB masih terus diperlukan agar *ciphertext* yang dihasilkan semakin sulit diprediksi dan memiliki hubungan yang semakin rendah terhadap *plaintext*. Oleh karena itu, penelitian ini dilakukan sebagai upaya pengembangan algoritma CFB berbasis *Base64* dengan

memanfaatkan pendekatan matematis melalui modifikasi deret Lucas guna menghasilkan proses pengkodean teks yang lebih kompleks dan meningkatkan tingkat keamanan *ciphertext* yang dihasilkan.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang tersebut, maka rumusan masalah dalam penelitian ini adalah Bagaimana implementasi algoritma *Cipher Feedback* (CFB) berbasis *Base64* dengan modifikasi deret Lucas dalam proses pengkodean teks serta bagaimana tingkat keamanan pengkodean teks yang dihasilkan?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan dari penelitian ini adalah mengimplementasikan algoritma *Cipher Feedback* (CFB) berbasis *Base64* dengan modifikasi deret Lucas serta menganalisis tingkat keamanan *ciphertext* yang dihasilkan pada proses pengkodean teks.

1.4 Manfaat Penelitian

Berdasarkan tujuan penelitian, maka manfaat dari penelitian ini adalah sebagai berikut.

1. Bagi pemerhati keamanan data, penelitian ini diharapkan dapat memberikan tambahan wawasan mengenai penerapan mode operasi *Cipher Feedback* (CFB) yang dikombinasikan dengan modifikasi deret Lucas sebagai alternatif pembangkitan *keystream* dalam sistem pengkodean teks, sehingga dapat memperkuat tingkat keamanan dan keacakan hasil pengkodean.
2. Bagi mahasiswa dengan ketertarikan pada topik yang serupa atau sama, penelitian ini diharapkan dapat memberikan referensi dan menjadi landasan dalam pengembangan studi lebih lanjut di bidang kriptografi dan keamanan informasi.

BAB 2. TINJAUAN PUSTAKA

Bab 2 membahas landasan teori dan konsep-konsep fundamental yang menjadi dasar dalam pengembangan sistem pengkodean teks menggunakan algoritma *Cipher Feedback* (CFB*) Base64* berdasarkan modifikasi deret Lucas.

2.1 Kriptografi

Konsep dan definisi dari kriptografi merujuk pada Aliyah dkk. (2025). Kriptografi *(cryptography)* berasal dari bahasa Yunani, yaitu *crypto* yang berarti menyembunyikan dan *graphia* yang berarti menulis. Kriptografi merupakan ilmu sekaligus seni menjaga keamanan dengan mengubah suatu informasi menjadi bentuk yang tidak dapat dipahami oleh pihak yang tidak berwenang. Dalam kriptografi terdapat 4 istilah penting, yaitu:

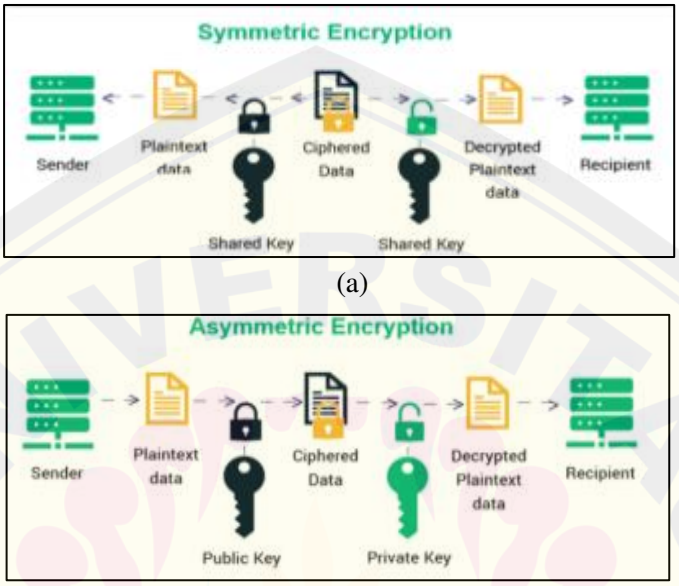
1. *Plaintext* merupakan bentuk asli dari pesan atau informasi.
2. *Ciphertext* merupakan hasil transformasi dari *plaintext* setelah melalui proses enkripsi.
3. Kunci merupakan parameter rahasia yang digunakan dalam proses enkripsi dan dekripsi.
4. Algoritma merupakan prosedur matemati yang digunakan untuk melakukan enkripsi dan dekripsi.

Proses enkripsi dan dekripsi merupakan inti dari kriptografi. Enkripsi merupakan proses mengubah *plaintext* menjadi *ciphertext* menggunakan kunci dan algooritma tertentu, sedangkan dekripsi adalah kebalikannya, yaitu mengembalikan *ciphertext* ke *plaintext* dengan kunci yang sesuai. Berdasarkan penggunaan kuncinya kriptografi terbagi menjadi dua jenis utama, yaitu kriptografi simetris dan kriptografi asimetris.

1. Kriptografi simetris, merupakan metode enkripsi yang menggunakan satu kunci rahasia yang sama untuk proses enkripsi dan dekripsi.
2. Kriptografi asimetris menggunakan dua kunci yang berbeda, yaitu kunci publik *(public key)* dan kunci privat *(private key)* dalam proses enkripsi dan dekripsi data. Kunci publik dapat dibagikan secara luas, sedangkan kunci

privat harus dijaga kerahasiaannya, sehingga pesan yang dienkripsi dengan kunci publik hanya dapat didekripsi oleh kunci privat yang sesuai.

Berikut merupakan ilustrasi dari konsep kriptografi simetris dan asimetris yang ditunjukkan pada Gambar 2.1.



(a)

(b)

(a) Kriptografi Simetris; (b) Kriptografi Asimetris

Gambar 2.1 Konsep kriptografi simetris dan asimetris (Sumber: Aliyah dkk., 2025)

2.2 Pengkodean Karakter *(Character Encoding)*

Pengkodean karakter merupakan proses mengubah bentuk informasi pesan yang berupa huruf, angka, dan karakter lainnya ke dalam bentuk kode biner. Salah satu pengkodean yang umum digunakan adalah *American Standard Code for Information Interchange (ASCII)* (Hanafi dkk., 2024).

Base64 adalah salah satu teknik pengkodean karakter suatu informasi dengan merepresentasikan bilangan biner 6 bit ke dalam format ASCII, yang didasarkan pada bilangan dasar 64. Berikut merupakan tabel *Base64* yang ditampikan pada Tabel 2.1.

Tabel 2.1 Tabel *Base64*

Des	Biner	Kar	Des	Biner	Kar	Des	Biner	Kar	Des	Biner	Kar
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	Y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	[spasi]
15	001111	P	31	011111	f	47	101111	v	63	111111	/

2.3 Algoritma *Cipher Feedback* (CFB)

Konsep, definisi, dan rumus dari CFB merujuk pada Aliyah dkk. (2025) dan Stallings (2017). Algoritma enkripsi *modern* pada dasarnya dibagi menjadi dua jenis, yaitu *block cipher* dan *stream cipher*. Pada *block cipher*, proses enkripsi dilakukan terhadap data dengan ukuran tetap, misalnya 128 bit, sehingga menghasilkan *ciphertext* dengan panjang yang sama. Apabila panjang pesan melebihi satu blok, maka pesan tersebut akan dibagi menjadi beberapa blok sebelum dilakukan proses enkripsi. Dalam proses enkripsi data berukuran besar, agar mekanisme enkripsi menjadi lebih aman dan efisien, digunakan berbagai mode operasi *(modes of operation)*.

Mode operasi *Cipher Feedback* (CFB) merupakan mode operasi yang mengubah *block cipher* menjadi *stream cipher*. Pada mode ini, proses enkripsi tidak dilakukan langsung terhadap *plaintext*, melainkan terhadap *Initialization Vector* (IV) atau *ciphertext* sebelumnya menggunakan kunci rahasia. IV adalah nilai awal yang digunakan sebelum proses enkripsi dimulai pada mode *Cipher Feedback* yang dinotasikan sebagai C_0. Hasil enkripsi tersebut menghasilkan *keystream* yang kemudian dikombinasikan dengan *plaintext* melalui operasi XOR untuk menghasilkan *ciphertext*. Selanjutnya, *ciphertext* yang dihasilkan digunakan kembali sebagai masukan (*feedback*) pada proses enkripsi berikutnya. Mekanisme

umpan balik ini berlangsung secara berantai hingga seluruh blok pesan atau teks selesai dienkripsi (Stallings, 2017).

Rumus enkripsi CFB dituliskan secara matematis pada Persamaan (2.1) sebagai berikut.

$$C_i = P_i \oplus E_K(C_{i-1}) \tag{2.1}$$

Sedangkan, rumus dekripsi dari CFB dituliskan pada Persamaan (2.2) sebagai berikut.

$$P_i = C_i \oplus E_K(C_{i-1}) \tag{2.2}$$

dengan,

P_i : *plaintext* ke – i

C_i : *ciphertext* ke - i

$E_K(C_{i-1})$: fungsi enkripsi terhadap *ciphertext* sebelumnya (C_{i-1}) menggunakan kunci K

$(i = 1, 2, 3, \dots, n)$

2.4 Deret Lucas

Deret Lucas merupakan salah satu deret penting dalam teori bilangan yang memiliki struktur rekursif mirip dengan deret Fibonacci tetapi dengan nilai awal berbeda, yaitu $L_0 = 2$ dan $L_1 = 1$. Secara umum, deret Lucas didefinisikan melalui persamaan rekursif linear orde dua pada Persamaan (2.3) sebagai berikut.

$$L_n = L_{n-1} + L_{n-2} \tag{2.3}$$

Deret Lucas memiliki bentuk representasi tertutup yang dikenal sebagai *Lucas Binet Formula*. Formula atau rumus tersebut digunakan untuk menentukan nilai suku ke n secara langsung tanpa harus menghitung suku-suku sebelumnya secara rekursif. Representasi tertutup deret Lucas ditunjukkan pada Persamaan (2.4) sebagai berikut.

$$L_n = \alpha^n + \beta^n, \quad \alpha = \frac{1+\sqrt{5}}{2}, \beta = \frac{1-\sqrt{5}}{2} \tag{2.4}$$

dengan,

L_n : suku ke-n

α, β : akar **-** akar karakteristik dari barisan Lucas

(Hoggatt, 1969).

2.5 Tingkat Perubahan Karakter (TPK)

Tingkat Perubahan Karakter (TPK) digunakan sebagai alat ukur kuantitatif yang menggambarkan tingkat perbedaan antara *plaintext* dan *ciphertext* yang dihasilkan. TPK dihitung berdasarkan perbandingan karakter antara *plaintext* dan *ciphertext* (Maulidani, 2023), dengan langkah-langkah sebagai berikut.

1. Hitung jumlah karakter *plaintext* $(P_i, i = 1, \dots, n)$ dan *ciphertext* $(C_j, j = 1, \dots, m)$
2. Memberikan bobot pada karakter *ciphertext* dengan aturan pada Persamaan (2.5) berikut.

$$B_i = \begin{cases} 1, \ P_i = C_j \text{ (untuk } i = j) \\ 2, \ P_i \neq C_j \text{ tetapi karakter } C_j \text{ terdapat dalam } plaintext \text{ (untuk } i = j) \\ 3, \ P_i \neq C_j \text{ dan karakter } C_j \text{ tidak terdapat dalam } plaintext \text{ (untuk } i = j) \end{cases} \quad (2.5)$$

3. Tingkat keamanan *ciphertext* dihitung berdasarkan TPK dengan rumus pada Persamaan (2.6) berikut.

$$TPK = \frac{nP}{\sum_{i=1}^{y} B_i + (x \times 4)} \quad (2.6)$$

dengan,

TPK : Tingkat Perubahan Karakter,

P_i : *plaintext* ke-i,

C_i : *ciphertext* ke-i,

nP : jumlah karakter pada *plaintext*,

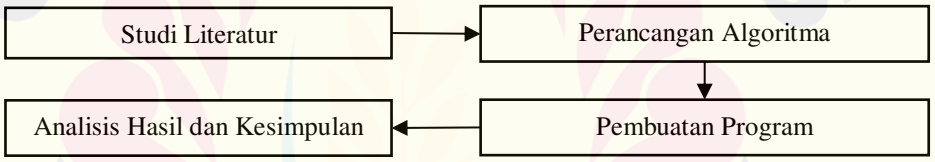
B_i : bobot *ciphertext* ke-i,

y : jumlah pasangan *plaintext* dan *ciphertext*,

x : selisih jumlah karakter *plaintext* dan *ciphertext*

BAB 3. METODOLOGI PENELITIAN

Bab 3 membahas metode penelitian yang digunakan dalam penelitian ini. Pembahasan meliputi data penelitian berupa teks, dan prosedur penelitian yang meliputi studi literatur, tahapan implementasi algoritma *Cipher Feedback* (CFB) berbasis Base64 berdasarkan modifikasi deret Lucas, perancangan pembuatan program serta analisis hasil dan kesimpulan. Data dalam penelitian ini berupa *plaintext* dan kunci sebagai data primer. Seluruh karakter pada *plaintext* dan kunci dibatasi pada 64 karakter *Base64* yang tercantum pada Tabel 2.1 agar mempermudah proses pengkodean, enkripsi, dan dekripsi. Penelitian ini dilakukan untuk mengoptimalkan keamanan pengkodean teks yaitu dengan menerapkan algoritma CFB *Base64* berdasarkan modifikasi deret Lucas. Alur proses pengkodean yang dikembangkan dijabarkan melalui prosedur yang disajikan pada Gambar 3.1.



Gambar 3.1 Prosedur penelitian

3.1 Studi Literatur

Studi literatur pada penelitian ini dilakukan dengan menelusuri dan memahami berbagai konsep yang berkaitan dengan kriptografi, pengkodeean karakter, algoritma *Cipher Feedback* (CFB), deret Lucas, dan Tingkat Perubahan Karakter (TPK). Proses penelusuran dilakukan melalui berbagai referensi yang relevan, meliputi artikel ilmiah, buku teks, dan dokumen akademik lainnya yang merujuk pada BAB 1 dan BAB 2, sehingga diperoleh landasan teori yang kuat untuk menunjang perancangan dan pengembangan algoritma pengkodean teks berbasis CFB dengan modifikasi deret Lucas.

3.2 Perancangan Algoritma

Perancangan algoritma dalam penelitian ini bertujuan untuk membangun prosedur penelitian yang sistematis pada proses enkripsi dan dekripsi menggunakan Algoritma *Cipher Feedback* (CFB) berbasis *Base64* dengan modifikasi deret Lucas. Algoritma secara umum, dirancang dalam dua bagian utama, yaitu proses enkripsi dan proses dekripsi. Kedua proses tersebut bekerja secara simetris karena operasi utama yang digunakan adalah operasi XOR yang bersifat pembalikan (*reversible*).

3.2.1 Proses Enkripsi

Proses enkripsi dilakukan melalui tahapan sebagai berikut.

1 Masukkan *plaintext* yang berisi karakter - karakter yang dimuat pada Tabel 2.1

2 Masukkan kunci

Kunci yang digunakan berbentuk modifikasi deret Lucas. Modifikasi deret Lucas dengan menggunakan *keyword*. Pada penelitian ini, *keyword* berfungsi sebagai penentu pola perubahan *keystream* melalui aturan vokal dan nonvokal (konsonan, angka dan beberapa tanda baca lain). Setiap karakter pada *keyword* memberikan pengaruh numerik terhadap elemen *keystream* yang bersesuaian. Deret Lucas termodifikasi dapat didefinisikan sebagai Persamaan (3.1), (3.2) dan (3.3) berikut.

$$L'_0 = L_0 + \delta(k_1) \tag{3.1}$$

$$L'_1 = L_1 + \delta(k_2) \tag{3.2}$$

$$L'_n = (L'_{n-1} + L'_{n-2}) + \delta(k_{n+1}), n \geq 2 \tag{3.3}$$

dengan,

L'_n = nilai deret Lucas termodifikasi pada suku ke-n

k_i = karakter ke i dari *keyword* (periodik) ($i = 1, 2, 3, \dots, n$)

$\delta(k_i)$ = fungsi penyesuaian karakter *keyword* vokal – nonvokal pada indeks ke i ($i = 1, 2, 3, \dots, n$)

$$\delta(k_i) = \begin{cases} 1, & x \in \{\text{A, I, U, E, O}\} \text{ (vokal)} \\ 0, & \text{lainnya} \end{cases}$$

$L_0 \quad = 2$

$L_1 \quad = 1$

Sehingga, langkah – langkah pembentukan kunci modifikasi deret Lucas dapat dituliskan sebagai berikut.

a Masukkan *keyword* dalam bentuk kata.

b Panjangkan *keyword* hingga panjangnya sama dengan panjang *plaintext,* Sementara itu, jika panjang kunci melebihi panjang *plaintext*, maka kunci dipangkas hingga panjangnya sesuai dengan *plaintext*.

c Tentukan $\delta(k_i)$ dengan menentukan $\delta(x)$ setiap karakter apakah termasuk vokal atau nonvokal (konsonan, angka , dan beberapa tanda baca lain) sebagaimana definisi fungsi indikator karakter di atas.

d Bentuklah kunci modifikasi deret Lucas dengan mensubtitusi $\delta(k_i)$ yang telah ditentukan sebelumnya ke dalam persamaan deret Lucas termodifikasi

e Terapkan operasi modulo 64 agar seluruh elemen *keystream* sesuai dengan rentang karakter *Base64*.

f Hasil dari modifikasi tersebut merupakan *keystream* berupa deret baru yang kemudian dikonversi menjadi nilai desimal berdasarkan Tabel 2.1.

3 Tentukan nilai *Initialization Vector* (IV). Pada tahap penentuan *Initialization Vector* (IV), digunakan sebuah *keyword* sebelumnya, yang terlebih dahulu dikonversi ke dalam nilai desimal berdasarkan tabel *Base64* untuk setiap karakter penyusunnya. Selanjutnya, nilai desimal tersebut direpresentasikan dalam bentuk biner 6 bit, kemudian seluruh biner hasil konversi dioperasikan menggunakan XOR secara berurutan untuk menghasilkan satu nilai biner akhr. Nilai biner hasil operasi XOR tersebut ditetapkan sebagai *Initialization Vector* (IV) yang digunakan sebagai masukan awal pada proses enkripsi.

4 Nilai *Initialization Vector* (IV) tersebut dienkripsi menggunakan fungsi enkripsi dasar dimana nilai IV disubtitusi sebagai C_0 lalu di XOR dengan *keystream* pertama. *Keystream* yang dimaksud berupa karakter pertama pada deret Lucas yang telah dimodifikasi. Hasil enkripsi IV tersebut dioperasikan

menggunakan operasi XOR dengan nilai *plaintext* pada karakter pertama yang bersesuaian. Proses ini merujuk pada Persamaan (2.1).

5 Hasil dari operasi XOR tersebut menghasilkan *ciphertext* pada karakter pertama yang selanjutnya dikonversi kembali ke dalam bentuk karakter *Base64* sesuai dengan Tabel 2.1.

6 Untuk karakter selanjutnya, Hasil *ciphertext* sebelumnya digunakan sebagai masukan umpan balik (*feedback*) menggantikan IV yaitu C_1 ,lalu di enkripsi kembali atau di XOR dengan *keystream* selanjutnya. Hasil enkripsi C_1 tersebut dioperasikan kembali menggunakan operasi XOR dengan nilai *plaintext* pada karakter kedua yang bersesuaian. Hasil XOR kembali dikonversi menjadi karakter *Base64* berdasarkan Tabel 2.1.

7 Langkah 6 terus diulang sebanyak panjang karakter *plaintext* yang akan dienkripsi

8 Hasil akhir proses enkripsi secara keseluruhan berupa *ciphertext.*

3.2.2 Proses dekripsi

Proses dekripsi dilakukan melalui tahapan sebagai berikut.

1 Masukkan *ciphertext* yang berisi karakter - karakter yang dimuat pada Tabel 2.1

2 Masukkan kunci

Tahapan pembentukan kunci pada proses dekripsi sama dengan proses enkripsi, yaitu melakukan kembali modifikasi terhadap deret Lucas menggunakan *keyword*. Karena algoritma bersifat simetris, pola modifikasi pada dekripsi harus mengikuti aturan yang sama seperti proses enkripsi agar *keystream* yang terbentuk identik. *Keyword* tetap digunakan sebagai penentu perubahan nilai pada setiap elemen *keystream* melalui aturan karakter vokal dan nonvokal (konsonan, angka, dan beberapa tanda baca lain).

3 Tentukan nilai *Initialization Vector* (IV). Pada tahap penentuan *Initialization Vector* (IV), digunakan sebuah *keyword* sebelumnya, yang terlebih dahulu dikonversi ke dalam nilai desimal berdasarkan tabel *Base64* untuk setiap karakter penyusunnya. Selanjutnya, nilai desimal tersebut direpresentasikan

dalam bentuk biner 6 bit, kemudian seluruh biner hasil konversi dioperasikan menggunakan XOR secara berurutan untuk menghasilkan satu nilai biner akhr. Nilai biner hasil operasi XOR tersebut ditetapkan sebagai *Initialization Vector* (IV) yang digunakan sebagai masukan awal pada proses enkripsi.

4 Nilai *Initialization Vector* (IV) tersbut didekripsi menggunakan fungsi dekripsi dasar dimana nilai IV disubtitusi sebagai C_0 lalu di XOR dengan *keystream* pertama. *Keystream* atau kunci yang dimaksud berupa karakter pertama pada deret Lucas yang telah dimodifikasi. Hasil enkripsi IV tersebut dioperasikan menggunakan operasi XOR lagi dengan nilai *ciphertext* pada karakter pertama yang bersesuaian. Proses ini merujuk Persamaan (2.2).

5 Hasil dari operasi XOR tersebut menghasilkan *plaintext* pada karakter pertama yang selanjutnya dikonversi kembali ke dalam bentuk karakter *Base64* sesuai dengan Tabel 2.1.

6 Karakter *ciphertext* pertama (C_1) menjadi masukan umpan balik (*feedback*), lalu di enkripsi kembali atau di XOR dengan *keystream* atau kunci selanjutnya. Hasil enkripsi C_1 tersebut dioperasikan kembali menggunakan operasi XOR dengan nilai *ciphertext* pada karakter kedua yang bersesuaian. Hasil XOR kembali dikonversi menjadi karakter *Base64* berdasarkan Tabel 2.1.

7 Langkah 6 terus diulang sebanyak panjang karakter *ciphertext* yang akan didekripsi

8 Hasil akhir proses dekripsi secara keseluruhan berupa *plaintext.*

3.3 Pembuatan Program

Program ini dirancang dan dikembangkan menggunakan MATLAB versi 2013a karena pada versi tersebut tersedia fitur *Graphical User Interface* (GUI) yang mendukung pembuatan antarmuka pengguna secara interaktif. Antarmuka yang dibangun terdiri atas beberapa komponen utama, yaitu panel input, tombol enkripsi, tombol dekripsi, serta area hasil, yang disusun secara sistematis agar pengguna dapat menjalankan proses tanpa perlu menuliskan perintah secara manual. Meskipun MATLAB 2013a memiliki keterbatasan dalam hal kecepatan

eksekusi dibandingkan versi yang lebih baru, program tetap dapat berfungsi sesuai dengan alur proses enkripsi dan dekripsi yang telah dirancang dalam penelitian ini.

3.4 Analisis Hasil dan Kesimpulan

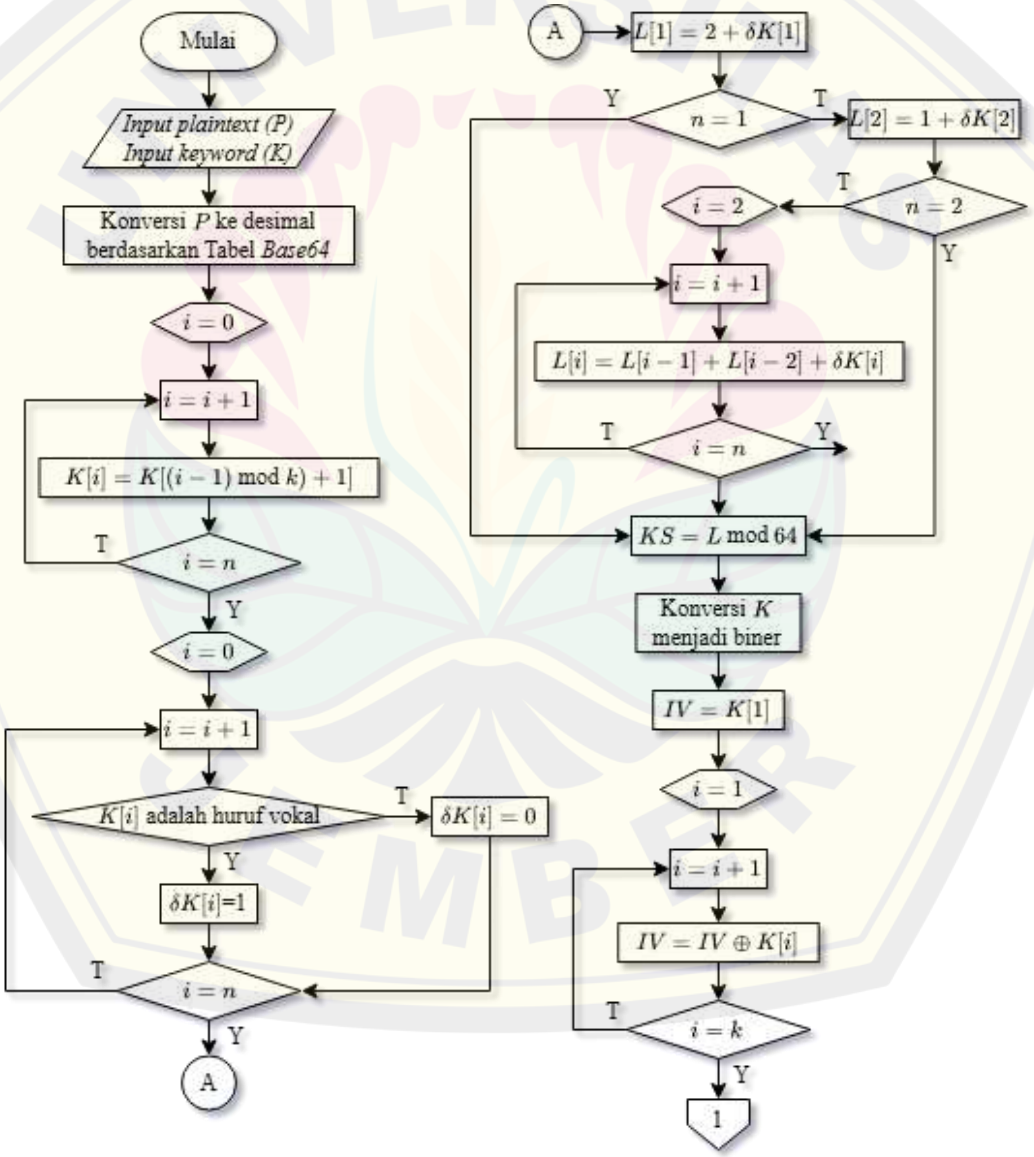
Analisis hasil penelitian dilakukan untuk menilai keselarasan antara output yang dihasilkan dengan konsep serta rancangan algoritma yang telah dirumuskan sebelumnya. Proses evaluasi difokuskan pada ketepatan mekanisme enkripsi dan dekripsi, serta pada aspek keamanan algoritma yang dikembangkan. Pengukuran tingkat keamanan dilakukan dengan cara membandingkan tingkat perubahan karakter (TPK) pada *ciphertext* yang dihasilkan oleh algoritma *Cipher Feedback* (CFB) dengan pembangkit kunci modifikasi deret Lucas dengan algoritma *Cipher Feedback* (CFB) standar tanpa modifikasi, sebagaimana dirumuskan pada Persamaan (2.6). Berdasarkan hasil analisis tersebut, selanjutnya ditarik kesimpulan untuk menentukan apakah algoritma yang dikembangkan telah mampu mencapai tujuan perancangan, baik ditinjau dari sisi fungsionalitas maupun tingkat keamanannya.

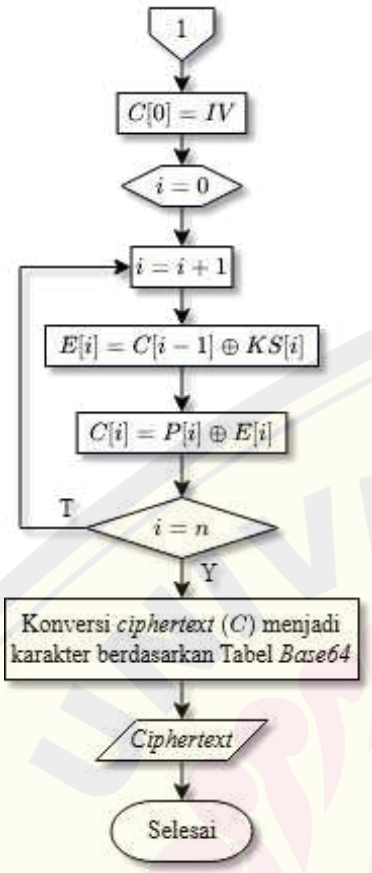
BAB 4. HASIL DAN PEMBAHASAN

Bab 4 memaparkan mengenai hasil penelitian berupa penerapan modifikasi algoritma CFB berdasarkan deret Lucas dan hasil TPK sebagai penguji tingkat keamanan dari metode yang diterapkan. Modifikasi algoritma diterapkan untuk pengkodean teks dengan memanfaatkan *sofware* MATLAB 2013a.

4.1 Proses Enkripsi

Program untuk proses enkripsi dirancang berdasarkan flowchart pada Gambar 4.1 berikut.





Keterangan:

P	: *Plaintext*
K	: *Keyword*
n	: Panjang *Plaintext*
k	: Panjang *Keyword*
KS	: *Keystream*
IV	: *Initialization Vector*
C	: *Ciphertext*
L	: Deret lucas termodifikasi (barisan *keystream* yang terbentuk)
i	: Indeks atau iterasi karakter
$K[i]$	: *Keyword* ke -i
$KS[i]$	: *Keystream* ke -i
$E[i]$	: Hasil Enkripsi *Feedback* pada Iterasi ke -i
$P[i]$	: *Plaintext* ke -i
$C[i]$	: *Ciphertext* ke -i
$L[i]$	: Suku ke -i deret lucas termodifikasi
$\delta K[i]$	: Fungsi indikator *keyword* ke -i
$L[1]$	: Suku pertama deret lucas termodifikasi
$L[2]$	: Suku kedua deret lucas termodifikasi

Gambar 4.1 *Flowchart* proses enkripsi

Sebagai simulasi untuk tahap enkripsi dilakukan dengan *plaintext* (P) misalnya "AKU" dan kunci enkripsi atau *keyword* (K) "OK" adalah sebagai berikut.

1. Setiap karakter *plaintext* dan *keyword* dikonversi terlebih dahulu ke dalam bentuk desimal berdasarkan tabel *Base64*, sehingga

$$P = [0, 10, 20]$$

$$K = [14, 10]$$

2. Panjang *plaintext* adalah 3 dan panjang *keyword* adalah 2. Panjang *keyword* lebih pendek dari *plaintext*, maka *keyword* diulang hingga panjangnya sama dengan *plaintext*. sehingga menjadi

$$K = [14, 10, 14]$$

3. Nilai fungsi indikator *keyword* ditentukan berdasarkan karakter vokal dan nonvokal (konsonan, angka dan beberapa tanda baca lain). Aturan nilai fungsi indikator adalah sebagai berikut.

   $\delta(k_i) = 1 \rightarrow$ Vokal

   $\delta(k_i) = 0 \rightarrow$ Lainnya

   Sehingga diperoleh :

   $$\delta(k) = [1, 0, 1]$$

4. *Keystream* dibentuk menggunakan modifikasi deret lucas berdasarkan Persamaan (3.1) hingga (3.3).

   $L'_0 = L_0 + \delta(k_1)$

   $L'_1 = L_1 + \delta(k_2)$

   $L'_n = (L'_{n-1} + L'_{n-2}) + \delta(k_{n+1}), n \geq 2$

   dengan, $L_0 = 2$, $L_1 = 1$

   Maka,

   $L'_0 = 2 + 1 = 3$

   $L'_1 = 1 + 0 = 1$

   $L'_2 = (1 + 3) + 1 = 5$

   Sehingga diperoleh *keystream* yaitu 3, 1, 5

   $$KS = [3, 1, 5]$$

   Seluruh elemen masih berada pada rentang *Base64*, maka *keystream* tetap

   $$KS \; mod \; 64 = [3, 1, 5].$$

5. *Initialization Vector* (IV) ditentukan dari hasil XOR berurutan *keyword*. *keyword* dikonversi ke biner sebagaimana yang tertulis pada Tabel 4.1 berikut ini.

   Tabel 4.1 Proses konversi *keyword*

Keyword	Desimal	Biner
O	14	001110
K	10	001010
O	14	001110

   Maka,

   $IV = 001110 \oplus 001010 \oplus 001110$

   $= 001010$

6. *Ciphertext* (*C*) dihitung berdasarkan Persamaan (2.1)

$C_i = P_i \oplus E_K(C_{i-1})$

$C_i = P_i \oplus (KS_i \oplus (C_{i-1}))$

$C_0 = IV$

Maka,

a. Iterasi 1

$C_1 = P_1 \oplus (KS_1 \oplus (C_{1-1}))$

$= P_1 \oplus (KS_1 \oplus C_0)$

$= 0 \oplus (3 \oplus 10)$

$= 000000 \oplus (000011 \oplus 001010)$

$= 000000 \oplus 001001$

$= 001001$

$C_1 = 9$

b. Iterasi 2

$C_2 = P_2 \oplus (KS_2 \oplus (C_{2-1}))$

$= P_2 \oplus (KS_2 \oplus C_1)$

$= 10 \oplus (1 \oplus 9)$

$= 001010 \oplus (000001 \oplus 001001)$

$= 001010 \oplus 001000$

$= 000010$

$C_2 = 2$

c. Iterasi 3

$C_3 = P_3 \oplus (KS_3 \oplus (C_{3-1}))$

$= P_3 \oplus (KS_3 \oplus C_2)$

$= 20 \oplus (5 \oplus 2)$

$= 010100 \oplus (000101 \oplus 000010)$

$= 010100 \oplus 000111$

$= 010011$

$C_3 = 19$

Diperoleh *ciphertext* akhir adalah sebagai berikut.

$$C = [9, 2, 19]$$

7. *Ciphertext* akhir dikonversi kembali ke dalam karakter *Base64*.

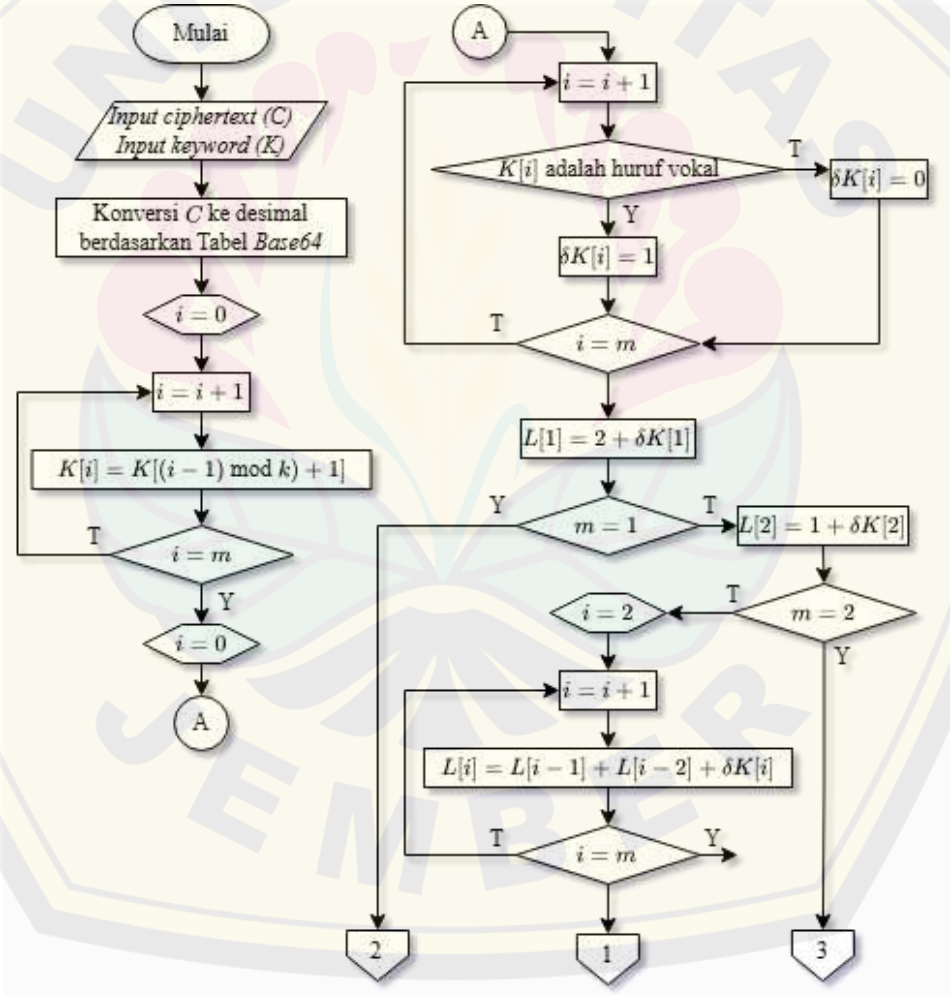
   $C = [9, 2, 19]$

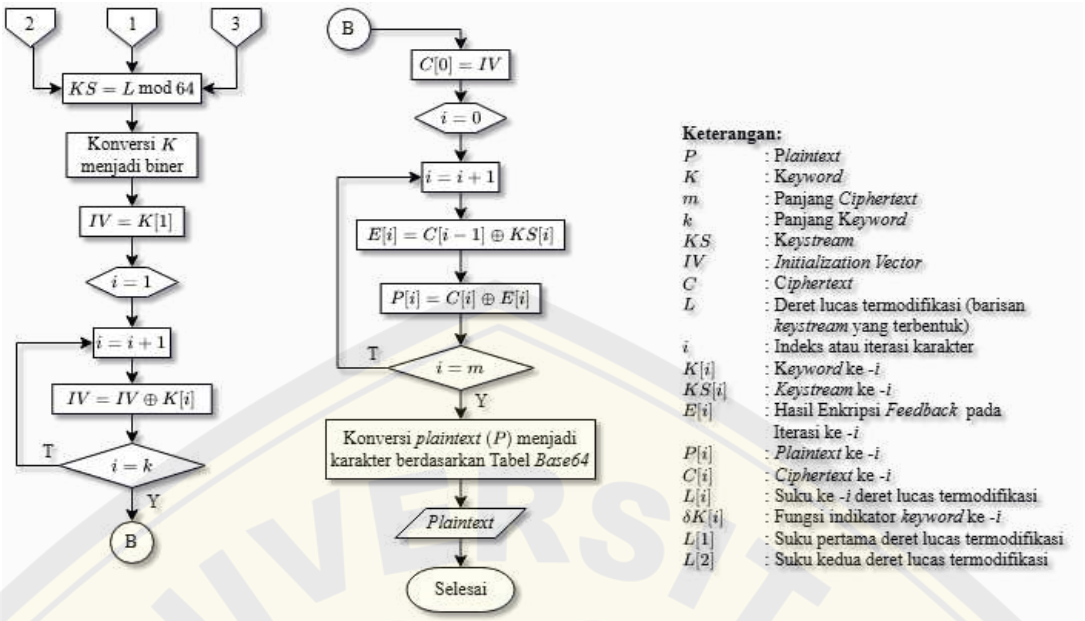
   $C = [J, C, T]$

   Sehingga *ciphertext* akhir adalah " JCT "

4.2 Proses dekripsi

Program untuk proses dekripsi dirancang berdasarkan *flowchart* pada Gambar 4.2 berikut.





Gambar 4.2 *Flowchart* proses dekripsi

Sebagai simulasi untuk tahap dekripsi dilakukan dengan *ciphertext* (*C*) misalnya "JCT" dan kunci enkripsi atau *keyword* (*K*) "OK" adalah sebagai berikut.

1. Setiap karakter *ciphertext* dan *keyword* dikonversi terlebih dahulu ke dalam bentuk desimal berdasarkan tabel *Base64*, sehingga

$$C = [9, 10, 19]$$

$$K = [14, 10]$$

2. Panjang *plaintext* adalah 3 dan panjang *keyword* adalah 2. Panjang *keyword* lebih pendek dari *plaintext*, maka *keyword* diulang hingga panjangnya sama dengan *plaintext*. sehingga menjadi

$$K = [14, 10, 14]$$

3. Nilai fungsi indikator *keyword* ditentukan berdasarkan karakter vokal dan nonvokal (konsonan, angka, dan beberapa tanda baca lain). Aturan nilai fungsi indikator adalah sebagai berikut.

   $\delta(k_i) = 1 \rightarrow$ Vokal

   $\delta(k_i) = 0 \rightarrow$ Lainnya

   Sehingga diperoleh :

$$\delta(k) = [1, 0, 1]$$

4. *Keystream* dibentuk menggunakan modifikasi deret lucas berdasarkan Persamaan (3.1) hingga (3.3).

   $L'_0 = L_0 + \delta(k_1)$

   $L'_1 = L_1 + \delta(k_2)$

   $L'_n = (L'_{n-1} + L'_{n-2}) + \delta(k_{n+1}), n \geq 2$

   dengan, $L_0 = 2$, $L_1 = 1$

   Maka,

   $L'_0 = 2 + 1 = 3$

   $L'_1 = 1 + 0 = 1$

   $L'_2 = (1 + 3) + 1 = 5$

   Sehingga diperoleh *keystream* yaitu 3, 1, 5

   $$KS = [3, 1, 5]$$

   Seluruh elemen masih berada pada rentang *Base64*, maka *keystream* tetap

   $$KS \; mod \; 64 = [3, 1, 5].$$

5. Initialization Vector (IV) ditentukan dari hasil XOR berurutan *keyword*. *keyword* dikonversi ke biner berdasarkan Tabel 4.1.

   Maka,

   $IV = 001110 \oplus 001010 \oplus 001110$

   $= 001010$

6. *Plaintext* (*P*) dihitung berdasarkan Persamaan (2.2)

   $P_i = C_i \oplus E_K(C_{i-1})$

   $P_i = C_i \oplus (KS_i \oplus (C_{i-1}))$

   $C_0 = IV$

   Maka,

   a. Iterasi 1

      $P_1 = C_1 \oplus (KS_1 \oplus (C_{1-1}))$

      $= C_1 \oplus (KS_1 \oplus C_0)$

      $= 9 \oplus (3 \oplus 10)$

      $= 001001 \oplus (000011 \oplus 001010)$

      $= 000011 \oplus 001001$

      $= 000000$

$P_1 = 0$

b. Iterasi 2

$P_2 = C_2 \oplus (KS_2 \oplus (C_{2-1}))$

$= C_2 \oplus (KS_2 \oplus C_1)$

$= 2 \oplus (1 \oplus 9)$

$= 000010 \oplus (000001 \oplus 001001)$

$= 000010 \oplus 001000$

$= 001010$

$P_2 = 10$

c. Iterasi 3

$P_3 = C_3 \oplus (KS_3 \oplus (C_{3-1}))$

$= C_3 \oplus (KS_3 \oplus C_2)$

$= 19 \oplus (5 \oplus 2)$

$= 011001 \oplus (000101 \oplus 000010)$

$= 010011 \oplus 000111$

$= 010100$

$P_3 = 20$

Diperoleh *ciphertext* akhir adalah sebagai berikut.

$$C = [0, 10, 20]$$

7. *Plaintext* akhir dikonversi kembali ke dalam karakter *Base64*.

$P = [0, 10, 20]$

$P = [A, K, U]$

Sehingga *plaintext* akhir adalah " AKU ".

4.3 Evaluasi Keamanan

Tingkat keamanan algoritma dianalisis menggunakan TPK berdasarkan Persamaan (2.4) dengan melakukan perbandingan nilai TPK antara algoritma *Cipher Feedback* (CFB) standar atau sebelum modifikasi dan algoritma *Cipher Feedback* (CFB) berdasarkan Modifikasi Deret Lucas. Berikut contoh perhitungan TPK dari *plaintext* "AKU" dan kunci (*keyword*) "OK"

a. *Ciphertext* dari algoritma *Cipher Feedback* (CFB) Standar dan Setelah Modifikasi

Ciphertext CFB standar dan CFB setelah modifikasi diuraikan pada Tabel 4.2 berikut.

Tabel 4.2 Perhitungan *Ciphertext* CFB standar dan CFB modifikasi

CFB Standar	CFB Modifikasi
$P = [0, 10, 20]$	$P = [0, 10, 20]$
$K = [14, 10]$	$K = [14, 10]$
maka,	maka,
$KS = [14, 10, 14]$	$K = [14,10,14]$
$IV = 0$	$\delta(k) = [1, 0, 1]$
$C_0 = IV$	$KS = [3, 1, 5]$
sehingga,	$IV = 001110 \oplus 001010 \oplus 001110$
$C_1 = 0 \oplus (14 \oplus 0) = 14$	$= 10$
$C_2 = 10 \oplus (10 \oplus 14) = 14$	$C_0 = IV$
$C_3 = 20 \oplus (14 \oplus 14) = 20$	sehingga,
$C = [14, 14, 20]$	$C_1 = 0 \oplus (3 \oplus 10) = 9$
$C = [O, O, U]$	$C_2 = 10 \oplus (1 \oplus 9) = 2$
Ciphertext akhir adalah "OOU"	$C_3 = 20 \oplus (5 \oplus 2) = 19$
	$C = [9, 2, 19]$
	$C = [J, C, T]$
	Ciphertext akhir adalah "JCT"

Sehingga *ciphertext* yang dihasilkan dari algoritma *Cipher Feedback* (CFB) standar atau sebelum modifikasi adalah "OOU' dan *ciphertext* yang dihasilkan algoritma *Cipher Feedback* (CFB) berdasarkan Modifikasi Deret Lucas adalah "JCT".

b. Nilai TPK dari algoritma *Cipher Feedback* (CFB) Standar dan Setelah Modifikasi.

Perhitungan Nilai TPK dari CFB standar dan CFB setelah modifikasi dapat diuraikan pada Tabel 4.3 berikut.

Tabel 4.3 Perhitungan nilai TPK *Ciphertext* CFB standar dan CFB modifikasi

CFB Standar	CFB Modifikasi
$P = [A,\ K,\ U]$ $C\ =\ [O, O, U]$ $C_1 = O, C_2 = O, C_3 = U$ maka, $B_1 = 3$ $B_2 = 3$ $B_3 = 1$ Sehingga nilai TPK nya, $TPK = \frac{3}{7+(0\times4)} = \frac{3}{7} = 0{,}4286$	$P = [A,\ K,\ U]$ $C\ =\ [J, C, T]$ $C_1 = \mathrm{J},\ C_2 = \mathrm{C},\ C_3 = \mathrm{T}$ maka, $B_1 = 3$ $B_2 = 3$ $B_3 = 3$ Sehingga nilai TPK nya, $TPK = \frac{3}{9+(0\times4)} = \frac{3}{9} = 0{,}3333$

Pengujian tingkat keamanan juga dilakukan terhadap beberapa *plaintext* dan *keyword* yang berbeda – beda. Hasil pengujian tingkat keamanan diuraikan pada Lampiran dan Tabel 4.4 berikut ini

Tabel 4.3 Pengujian tingkat keamanan *ciphertext*

Plaintext	*Keyword*	*Ciphertext*		Nilai TPK	
		CFB Standar	CFB Modifikasi	CFB Standar	CFB Modifikasi
DaTa12	kEy	n5YmXT	BZOSsL	0.3529	0.3333
KRipto	LUCAS	BEkNyR	Mf5XxL	0.3529	0.3333
Base64	CiPHER	D7YB/W	+mOWnO	0.3523	0.3222
MATEMATIKA	UNEJ	YVCPXaNMSf	WXBDDRdk/9	0.3448	0.3233
Keamanan1	Xor	drarZVYXJ	cAe+vZfWp	0.3600	0.3462
ALGORITMA	CFB	CMLHTaLCD	EOLBXUVEr	0.3913	0.3600
informasi	Akurat	ihQTipz73	9bBvI85kX	0.3750	0.3333
eNkriPSI	FEedBack	bSoe9omK	9yT/RK4E	0.3636	0.3121
UniversitasJemberJaya	Kampusmerdeka	kzdhL2Rtl7FW5/kGE	ejnhRWcgmhT+6WXvtK8os	0.3684	0.3387

Berdasarkan hasil perhitungan, nilai TPK algoritma CFB berbasis modifikasi deret Lucas lebih kecil dibandingkan algoritma CFB standar. Sesuai dengan Maulidani (2023), nilai TPK yang semakin kecil menunjukkan bahwa hubungan antara *plaintext* dan *ciphertext* menjadi semakin kecil juga. Kondisi tersebut mengindikasikan bahwa *ciphertext* yang dihasilkan memiliki tingkat kerandoman yang lebih tinggi dan lebih sulit diprediksi oleh pihak yang tidak memiliki kunci

4.4 Implementasi Algoritma *Cipher Feedback* (CFB) pada Matlab

Tahap pertama pada implementasi algoritma adalah melakukan validasi terhadap *plaintext*, *ciphertext* dan *keyword* menggunakan fungsi `validasi_enkripsi` dan `validasi_dekripsi`. Tujuan validasi adalah memastikan data yang dimasukkan pengguna telah memenuhi syarat sebelum proses enkripsi dan dekripsi dijalankan.

```
if isempty(plaintext) && isempty(keyword)
    isValid = false;
    pesan = 'Plaintext dan Keyword tidak boleh kosong!';
    return;
if isempty(ciphertext) && isempty(keyword)
    isValid = false;
    pesan = 'Ciphertext dan Keyword tidak boleh kosong!';
    return;
```

Sintaks `isempty()` digunakan untuk memeriksa apakah textbox *plaintext* atau *ciphertext* maupun *keyword* masih kosong. Jika salah satu atau keduanya kosong, program akan menampilkan pesan kesalahan dan menghentikan proses menggunakan perintah `return`. Dengan demikian, proses enkripsi dan dekripsi tidak akan dijalankan apabila data masukan belum lengkap. Selanjutnya dilakukan pemeriksaan karakter Base64 menggunakan sintaks berikut.

```
invalid_plain = plaintext(~ismember(plaintext, base64));
invalid_cipher = ciphertext(~ismember(ciphertext, base64));
invalid_key   = keyword(~ismember(keyword, base64));
```

Fungsi `ismember()` digunakan untuk memeriksa apakah setiap karakter *plaintext, ciphertext* dan *keyword* terdapat pada himpunan karakter Base64 yang telah ditentukan. Jika ditemukan karakter di luar himpunan tersebut, program akan

menampilkan pesan bahwa *plaintext* mengandung karakter yang tidak valid sehingga proses enkripsi dihentikan.

Tahap kedua adalah penentuan nilai δ dengan melakukan pemeriksaan setiap karakter *keyword* untuk menentukan apakah karakter tersebut merupakan karakter vokal atau nonvokal (konsonan, angka, dan beberapa tanda baca lain).

```
delta = zeros(1,n);

for i = 1:n
    huruf = key_full(i);

    if any(huruf == 'AEIOUaeiou')
        delta(i) = 1;
    else
        delta(i) = 0;
    end
end
```

Pembentukan nilai $\delta(k_i)$ sebagai fungsi indikator pada proses modifikasi deret Lucas diawali dengan menginisialisasi vektor delta menggunakan fungsi `zeros` untuk menyediakan tempat penyimpanan nilai fungsi indikator sebanyak panjang *plaintext*. Selanjutnya, perulangan `for` digunakan untuk memproses setiap karakter pada *keyword* yang telah disesuaikan panjangnya dengan *plaintext*. Pada setiap iterasi, karakter *keyword* disimpan ke dalam variabel huruf, kemudian dilakukan proses pemeriksaan menggunakan struktur percabangan `if` yang dipadukan dengan fungsi `any` untuk menentukan apakah karakter tersebut termasuk karakter vokal, yaitu A, E, I, O, U baik kapital maupun tidak. Apabila kondisi tersebut terpenuhi, maka elemen vektor `delta` (δ) pada indeks yang bersesuaian diberikan nilai 1, sedangkan apabila karakter yang diperiksa merupakan karakter nonvokal (konsonan, angka, dan beberapa tanda baca lain maka diberikan nilai 0. Proses ini berlangsung hingga seluruh karakter *keyword* selesai diperiksa sehingga diperoleh vektor `delta` (δ) yang berisi nilai fungsi indikator untuk setiap karakter *keyword*. Seluruh nilai δ diperoleh, selanjutnya program membentuk deret Lucas termodifikasi menggunakan sintaks berikut.

```
L(1) = 2 + delta(1);
L(2) = 1 + delta(2);
```

Sintaks tersebut digunakan untuk membentuk dua suku awal deret Lucas termodifikasi berdasarkan nilai awal deret Lucas dan fungsi indikator karakter *keyword*. Selanjutnya pembentukan suku berikutnya dilakukan menggunakan perulangan.

```
for i = 3:n
    % L_n' = L_(n-1)' + L_(n-2)' + delta(k_(n+1))
    L(i) = L(i-1) + L(i-2) + delta(i);
end
```

Perulangan `for` digunakan untuk menghitung setiap suku deret Lucas mulai dari suku ketiga hingga jumlah karakter *plaintext*. Setiap suku diperoleh dari penjumlahan dua suku sebelumnya yang ditambahkan dengan nilai fungsi indikator pada karakter *keyword* ke-*i*. Kemudian setelah seluruh deret Lucas diperoleh, dilakukan operasi modulo 64 sehingga terbentuklah *keystream*.

```
ks = mod(L, 64);
```

Fungsi `mod()` digunakan untuk memastikan seluruh nilai keystream berada pada rentang 0–63 sehingga sesuai dengan jumlah karakter pada tabel Base64. Nilai inilah yang digunakan sebagai keystream selama proses enkripsi.

Tahap selanjutnya adalah pembentukan IV menggunakan operasi XOR terhadap seluruh karakter *keyword*.

```
iv_bin = bin(1,:) - '0';

for i=2:size(bin,1)
    iv_bin = xor(iv_bin, bin(i,:) - '0');
end
```

Variabel `IV` diinisialisasi menggunakan representasi biner karakter pertama *keyword*. Selanjutnya fungsi `xor()` digunakan untuk melakukan operasi XOR secara berurutan terhadap seluruh karakter *keyword* hingga diperoleh satu nilai biner yang digunakan sebagai *Initialization Vector*.

Tahap utama implementasi algoritma pada program adalah proses enkripsi dan dekripsi menggunakan algoritma *Cipher Feedback* (CFB).

```
enc  = bitxor(prev, ks(i));
c(i) = bitxor(p(i), enc);
p(i) = bitxor(c(i), enc);
```

Fungsi `bitxor()` digunakan untuk melakukan operasi XOR antar data bertipe bilangan bulat. Pada sintaks pertama dilakukan proses pembentukan nilai *feedback* dengan melakukan operasi XOR pada *ciphertext* sebelumnya (`prev`)

terhadap keystream (`ks(i)`). Hasilnya disimpan pada variabel `enc`. Selanjutnya nilai `enc` di-XOR dengan *plaintext* sehingga menghasilkan *ciphertext* pada iterasi ke-*i* atau di - XOR dengan *ciphertext* sehingga menghasilkan *plaintext* pada iterasi ke -*i*. *Ciphertext* yang dihasilkan kemudian digunakan kembali sebagai *feedback* pada iterasi berikutnya sesuai mekanisme algoritma *Cipher Feedback* (CFB).

Tahap terakhir yaitu menghitung nilai Tingkat Perubahan Karakter (TPK) dengan membandingkan *plaintext* dan *ciphertext*.

```
ciphertext = regexprep(ciphertext, '[^A-Za-z0-9+/]', '');
```

Proses diawali dengan menghilangkan karakter yang bukan termasuk himpunan Base64 menggunakan fungsi `regexprep`, sehingga hanya karakter valid yang digunakan pada proses perhitungan.

```
n = length(plaintext);
m = length(ciphertext);
```

Variabel `n` dan `m` digunakan untuk mengetahui jumlah karakter *plaintext* dan *ciphertext* yang akan dibandingkan pada proses selanjutnya. Kemudian dibuat vektor `B` sebagai tempat penyimpanan bobot hasil perbandingan setiap karakter.

```
B = zeros(1,nC);
```

Vektor `B` diinisialisasi dengan nilai nol sehingga setiap elemen dapat diisi dengan bobot sesuai hasil perbandingan karakter *plaintext* dan *ciphertext*. Proses perbandingan karakter dilakukan menggunakan perulangan `for` dan struktur percabangan `if`.

```
for i = 1:nC
    if i <= nP
        if plaintext(i) == ciphertext(i)
            B(i) = 1;
        elseif ~isempty(strfind(plaintext, ciphertext(i))) % ganti
contains
            B(i) = 2;
        else
            B(i) = 3;
        end
    else
        B(i) = 4;
    end
end
```

Pada setiap iterasi, karakter *plaintext* dan *ciphertext* dibandingkan untuk menentukan bobot perubahan karakter. Apabila kedua karakter sama maka diberikan bobot 1, apabila karakter *ciphertext* masih terdapat pada *plaintext* tetapi

berada pada posisi yang berbeda diberikan bobot 2, sedangkan apabila karakter *ciphertext* tidak terdapat pada *plaintext* diberikan bobot 3. Jika panjang *ciphertext* melebihi panjang *plaintext*, maka diberikan bobot 4. Terakhir, nilai TPK dihitung menggunakan persamaan berikut.

```
TPK = nP / (sum(B) + (x * 4));
```

Sintaks tersebut menghitung nilai TPK berdasarkan perbandingan antara panjang *plaintext* dengan jumlah seluruh bobot perubahan karakter serta selisih panjang *plaintext* dan *ciphertext*. Nilai TPK yang lebih kecil menunjukkan bahwa *ciphertext* memiliki tingkat perubahan karakter yang lebih tinggi terhadap *plaintext*, sehingga hubungan antara *plaintext* dan *ciphertext* menjadi semakin kecil atau tidak signifikan.

4.5 Pembuatan Program menggunakan *Grapichal User Interface* (GUI)

Program dirancang menggunakan *Graphical User Interface* (GUI) pada MATLAB 2013a sebagaimana ditunjukkan pada Gambar 4.3 berikut. Program tersebut terdiri atas tiga bagian utama, yaitu bagian enkripsi, dekripsi, dan perhitungan tingkat perubahan karakter (TPK).

Gambar 4.3 Tampilan program algoritma CFB

Simulasi program dimulai dari proses enkripsi dengan memasukkan *plaintext* berupa karakter *Base64* ke dalam edit text pada bagian “Masukkan

Plaintext (*Base64*)". Selanjutnya, *keyword* berupa karakter *Base64* dimasukkan ke dalam edit text pada bagian "Masukkan *Keyword* (*Base64*)". Proses enkripsi dijalankan dengan menekan *push button* "Enkripsi". Setelah tombol ditekan, program akan membangkitkan *keystream* menggunakan modifikasi deret Lucas berdasarkan *keyword*, kemudian menentukan nilai Initialization Vector (IV) menggunakan operasi XOR berurutan dari *keyword*, dan selanjutnya melakukan proses enkripsi menggunakan algoritma *Cipher Feedback* (CFB). Hasil proses enkripsi berupa *ciphertext* akan ditampilkan pada text area bagian "*Ciphertext*", sedangkan nilai *keystream* dan IV ditampilkan pada text area bagian "*Keystream*" dan "IV". Program juga menyediakan *push button* "*Copy*" untuk menyalin hasil *keystream*, IV, maupun *ciphertext*, serta *push button* "*Print*" untuk mencetak hasil proses enkripsi. Selain itu, tersedia *push button* "Hapus" yang digunakan untuk mengosongkan seluruh edit text dan text area pada panel enkripsi.

Simulasi program pada tahap dekripsi dimulai dengan memasukkan *ciphertext* berupa karakter *Base64* ke dalam *edit text* pada bagian "Masukkan *Ciphertext* (*Base64*)". Selanjutnya, *keyword* yang sama dengan proses enkripsi dimasukkan pada bagian "Masukkan *Keyword* (*Base64*)". Proses dekripsi dijalankan dengan menekan *push button* "Dekripsi". Setelah tombol ditekan, program akan membentuk kembali *keystream* modifikasi deret Lucas dan Initialization Vector (IV) berdasarkan *keyword* yang diberikan, kemudian melakukan proses dekripsi menggunakan algoritma *Cipher Feedback* (CFB). Hasil dekripsi berupa *plaintext* akan ditampilkan pada text area bagian "*Plaintext*", sedangkan nilai *keystream* dan IV hasil pembentukan kembali ditampilkan pada bagian "*Keystream*" dan "IV". Program juga menyediakan *push button* "Copy" untuk menyalin hasil *plaintext* serta *push button* "Print" untuk mencetak hasil proses dekripsi. Selain itu, terdapat *push button* "Hapus" untuk membersihkan seluruh *edit text* dan *text area* pada panel dekripsi.

Perhitungan nilai Tingkat Perubahan Karakter (TPK) dilakukan melalui panel "TINGKAT PERUBAHAN KARAKTER (TPK)" dengan menekan *push button* "Hitung TPK". Setelah tombol ditekan, program secara otomatis mengambil *plaintext* dari proses enkripsi dan menghasilkan *ciphertext*

menggunakan metode CFB standar serta metode CFB berbasis modifikasi deret Lucas. Hasil *plaintext*, *ciphertext* CFB standar, dan *ciphertext* modifikasi akan ditampilkan pada text area yang tersedia. Selanjutnya, nilai TPK dari masing-masing metode ditampilkan pada bagian "Nilai TPK *Ciphertext* CipherFeedback CFB Standar" dan "Nilai TPK *Ciphertext* Modifikasi". Selain itu, program dilengkapi dengan *push button* "RESET ALL" yang berfungsi untuk menghapus seluruh isi edit text, text area, dan hasil perhitungan pada program secara keseluruhan.

Percobaan program dilakukan menggunakan *plaintext* "AKU" dan *keyword* "OK". Proses enkripsi menghasilkan *ciphertext* "JCT" pada metode *Cipher Feedback* (CFB) berbasis modifikasi deret Lucas, sedangkan metode CFB standar menghasilkan *ciphertext* "OOU". Selanjutnya, *ciphertext* "JCT" dan *keyword* "OK" digunakan kembali pada proses dekripsi sehingga diperoleh *plaintext* semula yaitu "AKU". Hasil percobaan program enkripsi, dekripsi, serta perhitungan nilai TPK ditampilkan pada Gambar 4.4 berikut.

Gambar 4.4 Percobaan program pada MATLAB

4.6 Analisis Hasil dan Pembahasan

Algoritma *Cipher Feedback* (CFB) pada penelitian ini merupakan algoritma kriptografi simetris yang menggunakan *keyword* yang sama pada proses enkripsi dan dekripsi. Perbedaan utama metode yang diusulkan dengan CFB standar terletak pada proses pembentukan *keystream* dan *Initialization Vector* (IV). Pada metode yang diusulkan, *keystream* dibentuk menggunakan modifikasi deret Lucas berdasarkan fungsi indikator karakter vokal dan nonvokal (konsonan, angka dan beberapa tanda lain) pada *keyword*, sedangkan IV dibentuk melalui operasi XOR secara berurutan terhadap *keyword*. Modifikasi tersebut menghasilkan proses enkripsi yang lebih kompleks dibandingkan CFB standar karena nilai *keystream* dan IV tidak hanya bergantung pada pengulangan *keyword*, tetapi juga dipengaruhi oleh karakteristik *keyword* yang digunakan.

Berdasarkan hasil pengujian, *ciphertext* yang dihasilkan menggunakan algoritma CFB berbasis modifikasi deret Lucas memiliki karakter yang lebih bervariasi dibandingkan *ciphertext* yang dihasilkan oleh algoritma CFB standar. Pada CFB standar, *keystream* dibentuk dari pengulangan *keyword* sehingga pola *ciphertext* masih cenderung dipengaruhi oleh pola *keyword* tersebut. Sebaliknya, pada metode yang diusulkan, pembentukan *keystream* menggunakan modifikasi deret Lucas dan pembentukan IV melalui operasi XOR *keyword* menghasilkan pola *ciphertext* yang lebih beragam sehingga hubungan antara *plaintext* dan *ciphertext* menjadi lebih sulit dikenali.

Sebagai ilustrasi, pada *plaintext* "AKU" dengan *keyword* "OK", algoritma CFB standar menghasilkan *ciphertext* "OOU", sedangkan metode CFB berbasis modifikasi deret Lucas menghasilkan *ciphertext* "JCT". *Ciphertext* "OOU" masih memiliki karakter "U" yang sama dengan *plaintext* sehingga masih menunjukkan kemiripan terhadap *plaintext*. Sebaliknya, *ciphertext* "JCT" tidak memiliki karakter yang sama dengan *plaintext*, sehingga menunjukkan tingkat perubahan karakter yang lebih tinggi. Contoh tersebut menggambarkan bagaimana modifikasi deret Lucas mampu menghasilkan *ciphertext* yang lebih beragam dibandingkan metode CFB standar.

Hasil pengujian menggunakan parameter Tingkat Perubahan Karakter (TPK) menunjukkan bahwa metode CFB berbasis modifikasi deret Lucas secara konsisten menghasilkan nilai TPK yang lebih kecil dibandingkan CFB standar. Pada contoh *plaintext* "AKU" dan *keyword* "OK", CFB standar menghasilkan nilai TPK sebesar 0,4286, sedangkan metode modifikasi menghasilkan nilai 0,3333. Hasil yang sama juga diperoleh pada beberapa variasi *plaintext* dan *keyword* lainnya yang disajikan pada Tabel 4.4 dan Lampiran 1, di mana nilai TPK metode modifikasi selalu lebih kecil dibandingkan CFB standar. Mengacu pada Maulidani (2023), nilai TPK yang lebih kecil menunjukkan bahwa hubungan antara *plaintext* dan *ciphertext* semakin kecil atau tidak signifikan karena tingkat perubahan karakter yang dihasilkan semakin tinggi. Dengan demikian, hasil pengujian pada seluruh sampel menunjukkan bahwa modifikasi deret Lucas pada pembentukan *keystream* mampu meningkatkan keragaman *ciphertext* dan memberikan tingkat keamanan yang lebih baik dibandingkan algoritma CFB standar.

BAB 5. KESIMPULAN DAN SARAN

Bab 5 berisi kesimpulan dari hasil penelitian dari implementasi dari kriptografi menggunakan algoritma *Cipher Feedback* (CFB) *Base64* berdasarkan modifikasi deret Lucas dan juga hasil TPK sebagai pengujian tingkat keamanannya serta saran sebagai rekomendasi untuk pengembangan penelitian selanjutnya.

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan, diperoleh kesimpulan bahwa modifikasi algoritma *Cipher Feedback* (CFB) menggunakan pembentukan *keystream* berbasis modifikasi deret Lucas berhasil diterapkan pada proses pengkodean teks berbasis *Base64*. *Keystream* yang dibentuk menggunakan aturan vokal dan nonvokal (konsonan, angka dan beberapa tanda baca lain) pada *keyword* menghasilkan pola *ciphertext* yang lebih bervariasi dibandingkan metode CFB standar. Hasil pengujian pada *plaintext* "AKU" dan *keyword* "OK" serta beberapa variasi *plaintext* dan *keyword* yang diuraikan pada Bab 4, menunjukkan bahwa metode yang diusulkan secara konsisten menghasilkan nilai TPK yang lebih kecil dibandingkan algoritma CFB standar. Kondisi tersebut menunjukkan bahwa hubungan antara *plaintext* dan *ciphertext* menjadi semakin kecil sehingga *ciphertext* yang dihasilkan lebih bervariasi dan lebih sulit diprediksi. Oleh karena itu, modifikasi deret Lucas yang diusulkan dapat meningkatkan tingkat perubahan karakter dan memberikan peningkatan keamanan dibandingkan algoritma CFB standar.

5.2 Saran

Penelitian selanjutnya disarankan untuk mengembangkan algoritma menggunakan ruang karakter yang lebih luas, seperti printable ASCII atau Unicode, agar karakter *ciphertext* yang dihasilkan semakin beragam dan bervariasi . Selain itu, metode pembentukan keystream dapat dikembangkan atau dikombinasikan dengan algoritma kriptografi lain untuk meningkatkan keamanan *ciphertext* yang dihasilkan.

DAFTAR PUSTAKA

Aliyah, A., Waseso, B., Kurniabudi, K., Meilano, R., Riza, F., Jinan, A., Gunawan, I. M. A. O., Muis, A., Intan, D. N., & Sembiring, A. (2025). *Dasar Dasar Cybersecurity*. Faaslib Serambi Media. https://books.google.co.id/books?id=slyPEQAAQBAJ

Astuti, L. F., Santoso, K. A., & Kamsyakawuni, A. (2019). Pengamanan Polyalphabetic dengan Affine. *Majalah Ilmiah Matematika Dan Statistika*, *19*(2), 95–103. https://jurnal.unej.ac.id/index.php/MIMS/index

Badan Siber dan Sandi Negara. (2024). Lanskap Keamanan Siber Indonesia. In *Id-SIRTII /CC – BSSN* (Issue 70). https://www.bssn.go.id/monitoring-keamanan-siber/

Dalimuthe, M. F. (2019). Perancangan Aplikasi Pengamanan File Dokumen Teks Dengan Menggunakan Algoritma Cipher Feedback. *Jurnal Teknik Informatika UNIKA Santo Thomas*, *04*, 70–75. http://ejournal.ust.ac.id/index.php/JTIUST/article/view/513

Dola, R., Jayadi, & Agung, R. R. (2024). Strategi Perlindungan Data Menggunakan Sistem Kriptografi Dalam Keamanan Informasi. *Journal of International Multidisciplinary Research*, *2*(6), 665–671. https://doi.org/10.62504/jimr679

Duman, M., & Duman, M. G. (2021). Encryption and Decryption of the Data by Using the Terms of the Lucas. *Journal of Science & Technology Research*, *9*, 1–7. https://doi.org/10.29130/dubited.825315

Gusti, D. R. K., Santoso, K. A., & Kamsyakawuni, A. (2020). Vigenere Cipher dengan Modifikasi Plaintext (Vigenere Cipher Using Plaintext Modification). *Majalah Ilmiah Matematika Dan Statistika*, *20*(1), 15–26.

Hanafi, Rachmawati, Suandi, I., & Dewi, A. F. (2024). *Sistem Komunikasi Data-*. CV. Andi Offset. https://books.google.co.id/books?id=K5UNEQAAQBAJ

Hoggatt, V. E. (1969). *Fibonacci and Lucas Numbers*. Houghton Mifflin Company.

Jayanti, S., Chittibabu, K., & Akkapeddi, C. S. (2022). *Pseudorandom Numbers Generation : An Implementation To A Secure Cryptosystem*. *20*(9), 944–947. https://doi.org/10.14704/nq.2022.20.9.NQ440105

Makhomah, R., Santoso, K. A., & Kamsyakawuni, A. (2021). Pengkodean Teks Menggunakan Kombinasi Hill Cipher dan Operasi XOR. *PRISMA, Prosiding Seminar Nasional Matematika*, *4*, 548–552.

Maulidani, D. B. A. (2023). *Analisis Tingkat Keamanan Cipherteks Berdasarkan Perubahan Karakter pada Plainteks*. Universitas Jember.

Meko, D. A. (2018). Perbandingan Algoritma DES, AES, IDEA Dan Blowfish dalam Enkripsi dan Dekripsi Data. *Jurnal Teknologi Terpadu*, *4*(1), 8–15.

Rohim, M. A., Santoso, K. A., & Hadi, A. F. (2022). Primary Key Encryption Using Hill Cipher Chain (Case Study : STIE Mandala PMB Site). *Proceedings of the International Conference on Mathematics, Geometry, Statistics, and Computation (IC-MaGeStiC 2021)*, *96*, 222–227.

Santoso, K. A., Pradjaningsih, A., & Delenia, E. (2024). Pengamanan Teks

dengan Kombinasi Metode *Electronic Code Book* (ECB) dan Kode *Seven Segment Display*. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *11*(1), 85–94. https://doi.org/10.25126/jtiik.20241117448

Stallings, W. (2017). Cryptography and Principles and Practice (7th ed.). In *Pearson*.

Tsypyschev, V., & Morgasov, I. (2022). Provable security of {CFB} mode of operation with external re-keying. *Cryptology EPrint Archive*, 1–15.

Zhu, X., Liu, F., Jia, Y., Xu, J., & Wang, P. (2025). Cryptanalysis of IAR-CTR and IAR-CFB and a fixing method. *Cybersecurity*, *8*(1). https://doi.org/10.1186/s42400-024-00312-x

LAMPIRAN

Lampiran 1 Perhitungan TPK pada beberapa *plaintext* dan *keyword*

Berikut merupakan tabel perhitungan TPK

Plaintext	*Keyword*	*Ciphertext*		Nilai TPK	
		CFB Standar	CFB Modifikasi	CFB Standar	CFB Modifikasi
AKUR	OK	OOUP	DIZO	0.4000	0.3333
AKUN	OK	OOUT	DIZS	0.4000	0.3333
AKU AKU	OK	OOUTguu0	JCTrn/0	0.3684	0.3333
AKUN AKU	OK	OOUTjptz	DIZSgynC	0.3636	0.3333
AKUN AKUR	AKU	AAAN5tn58	dWHN/rAhn	0.4286	0.3750
OK	AKUR	OO	HM	0.6667	0.3333
OKE	AKUR	OOe	TYZ	0.5000	0.3333
OKE OKE	AKU AKU	OOeeQQA	z45BDbA	0.3889	0.3333
OKE AKU	AKU OKE	OOeeQQA	tmnfTLA	0.4418	0.3500

Berikut merupakan *screenshoot* Hasil program pada matlab

Tautan : https://unej.id/PerhitunganTPKCFB

QR Code :

Lampiran 2 Program MATLAB

Tautan : https://unej.id/ProgramCFBLucas