IMPLEMENTASI ALGORITMA *LOGISTIC MAP* DAN *ARNOLD'S CAT MAP* PADA PENGAMANAN PESAN TEKS

SKRIPSI

OLEH
ROFIFAH
NIM. 220601110015

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

IMPLEMENTASI ALGORITMA *LOGISTIC MAP* DAN *ARNOLD'S CAT MAP* PADA PENGAMANAN PESAN TEKS

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
ROFIFAH
NIM. 220601110015

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

IMPLEMENTASI ALGORITMA *LOGISTIC MAP* DAN *ARNOLD'S CAT MAP* PADA PENGAMANAN PESAN TEKS

SKRIPSI

Oleh
Rofifah
NIM. 220601110015

Telah Disetujui Untuk Diuji

Malang, 12 Desember 2025

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Khudzaifah, M.Si.	Dr. Ach. Nashichuddin, M.A.
NIPPPK. 19900511 202321 1 029	NIP. 19730705 200003 1 002

Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M.Si.
NIP. 19800527 200801 1 012

IMPLEMENTASI ALGORITMA *LOGISTIC MAP* DAN *ARNOLD'S CAT MAP* PADA PENGAMANAN PESAN TEKS

SKRIPSI

Oleh
Rofifah
NIM. 220601110015

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima sebagai Salah satu Persyaratan
untuk Memperoleh Gelar Sarjana Matematika (S.Mat)

Tanggal 23 Desember 2025

Ketua Penguji	: Prof. Dr. H. Turmudi, M.Si., Ph.D.	.....
Anggota Penguji 1	: Intan Nisfulaila, M.Si.	.....
Anggota Penguji 2	: Muhammad Khudzaifah, M.Si.	.....
Anggota Penguji 3	: Dr. Ach. Nashichuddin, M.A.	.....

Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M.Si.
NIP. 19880527 200801 1 012

PERNYATAAN KEASLIAN TULISAN

Saya bertanda tangan di bawah ini

Nama : Rofifah

NIM : 220601110015

Program Studi : Matematika

Fakultas : Sains dan Teknologi

Judul Skripsi : Implementasi Algoritma *Logistic Map* dan *Arnold's Cat Map* pada Pengamanan Pesan Teks

Menyatakan bahwa skripsi yang saya tulis ini merupakan hasil karya sendiri, bukan mengambil alih tulisan atau pemikiran orang lain. Saya menyatakan skripsi ini sebagai pemikiran saya, kecuali dengan mencantumkan sumber kutipan pada daftar rujukan di halaman terakhir. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil plagiasi atau tiruan orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 23 Desember 2025

Rofifah

NIM. 220601110015

MOTTO

“Kesulitan membuat kita lelah, namun bukan untuk berhenti. Teruslah berusaha, sebab keberuntungan selalu menghampiri mereka yang tidak menyerah.”

-Rofifah-

“*Never let anyone, including yourself, belittle your dreams.”*

-Joshua Hong-

PERSEMBAHAN

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Skripi ini dipersembahkan kepada:

"Ibunda Siti Aminah, yang sudah mendukung, membantu, memberi semangat, dan tidak lupa selalu memanjatkan doanya untuk kelancaran penulis dalam menyelesaikan skripsi ini, Bapak Riawan Dardianto, yang penuh kesabaran selalu memberikan arahan, dorongan, serta keteguhan hati kepada penulis dalam melalui setiap proses dan tantangan selama penyusunan skripsi ini. Persembahan ini juga penulis dedikasikan kepada seluruh keluarga tercinta yang senantiasa memberikan dukungan, serta menjadi sumber kekuatan dalam setiap langkah perjalanan akademik penulis. Serta tidak lupa untuk diri saya sendiri yang sudah hebat bisa sampai menyelesaikan skripsi ini."

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Segala puji bagi Allah *Subhanahu Wa Ta'ala* yang telah melimpahkan rahmat, taufik, dan hidayah-Nya, sehingga penulis senantiasa diberikan kesehatan dan kesempatan untuk menyelesaikan skripsi dengan judul "Implementasi Algoritma *Logistic Map* dan *Arnold's Cat Map* pada Pengamanan Pesan Teks" sebagai salah satu syarat untuk memperoleh gelar sarjana dalam bidang matematika di Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Sholawat dan salam senantiasa tercurahkan kepada junjungan kita Nabi Muhammad *Shallallahu 'Alaihi Wasallam*, yang telah membawa kita dari zaman jahiliah menuju zaman yang penuh rahmat yakni zaman Islamiah.

Penulis mengucapkan terima kasih kepada berbagai pihak yang telah membimbing dan membantu dalam penyusunan skripsi ini, sehingga skripsi ini dapat diselesaikan sesuai yang diharapkan. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si., CHARM., CRMP selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim.
2. Dr. Agus Mulyono, M.Kes., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim.
3. Dr. Fachrur Rozi, M.Si., selaku Ketua Program Studi Matematika, Universitas Islam Negeri Maulana Malik Ibrahim.
4. Muhammad Khudzaifah, M.Si., selaku Dosen pembimbing I yang senantiasa memberikan berbagai pengetahuan, nasehat, motivasi, dan arahan selama proses penyusunan skripsi ini.
5. Dr. Ach. Nashichuddin, M.A., selaku Dosen pembimbing II yang telah memberikan arahan dan masukan yang sangat bermanfaat dalam penulisan skripsi ini.
6. Prof. Dr. H. Turmudi, M.Si., Ph.D., selaku Ketua Penguji yang telah meluangkan waktu, tenaga, dan perhatian dalam memberikan masukan, arahan, serta koreksi yang sangat berharga demi penyempurnaan skripsi ini. Saran dan pandangan beliau menjadi dorongan penting bagi penulis untuk meningkatkan kualitas penelitian ini.

7. Intan Nisfulaila, M.Si., selaku Penguji I yang dengan penuh kesabaran memberikan kritik, saran, serta bimbingan selama proses ujian dan perbaikan skripsi. Setiap masukan yang diberikan sangat membantu penulis dalam memperbaiki kekurangan dan menyempurnakan penulisan skripsi ini.
8. Seluruh dosen Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim yang telah memberikan ilmu, bimbingan, serta pengalaman berharga selama masa perkuliahan.
9. Ibu saya yang bernama Siti Aminah yang senantiasa memberikan dukungan berupa doanya, membantu menyusun sebagian kata-kata dalam skripsi ini, mendengarkan keluh dan kesah penulis selama menulis skripsi ini dan beliau adalah salah satu sumber motivasi utama untuk segera menyelesaikan skripsi ini.
10. Bapak saya yang bernama Riawan Dardianto yang senantiasa memberikan doa, dukungan, semangat, serta kasih sayang tanpa henti. Beliau adalah salah satu sumber motivasi utama untuk segera menyelesaikan skripsi ini.
11. Seluruh mahasiswa angkatan 2022, khususnya kelas A yang sudah memberi semangat penulis untuk segera menyelesaikan skripsi ini dan selalu mendukung satu sama lain.
12. Teman-temanku, Ica, Zulfa, dan Firda, yang sudah selalu memberi semangat, masukan dari awal mahasiswa baru, terimakasih banyak sudah selalu menemani penulis dari ujian seminar proposal hingga terselesaikannya skripsi ini hingga akhir.
13. Cimit, kucing baruku terimakasih sudah menemani penulis, menghibur penulis disaat mengerjakan skripsi ini hingga akhir.
14. Grup Idol bernama "Seventeen" terimakasih banyak sudah sangat menghibur penulis dalam masa-masa mengerjakan skripsi ini, dengan konten-konten kocaknya di GOSE, dan juga lagu-lagu indahnya yang menemani penulis mengerjakan skripsinya.
15. Dan seluruh pihak yang sudah terlibat secara langsung maupun tidak langsung yang tidak bisa penulis tuliskan satu per-satu.

Dengan seluruh kerendahan hati, peneliti menyadari terciptanya skripsi ini masih belum bisa dikatakan sempurna. Maka dari itu, peneliti berharap kritik dan

saran yang membangun demi terlahirnya penyempurnaan skripsi ini. Semoga karya ini mampu memberikan kebermanfaatan untuk orang banyak. *Aamiin ya Rabbal 'Alamiin*.

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Malang, 23 Desember 2025

Penulis

DAFTAR ISI

HALAMAN JUDUL ..... **i**
HALAMAN PENGAJUAN ..... **ii**
HALAMAN PERSETUJUAN .....
..... Error! Bookmark not defined.
HALAMAN PENGESAHAN .....
..... Error! Bookmark not defined.
PERNYATAAN KEASLIAN TULISAN .....
..... Error! Bookmark not defined.
MOTTO ..... **iv**
PERSEMBAHAN ..... **vii**
KATA PENGANTAR ..... **viii**
DAFTAR ISI ..... **xi**
DAFTAR TABEL ..... **xii**
DAFTAR GAMBAR ..... **xiii**
DAFTAR SIMBOL ..... **xiv**
ABSTRAK ..... **xv**
ABSTRACT ..... **xvi**
مستخلص البحث ..... **xvii**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Tujuan Penelitian 4
1.4 Manfaat Penelitian 5
1.5 Batasan Masalah 6
1.6 Definisi Istilah 6
BAB II KAJIAN TEORI **10**
2.1 Kriptografi dan Keamanan Informasi 10
2.1.1 Pengertian Kriptografi 10
2.1.2 Jenis Kriptografi 12
2.2 Teori *Chaos* dalam Kriptografi 14
2.2.1 Konsep Dasar *Chaos* 14
2.2.2 Keunggulan *Chaos* untuk Enkripsi Data Teks 16
2.3 Algoritma *Logistic Map* 17
2.3.1 *Logistic Map* sebagai Penghasil *Keystream* 19
2.3.2 Operasi XOR Berbasis *Logistic Map* 19
2.3.3 Permutasi Baris dan Kolom Berbasis *Logistic Map* 22
2.4 Algoritma *Arnold's Cat Map* 25
2.4.1 Kongruensi Matriks 28
2.5 Amanah dalam Islam 30
2.6 Kajian Topik dengan Teori Pendukung 33
BAB III METODE PENELITIAN **35**
3.1 Jenis Penelitian 35
3.2 Tahapan Penelitian 35
3.2.1 Proses Pembentukan Kunci (*Keystream*) 35

3.2.2 Proses Enkripsi Pesan Teks 36
3.2.3 Proses Dekripsi Pesan Teks 37
3.3 *Flowchart* 39
BAB IV HASIL DAN PEMBAHASAN **41**
4.1 Gambaran Umum Implementasi 41
4.2 Implementasi Pembangkitan Kunci (*Keystream*) 43
4.3 Implementasi Proses Enkripsi Pesan Teks 49
4.4 Implementasi Proses Dekripsi Pesan Teks 67
4.5 Integrasi Nilai Amanah dalam Pengamanan Pesan Teks 77
BAB V PENUTUP **80**
5.1 Kesimpulan 80
5.2 Saran 81
DAFTAR PUSTAKA **83**
LAMPIRAN **86**
RIWAYAT HIDUP **106**

DAFTAR TABEL

Tabel 2.1 Operasi XOR..... 20
Tabel 4.1 Konversi Karakter Alfabet Menjadi Kode ASCII (desimal) 44
Tabel 4.2 *Output* Kode Python 48
Tabel 4.3 Konversi Kode ASCII (desimal) Ke Biner 8 Bit 51

DAFTAR GAMBAR

Gambar 2.1 Proses Enkripsi dan Dekripsi 11
Gambar 3.1 *Flowchart* Tahapan Enkripsi..... 39
Gambar 3.2 *Flowchart* Tahapan Dekripsi..... 40
Gambar 4.1 Kode Python Pembangkitan Kunci 46
Gambar 4.2 Hasil Iterasi Ke-6 Hingga Ke-20..... 48

DAFTAR SIMBOL

x_n	: Nilai populasi atau keadaan sistem pada iterasi ke–n
$x(n+1)$	: Nilai populasi atau keadaan sistem pada iterasi berikutnya
x_0	: Kondisi awal (*initial condition*) pada *Logistic Map*
r	: Parameter pengendali (*control parameter*) pada *Logistic Map*
n	: Indeks waktu diskret atau jumlah iterasi
$f(x)$	: Fungsi *Logistic Map* $f(x) = r \cdot x\,(1 - x)$
K	: Ruang kunci (*key space*) yang digunakan dalam sistem
k_i	: Elemen ke–i dari *keystream* yang dihasilkan *Logistic Map*
P	: Plainteks (pesan asli yang belum dienkripsi)
C	: Cipherteks (pesan yang sudah terenkripsi)
E	: Fungsi enkripsi
D	: Fungsi dekripsi
$\oplus$	: Operasi logika XOR (*Exclusive–OR*)
(x, y)	: Koordinat titik sebelum transformasi pada *Arnold's Cat Map*
(x', y')	: Koordinat titik setelah transformasi pada *Arnold's Cat Map*
N	: Ukuran bidang atau dimensi matriks pada *Arnold's Cat Map*
M	: Matriks *ciphertext* hasil operasi XOR yang akan diacak
$ASCII\ (P_i)$	: Kode ASCII dari karakter plainteks ke–i
Key Stream (i)	: Deret bit *pseudo*-acak hasil iterasi ke-i dari *Logistic Map*
Padding	: Data tambahan untuk menyesuaikan ukuran matriks persegi

ABSTRAK

Rofifah, Rofifah. 2025. **Implementasi Algoritma *Logistic Map* dan *Arnold's Cat Map* pada Pengamanan Pesan Teks.** Skripsi. Jurusan Matematika fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Muhammad Khudzaifah, M.Si. (II) Ach. Nashichuddin, M.A.

Kata Kunci: kriptografi, teori *chaos*, algoritma *Logistic Map*, algoritma *Arnold's Cat Map*, enkripsi pesan teks, XOR.

Perkembangan teknologi komunikasi digital telah meningkatkan kebutuhan akan keamanan pesan teks terhadap ancaman seperti penyadapan, pencurian data, dan manipulasi informasi. Penelitian ini bertujuan untuk mengimplementasikan algoritma *Logistic Map* dan *Arnold's Cat Map* pada sistem pengamanan pesan teks digital. *Logistic Map* digunakan sebagai pembangkit *keystream* melalui proses iterasi yang menghasilkan deret bit *pseudo*-acak, kemudian *keystream* tersebut dipermutasi menggunakan pergeseran baris dan kolom. Selanjutnya, proses enkripsi dilakukan dengan operasi XOR antara *plaintext* dan *keystream* yang telah dipermutasi. Untuk memperkuat difusi, *ciphertext* hasil XOR diacak kembali menggunakan algoritma *Arnold's Cat Map* sehingga menghasilkan pola *ciphertext* yang tidak menampilkan keteraturan statistik. Kedua algoritma tersebut didasarkan pada prinsip teori *chaos* yang memiliki sifat deterministik namun sangat sensitif terhadap kondisi awal. Karakteristik ini memungkinkan sistem menghasilkan kunci *pseudo*-acak yang sulit diprediksi, sehingga meningkatkan tingkat *confusion* dan *diffusion* pada proses enkripsi. Hasil penelitian menunjukkan bahwa kombinasi *Logistic Map* dan *Arnold's Cat Map* mampu menghasilkan sistem enkripsi dengan sensitivitas tinggi terhadap perubahan parameter awal dan mampu mengembalikan pesan asli secara akurat melalui proses dekripsi. Dengan demikian, implementasi kedua algoritma *chaos* ini dapat dijadikan pendekatan alternatif dalam pengamanan pesan teks digital.

ABSTRACT

Rofifah, Rofifah. 2025. **Implementation of Logistic Map and Arnold's Cat Map Algorithms in text massage security**. Thesis. Department of Mathematics, Faculty of Science and Technology, Universitas Islam Hegeri Maulanan Malik Ibrahim Malang. Advisors: (I) Muhammad Khudzaifah, M.Si. (II) Ach. Nashichuddin, M.A.

Keywords: cryptography, chaos theory, Logistic Map algorithm, Arnold's Cat Map algorithm, text message encryption, XOR.

The development of digital communication technology has increased the need for text message security against threats such as wiretapping, data theft, and information manipulation. This study aims to implement the Logistic Map and Arnold's Cat Map algorithms in a digital text message security system. The Logistic Map is used as a keystream generator through an iterative process that produces a pseudo-random bit sequence, which is then permuted using row and column shifts. Next, the encryption process is carried out by performing an XOR operation between the plaintext and the permuted keystream. To strengthen diffusion, the XOR ciphertext is randomized again using the Arnold's Cat Map algorithm, resulting in a ciphertext pattern that does not display statistical regularity. Both algorithms are based on the principles of chaos theory, which is deterministic but highly sensitive to initial conditions. This characteristic allows the system to generate pseudo-random keys that are difficult to predict, thereby increasing the level of confusion and diffusion in the encryption process. The results of the study show that the combination of the Logistic Map and Arnold's Cat Map is capable of producing an encryption system with high sensitivity to changes in initial parameters and is able to accurately restore the original message through the decryption process. Thus, the implementation of these two chaos algorithms can be used as an alternative approach in securing digital text messages.

مستخلص البحث

روفيفة ، روفيفة. ٢٠٢٥. **تطبيق خريطة لوجستية وخوارزميات خريطة قطة أرنولد في أمن الرسائل النصية**. البحث الجامعي. قسم الرياضيات، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم لإسلامية الحكومية مالانج. ا المشرف :(١) محمد خديفة، الماجستير (٢) احمد ناصح الدّين، الماجستير.

الكلمات الأسا سية: التشفير، نظرية الفوضى، خوارزمية الخريطة اللوجستية، خوارزمية خريطة قطة أرنولد، تشفير الرسائل النصية، XOR.

أدى تطور تكنولوجيا الاتصالات الرقمية إلى زيادة الحاجة إلى تأمين الرسائل النصية ضد التهديدات مثل التنصت وسرقة البيانات والتلاعب بالمعلومات. تهدف هذه الدراسة إلى تطبيق خوارزميات الخريطة اللوجستية وخريطة قط أرنولد في نظام أمان الرسائل النصية الرقمية. تُستخدم الخريطة اللوجستية كمولد تيار مفاتيح من خلال عملية تكرارية تنتج تسلسل بتات عشوائي زائف، والذي يتم بعد ذلك تبديله باستخدام تحويلات الصفوف والأعمدة. بعد ذلك، يتم تنفيذ عملية التشفير عن طريق إجراء عملية XOR بين النص العادي وتدفق المفاتيح المبدل. لتعزيز الانتشار، يتم تشويش النص المشفر XOR مرة أخرى باستخدام خوارزمية خريطة قطة أرنولد، مما ينتج عنه نمط نص مشفر لا يعرض انتظامًا إحصائيًا. تستند كلتا الخوارزميتين إلى مبادئ نظرية الفوضى، وهي نظرية حتمية ولكنها حساسة للغاية للظروف الأولية. تتيح هذه الخاصية للنظام إنشاء مفاتيح عشوائية زائفة يصعب التنبؤ بها، مما يزيد من مستوى الارتباك والانتشار في عملية التشفير. تظهر نتائج الدراسة أن الجمع بين خريطة لوجستية وخريطة قطة أرنولد قادر على إنتاج نظام تشفير عالي الحساسية للتغيرات في المعلمات الأولية وقادر على استعادة الرسالة الأصلية بدقة من خلال عملية فك التشفير. وبالتالي، يمكن استخدام تنفيذ هاتين الخوارزميتين الفوضويتين كنهج بديل في تأمين الرسائل النصية الرقمية.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi, informasi, dan komunikasi (TIK) telah memicu perubahan besar dalam cara berinteraksi masyarakat, khususnya melalui pemanfaatan aplikasi *instan messaging* seperti WhatsApp, Line, dan Telegram yang saat ini berfungsi sebagai alat utama komunikasi harian (Taipale & Farinosi, 2018). Platform-platform tersebut tidak hanya memungkinkan pertukaran pesan teks, tetapi juga menunjang komunikasi multimodal berupa gambar, audio, dan video, yang meningkatkan keintiman sosial secara *online* (Taipale & Farinosi, 2018). Namun, kemajuan ini juga meningkatkan risiko kemanan pesan teks, seperti penyadapan komunikasi, pencurian data, dan manipulasi pesan, yang menyangkut kerahasiaan, integrasi, dan autentikasi informasi (Lesnussa & Tomasouw, 2024)

Kriptografi hadir sebagai solusi utama, namun metode konvensional seperti *Advanced Encryption Standard* (AES) atau *Rivest Shamir Adleman* (RSA) memiliki keterbatasan dalam menghadapi serangan modern seperti diferensial, *brute-force*, dan analisis statistik (Ramakrishna & Ali Shaik, 2025). alternatif yang potensial adalah algoritma berbasis teori *chaos*, yang bersifat deterministik namun sangat sensitif terhadap kondisi awal, sehingga mampu menghasilkan *keystream* acak yang kuat (Zhang & Liu, 2023). Dalam konteks pengamanan pesan teks, *Logistic Map* digunakan sebagai pembangkit *keystream* melalui deret *pseudo*-acak yang dikombinasikan dengan pesan via operasi *Exclusive-OR* (XOR), yaitu proses logika biner yang menghasilkan nilai 1 jika kedua bit berbeda dan 0 jika keduanya sama, untuk melakukan substitusi, sedangkan *Arnold's Cat Map* mengacak posisi karakter

setelah *Exclusive-OR* (XOR) guna memperkuat pengacakan (Alkhonaini et al., 2024).

Penelitian sebelumnya menunjukkan efektivitas *Logistic Map* dalam pengacakan dan ketahanan terhadap serangan (Arif et al., 2022), serta potensi integrasi *multi-chaotic maps* (Alkhonaini et al., 2024). Namun, penerapan langsung pada pesan teks masih jarang, menimbulkan kesenjangan penelitian karena kajian dominan bersifat kuantitatif dan minim eksplorasi kualitatif tentang efektivitas serta tantangan implementasi (Tazi et al., 2024). padahal, pesan teks memiliki karakteristik unik seperti ukuran data kecil dan struktur linier, sehingga diperlukan implementasi spesifik *Logistic Map* dan *Arnold's Cat Map* untuk meningkatkan keamanan komunikasi berbasis teks.

Pesan teks sering memuat informasi sensitif, seperti kode verifikasi dan data pribadi (Munir, 2021), sehingga sistem penagamanan perlu dikembangkan dengan efisiensi komputasi tinggi dan ketahanan terhadap serangan *brute-force* melalui ruang kunci besar dari algoritma *chaos* (C. Irawan & Rachmawanto, 2022). Penggabungan *Logistic Map* dan *Arnold's Cat Map* menghasilkan pola enkripsi hybrid yang lebih tahan terhadap serangan kriptografi kontemporer (Mukherjee, 2023).

Menurut perspektif Islam, menjaga kerahasiaan dan amanah informasi merupakan bagian moral yang sangat penting. Al-Qur'an menegaskan pada QS. Al-Mu'minun 08 (Kemenag, 2022):

وَالَّذِيْنَ هُمْ لِاَمٰنٰتِهِمْ وَعَهْدِهِمْ رَاعُوْنَۙ ﴿٨﴾

Artinya: *"Dan (sungguh beruntung) orang yang memelihara amanat-amanat dan janjinya,"*

Ayat ini, menunjukkan bahwa amanat itu sangat penting dalam kehidupan sesama manusia agar dapat hidup dengan adil dan makmur, karena setiap orang menjalankan amanahnya sesuai dengan porsinya masing-masing (Hamka, 2003). Demikian juga, prinsip menjaga kerahasiaan komunikasi ditekankan dalam QS. An-Nur 27 (Kemenag, 2022):

يٰٓاَيُّهَا الَّذِيْنَ اٰمَنُوْا لَا تَدْخُلُوْا بُيُوْتًا غَيْرَ بُيُوْتِكُمْ حَتّٰى تَسْتَأْنِسُوْا وَتُسَلِّمُوْا عَلٰٓى اَهْلِهَاۗ ذٰلِكُمْ خَيْرٌ لَّكُمْ لَعَلَّكُمْ تَذَكَّرُوْنَ ٢٧

Artinya: *"Wahai orang-orang yang beriman! Janganlah kamu memasuki rumah yang bukan rumahmu sebelum meminta izin dan memberi salam kepada penghuninya. Yang demikian itu lebih baik bagimu, agar kamu (selalu) ingat."*

QS. An-Nur ayat 27 dipahami sebagai aturan adab Islam untuk menghormati privasi, di mana seorang mu'min dilarang memasuki rumah orang lain tanpa izin dan salam. Hamka menegaskan bahwa rumah orang adalah tempat menyimpan rahasia pribadi, mulai dari kondisi sederhana, keterbatasan rezeki, hingga urusan rumah tangga, yang tidak boleh diketahui orang lain tanpa seizin pemiliknya. Ayat ini menegaskan prinsip hak kedaulatan pribadi atas ruang privat, jauh sebelum konsep modern seperti *hak asasi* manusia atau *privacy rights* dikenal. Prinsip tersebut sejalan dengan amanah dalam Islam, yakni menjaga sesuatu yang bukan hak kita, menghormati privasi orang lain, dan tidak melanggar batas yang ditetapkan Allah. Dalam konteks modern, amanah ini relevan dengan menjaga keamanan data dan pesan digital, di mana informasi pribadi diibaratkan sebagai rahasia rumah tangga yang tidak boleh diakses tanpa izin pemiliknya (Hamka, 2003)

Penelitian ini memiliki tujuan untuk mengimplementasikan algoritma *Logistic Map* dan *Arnold's Cat Map* dalam pengamanan pesan teks, serta memberikan rekomendasi untuk pengembangan teknik enkripsi teks yang lebih

baik. Selain aspek teknis, penelitian ini mengintegrasikan nilai-nilai etika dan amanah Islam dalam pengembangan teknologi keamanan informasi, sehingga kontribusi mencakup penguatan sistem kriptografi dan penanaman prinsip amanah serta tanggung jawab sesuai ajaran Islam. Integrasi ini menjadikan penelitian lebih komprehensif dan relevan secara akademis maupun sosial.

1.2 Rumusan Masalah

Berdasarkan pengetahuan yang diberikan dalam latar belakang, penelitian ini merumuskan rumusan masalah sebagai berikut:

1. Bagaimana proses pembangkitan kunci (*keystream*) menggunakan algoritma *Logistic Map* untuk pengamanan pesan teks?
2. Bagaimana proses enkripsi pesan teks dengan memanfaatkan *keystream* yang dihasilkan dari algoritma *Logistic Map* dan proses
3. Bagaimana proses dekripsi pesan teks agar dapat mengembalikan pesan asli secara akurat menggunakan algoritma *Logistic Map* dan *Arnold's Cat Map*?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan yang ingin dicapai adalah sebagai berikut:

1. Menjelaskan dan mengimplementasikan proses pembangkitan kunci (*keystream*) menggunakan algoritma *Logistic Map* untuk pengamanan pesan teks.
2. Menjelaskan proses enkripsi pesan teks dengan memanfaatkan *keystream* hasil dari algoritma *Logistic Map* dan pengacakan menggunakan algoritma *Arnold's Cat Map*.

3. Menerapkan proses dekripsi pesan teks agar pesan asli dapat dipulihkan secara akurat menggunakan algoritma *Logistic Map* dan *Arnold's Cat Map*.

1.4 Manfaat Penelitian

Penelitian ini memiliki beberapa manfaat, beberapa diantaranya adalah sebagai berikut:

1. Manfaat Teoritis

   Penelitian ini memberikan kontribusi teoritis dengan memperluas pemahaman mengenai penerapan algoritma *Logistic Map* dan *Arnold's Cat Map* dalam pengamanan pesan teks. Hasil penelitian ini dapat memperkaya studi kriptografi yang berbasis teori *chaos*, terutama dalam aspek enkripsi dan dekripsi pesan teks digital. Selain itu, penelitian ini juga menjadi dasar bagi peneliti berikutnya untuk mengembangkan metode enkripsi teks yang lebih kuat, serta mengintegrasikan nilai-nilai etika dalam teknologi keamanan informasi.

2. Manfaat Praktis

   a. Bagi peneliti

      i. Peneliti dapat mendalami bidang kriptografi berbasis teori *chaos* dan implementasi algoritma *Logistic Map* dan *Arnold's Cat Map* secara lebih mendalam.

      ii. Peneliti memperoleh pengalaman praktis dalam merancang dan menguji sistem enkripsi dan dekripsi pesan teks yang aman.

      iii. Peneliti dapat mengintegrasikan aspek teknis dan nilai moral (amanah) dalam pengembangan teknologi keamanan informasi.

b. Bagi pembaca

i. Pembaca akan mendapatkan pemahaman lebih mendalam mengenai proses pembangkitan kunci, mekanisme enkripsi, dan dekripsi pesan teks menggunakan algoritma *chaos*.

ii. Penelitian ini dapat menginspirasi pembaca untuk melakukan penelitian lanjutan dalam pengembangan teknik enkripsi pesan teks yang lebih kuat, dan beretika.

1.5 Batasan Masalah

Adapun beberapa batasan masalah untuk penelitian ini, beberapa diantaranya adalah sebagai berikut:

1. Proses enkripsi dan dekripsi dilakukan dengan asumsi bahwa kunci awal (*initial conditions*) diketahui oleh pihak yang berwenang untuk memastikan reproduksibilitas hasil dekripsi.
2. Penelitian ini membatasi pembahasan pada integrasi nilai-nilai etika dan amanah dalam konteks pengembangan teknologi keamanan informasi, tanpa membahas aspek hukum atau kebijakan terkait keamanan data.
3. Pendekatan penelitian yang digunakan bersifat kualitatif dengan fokus pada proses implementasi algoritma dan proses enkripsi-dekripsi, tanpa melakukan pengujian kuantitatif terhadap performa atau keamanan sistem.

1.6 Definisi Istilah

1. Kriptografi

   Ilmu dan seni mengamankan informasi melalui teknik matematis, termasuk enkripsi, dekripsi, autentikasi, serta integritas data.

2. Teori *Chaos*

   Cabang matematika yang mempelajari sistem dinamis yang sangat sensitif terhadap kondisi awal, menghasilkan perilaku yang tampak acak namun deterministik, yang dimanfaatkan dalam kriptografi untuk meningkatkan keamanan

3. *Logistic Map*

   Sebuah fungsi matematis dalam teori *chaos* yang digunakan untuk menghasilkan deret bilangan *pseudo*-acak yang sangat sensitif terhadap kondisi awal, sering diaplikasikan dalam pembangkitan kunci kriptografi.

4. *Arnold's Cat Map*

   Sebuah transformasi matematis yang memetakan titik-titik dalam bidang dua dimensi secara periodik dan kaotik, digunakan untuk mengacak data dalam proses enkripsi agar pola data lebih sulit ditebak.

5. *Keystream* (Kunci Alir)

   Deret bilangan yang dihasilkan dari algoritma tertentu (dalam penelitian ini *Logistic Map*) yang digunakan sebagai kunci dalam proses enkripsi dan dekripsi teks.

6. Enkripsi

   Proses mengubah pesan asli (plainteks) menjadi bentuk tidak terbaca (cipherteks) menggunakan algoritma dan kunci tertentu untuk menjaga kerahasiaan informasi.

7. Dekripsi

   Proses mengembalikan cipherteks menjadi pesan asli (plainteks) dengan menggunakan algoritma dan kunci yang sesuai dengan proses enkripsi.

8. Plainteks atau *Plaintext*

   Pesan asli yang masih dapat dibaca dan dipahami sebelum melalui proses enkripsi.

9. Cipherteks atau *Ciphertext*

   Hasil dari proses enkripsi berupa pesan dalam bentuk tidak terbaca atau acak, yang hanya dapat dikembalikan menjadi *plaintext* melalui dekripsi dengan kunci yang benar.

10. *American Standard Code for Information Interchange* (ASCII)

    Standar pengkodean karakter yang merepresentasikan teks dalam bentuk bilangan biner, digunakan untuk memproses pesan teks digital dalam komputer.

11. *Exclusive-OR* (XOR)

    Operasi logika biner yang menghasilkan nilai benar (1) jika input berbeda, dan salah (0) jika input sama, sering digunakan dalam proses kriptografi untuk menggabungkan *keystream* dengan *plaintext* dalam enkripsi maupun dekripsi.

12. *Public Key*

    Kunci publik merupakan kunci untuk enkripsi kepada publik (tidak rahasia), misalnya disimpan di dalam repositori yang dapat diakses oleh publik.

13. *Private Key*

    Kunci privat merupakan kunci privat yang rahasia, hanya pemilik kunci privat itu sendiri yang mengetahui kuncinya sendiri.

14. Orbit

    Rangkaian nilai atau posisi yang dihasilkan dengan mengiterasi fungsi tertentu berulang kali dari satu titik awal.

15. Ruang Kunci

    Himpunan semua kemungkinan nilai kunci (*key*) yang dapat digunakan oleh sistem .

BAB II

KAJIAN TEORI

2.1 Kriptografi dan Keamanan Informasi

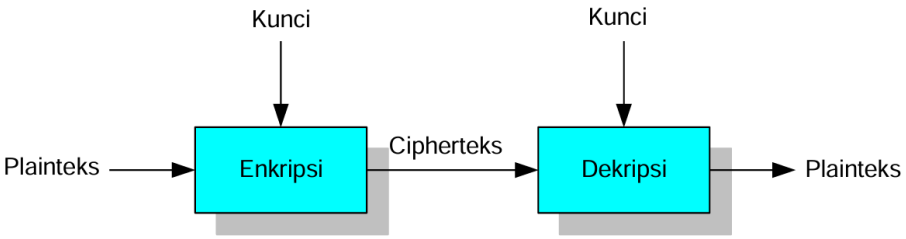
2.1.1 Pengertian Kriptografi

Istilah kriptografi berasal dari bahasa Yunani, yaitu *krupto* yang berarti "tersembunyi" dan *graphia* yang berarti "tulisan", maka secara literal berarti "tulisan rahasia" atau *secret writing*. Dalam pengertian klasik, kriptografi merupakan ilmu dan seni yang bertujuan menjaga kerahasiaan pesan dengan menyembunyikan atau mengamankan dengan sandi tertentu ke dalam bentuk yang tidak semua orang bisa memahami maknanya (Munir, 2008).

Kriptografi juga berperan dalam menjaga integritas (*integrity*) pesan, memastikan bahwa pesan tidak mengalami modifikasi selama proses pengiriman. Selanjutnya, mekanisme autentikasi (*authentication*) digunakan untuk memverifikasi identitas pengirim dan penerima pesan secara akurat. Terakhir, fungsi *non-repudiation* berperan dalam mencegah pengirim agar tidak dapat menyangkal keterlibatannya dalam pengiriman pesan (Katz, Jonathan and Lindell, 2015). Bidang kriptografi menggunakan beberapa istilah, diantaranya:

1. *Plaintext* (M) adalah pesan asli yang akan dikirmkan.
2. *Ciphertext* (C) adalah pesan yang telah disandikan atau ter-enkripsi.
3. Enkripsi ($fungsi\ E$) adalah proses menyandikan atau mengamankan plainteks menjadi cipherteks.
4. Dekripsi ($fungsi\ D$) adalah proses mengembalikan cipherteks menjadi plainteks awal.
5. Kunci adalah suatu bilangan rahasia yang digunakan dalam proses enkripsi.

Enkripsi dan Dekripsi merupakan aplikasi utama dari kriptografi yang bertujuan membuat data tidak dapat dipahami agar tetap rahasia dan dekripsi yang merupakan proses sebaliknya yang mengembalikan *ciphertext* ke bentuk awal agar pesan bisa terbaca kembali (Aumasson, 2017). Berikut gambaran proses enkripsi dan dekripsi:



Gambar 2.1 Proses Enkripsi dan Dekripsi

Algoritma kriptografi umumnya terbentuk dari serangkaian langkah-langkah terstruktur yang diterapkan untuk proses enkripsi dan dekripsi. Proses enkripsi serta dekripsi data melibatkan berbagai elemen penting, termasuk kunci yang melindungi kerahasiaan *ciphertext* serta mencegah terjadinya pelanggaran data, dan juga algoritma yang membentuk dasar sistem keamanan. Dalam ilmu kriptografi dibagi menjadi dua jenis umum, yaitu simetris dan asimetris (kunci publik). Menurut (Alfred, 2014) simetris (*symmetric-key*) adalah kunci yang sama pada proses enkripsi dan dekripsi, jadi ada beberapa nama lain yang mengartikan *symmetric-key*, yaitu *single-key*, *one-key*, dan *conventional encryption*. Asimetris adalah skema kriptografi yang menggunakan sepasang kunci berbeda, dalam artian kunci pada enkripsi dan dekripsi berbeda atau biasa disebut dengan kunci publik (*public-key*) dan kunci privat (*private-key*).

2.1.2 Jenis Kriptografi

Sebagai bagian penting dari ilmu yang berfokus pada perlindungan keamanan data digital kriptografi memiliki serangkaian langkah-langkah yang terstruktur yang diterapkan pada proses enkripsi dan dekripsi. Menurut (Alfred, 2014) ada dua jenis umum teknik kriptografi, yaitu:

1. Kunci Simetris

   Dalam sebagian besar sistem enkripsi kunci simetris, kunci yang digunakan untuk enkripsi dan dekripsi pada dasarnya memiliki fungsi yang sama. Oleh karena itu, istilah kunci simetris menjadi tepat. Suatu sistem disebut menggunakan kunci simetris menjadi tepat. Suatu sistem disebut menggunakan kunci simetris apabila kunci untuk enkripsi (E) dan dekripsi (D) dapat saling dihitung dengan mudah menggunakan komputer. Ini membuat kedua kunci saling bergantung, dan biasanya keduanya sama atau sangat mirip. Dengan kunci yang sama maka kecepatan dalam proses enkripsi dan dekripsi tinggi, serta kunci relatif pendek, namun harus diperhatikan dalam kerahasiaan kunci di kedua pihak, yaitu pihak pengirim maupun penerima, dan kunci perlu sering diganti untuk menghindari pihak yang tidak seharusnya menerima pesan ataupun data yang dikirim oleh pengirim. Adapun salah satu penggunaan kunci simetris pada *stream* cipher. *Stream* cipher termasuk kategori utama dalam skema enkripsi kunci simetris. Dalam pengertian tertentu, ini merupakan bentuk cipher blok yang paling sederhana, dengan panjang blok hanya satu unit. Kepraktisan *stream* cipher timbul dari kemampuan transformasi enkripsi yang bisa bervariasi untuk tiap karakter *palintext* yang di enkripsi. Pada kondisi di mana

kesalahan pengiriman data sering muncul, stream cipher lebih unggul karena kesalahan tersebut tidak menyebar ke bagian data lainnya. Selain itu, *stream* cipher cocok untuk kasus di mana data perlu ditangani secara bertahap per simbol (misalnya, pada perangkat dengan memori minim atau kapasitas penyimpanan terbatas) (Alfred, 2014).

Definisi Misalkan K adalah ruang kunci untuk suatu himpunan transformasi enkripsi. Urutan simbol $e_1 e_2 e_3 \ldots e_i \in K$, disebut sebagai aliran kunci (*keystream*).

Contoh:

Misalkan $K = \{0, 1\}$, yang merupakan ruang kunci untuk enkripsi biner (seperti dalam *stream* cipher).

Keystream $e_1 = 1,\ e_2 = 0,\ e_3 = 1, e_4 = 1, e_5 = 0, \ldots$ (urutan ini bisa dihasilkan secara acak atau dari algoritma).

Stream cipher melakukan operasi enkripsi dasar berdasarkan aliran kunci yang sedang dipakai. Aliran kunci itu dapat dibuat secara random, atau lewat algoritma yang memproduksi aliran kunci dari urutan kunci pendek awal (yang disebut *seed*), atau dari *seed* ditambah simbol terenskripsi sebelumnya. Algoritma seperti demikian dinamakan pembangkit aliran kunci (*keystream generator*) (Alfred, 2014).

2. Kunci Asimetris

   Kunci asimetris merupakan kunci yang dimana dalam proses enkripsi dan dekripsi berbeda, misal pengirim memilih sepasang kunci, kemudian pengirim mengirimkan kunci enkripsi (kunci publik) kepada penerima melalui saluran komunikasi apapun, tetapi menyimpan kunci dekripsi secara

aman dan rahasia. Selanjutnya, penerima dapat mengirimkan sebuah pesan M kepada pengirim dengan menerapkan transformasi enkripsi yang ditentukan oleh kunci publik pengirim sehingga menghasilkan *ciphertext*. Pengirim kemudian mendekripsikan *ciphertext* tersebut dengan menerapkan transformasi invers yang secara unik ditentukan oleh kunci privat dari penerima. Kunci asimetris memiliki kelebihan, hanya kunci privat yang harus dijaga kerahasiaannya, pasangan kunci bisa digunakan dalam jangka waktu yang lama, tetapi memiliki kekurangan dalam hal kecepatan, kunci asimetris jauh lebih lambat dibandingkan kunci simetris, dan juga ukuran kunci lebih besar.

2.2 Teori *Chaos* dalam Kriptografi

2.2.1 Konsep Dasar *Chaos*

Teori *chaos* merupakan salah satu bidang ilmu pengetahuan yang mempelajari karakteristik sistem dinamis yang amat labil. Teori *chaos* ini berhubungan dengan bagaimana variasi kecil pada kondisi permulaan dapat menimbulkan perbedaan yang sangat signifikan dalam evolusi sistem dalam jangka waktu yang panjang (Sanjaya W.S, 2025).

Penelitian tentang *chaos* dimulai pada tahun 1963 ketika Edward Lorenz dari MIT menemukan bahwa variasi kecil pada kondisi permulaan dalam simulasi cuaca bisa menghasilkan perbedaan yang sangat signifikan dalam hasil jangka panjang. Fenomena tersebut kemudian dikenal sebagai "*butterfly effect*", yang menggambarkan betapa sensitifnya sistem dinamis terhadap kondisi awal. Penemuan ini memicu kelahiran ilmu kekacauan (*chaos*) yang menekankan pada sifat-sifat sistem yang labil, sulit untuk diramalkan, dan kerap kali acak. Ilmu ini

berkembang dengan cepat dan diaplikasikan dalam berbagai bidang ilmu, seperti meteorologi, mekanika kuantum, serta kompleksitas sosial. Di samping *attractor Lorenz*, terdapat juga *attractor* lain seperti *Rössler attractor* yang memperkaya pemahaman mengenai dinamika kekacauan (*chaos*) (Sanjaya W.S, 2025).

Sistem dinamis yang menampilkan karakteristik kekacauan dikenal sebagai sistem dinamis *chaotik*. Sistem yang *chaotik* bisa dikategorikan sebagai sistem deterministik. Sistem deterministik adalah sistem di mana hasilnya (*output*) sepenuhnya ditentukan oleh masukan (*input*) dan keadaan awal sistem, tanpa adanya unsur keacakanatau kesalahan pengukuran. Sistem *chaotik* juga sepenuhnya ditentukan oleh *input* dan kondisi awalnya, meskipun sifatnya sangat tidak stabil. Dalam sistem *chaotik*, hasil dari sistem diatur oleh hukum-hukum fisika atau matematika yang mengendalikan sistem tersebut, sehingga *output*nya bisa dipastikan untuk setiap input dan kondisi awal yang diberikan (Sanjaya W.S, 2025). *Chaos* merupakan suatu sistem deterministik yang memiliki karakteristik sangat sensitif terhadap kondisi awal, sehingga perbedaan kecil dalam keadaan awal dapat berkembang menjadi perbedaan besar pada perkembangan jangka panjang sistem. Sifat ini menjadikan sistem *chaotik* sulit diprediksi, berbeda dengan sistem deterministik konvensional. Menurut (Sanjaya W.S, 2025) karakteristik lain dari *chaos* meliputi:

1. Perilaku Aperiodik: Sistem tidak selalu mengulang pola yang sama dari waktu ke waktu.
2. Perilaku Ergodik: Sistem tidak menunjukkan pola yang dapat diprediksi dalam jangka panjang.

3. Perilaku Divergensi: Lintasan awal yang berdekatan, di mana perbedaan kecil antar lintasan dapat menghasilkan perbedaan besar pada evolusi sistem di kemudian hari.

2.2.2 Keunggulan *Chaos* untuk Enkripsi Data Teks

Pada sub-bab ini menjelaskan manfaat teori *chaos* dalam mengenkripsi data teks, yang semakin krusial di zaman digital di mana keamanan informasi menjadi hal utama. Teori *chaos*, yang punya sifat tidak lurus (*non-linear*) dan sangat peka terhadap kondisi permulaan, menawarkan cara kriptografi yang tangguh untuk menjaga data teks seperti pesan, dokumen, atau surel dari berbagai jenis ancaman serangan siber. Tidak seperti metode enkripsi biasa seperti *Advanced Encryption Standard* (AES) yang bergantung pada operasi lurus (linear), *chaos* memanfaatkan pergerakan sistem yang kelihatan acak tapi sebenarnya pasti (deterministik), sehingga menciptakan kunci enkripsi yang sulit ditebak dan sangat aman untuk penggunaan teks, termasuk penyesuaian algoritma berbasis gambar untuk enkripsi teks (Lawnik et al., 2022). Kelebihan ini menjadikan *chaos* sebagai pilihan terbaik untuk mengenkripsi data teks, di mana kecepatan dan ketangguhan melawan serangan adalah faktor paling penting.

Salah satu keunggulan utama *chaos* dalam enkripsi data teks adalah sensitivitasnya terhadap kondisi awal, yang dikenal sebagai efek kupu-kupu (*butterfly effect*). Perubahan kecil pada nilai awal, seperti parameter kunci, dapat menghasilkan *output* yang sangat berbeda, sehingga enkripsi menjadi tahan terhadap serangan *brute-force* atau analisis diferensial (Lawnik et al., 2022). Hal ini sangat berguna untuk data teks, di mana setiap karakter dapat dienkripsi menggunakan urutan *chaos* yang unik, membuat proses dekripsi tanpa kunci tepat

menjadi hampir mustahil (Riaz et al., 2024). Penelitian terbaru oleh (Jin et al., 2024) menunjukkan bahwa sistem *chaos* dengan penundaan waktu bervariasi mampu menghasilkan urutan *pseudo*-acak yang sangat kompleks dan efisien untuk proses enkripsi. Karakteristik ini memperkuat difusi dan konfusi dalam kriptografi, sehingga meningkatkan keamanan data sekaligus mempertahankan efisiensi. Temuan ini menegaskan kembali bahwa algoritma berbasis *chaos* lebih tangguh dibandingkan metode tradisional karena memanfaatkan sifat deterministik yang tampak acak, sensitif terhadap kondisi awal, dan sulit diprediksi.

2.3 Algoritma *Logistic Map*

Algoritma *Logistic Map* merupakan salah satu bentuk peta dinamis satu dimensi yang banyak digunakan dalam kajian sistem *chaos*. *Logistic Map* didefinisikan dengan persamaan rekursif, sebagai berikut:

$$x_{n+1} = r.x_n(1 - x_n)$$

dimana:

x_n : Nilai populasi (atau keadaan sistem) pada iterasi ke-n, dengan $x_n \in [0, 1]$

x_0 : Kondisi awal.

x_{n+1} : Nilai populasi pada iterasi berikutnya.

r : Parameter pengendali (kontrol parameter)

n : Indeks waktu diskret (iterasi ke-n), dengan $n = 0, 1, 2, \ldots$

$(1 - x_n)$: Faktor pembatas yang merepresentasikan efek kepadatan populasi.

Pada awalnya, model ini diperkenalkan oleh May (1976) untuk menggambarkan pertumbuhan populasi serangga secara sederhana, dengan asumsi

setiap individu bertelur, menetas, berkembang biak, dan mati dalam siklus tahunan. Pertumbuhan populasi akan meningkat secara eksponensial ketika nilai parameter r relatif kecil, namun dengan bertambahnya populasi akan terjadi efek kepadatan (*overcrowding*) yang mengurangi jumlah individu pada generasi berikutnya (Ott, 2002).

Dinamika *Logistic Map* memperlihatkan perilaku yang kompleks ketika nilai parameter r divariasikan. Fungsi pemetaan $x_{n+1} = r.x_n(1 - x_n)$ menunjukkan bahwa nilai keadaan pada iterasi berikutnya (x_{n+1}) bergantung pada nilai keadaan saat ini (x_n) dengan faktor pengendali berupa parameter r. Nilai x_n dibatasi dalam interval $[0, 1]$, sedangkan r biasanya diambil pada rentang $0 < r \leq 4$.

1. Pada $0 < r \leq 1$, semua orbit cenderung menuju titik tetap $x = 0$.
2. Pada $1 < r \leq 3$, sistem memiliki titik tetap stabil di $x = \frac{r-1}{r}$.
3. Pada $3 < r \leq 3{,}7$; sistem mulai mengalami bifurkasi (transisi dari keadaan stabil ke keadaan *chaotic*) berulang, di mana orbit berosilasi (variabel berubah secara berulang dalam pola) secara periodik.
4. Jika, $r > 3{,}7$;sistem masuk ke dalam fase *chaos*, orbit menjadi sangat sensitif terhadap kondisi awal meskipun persamaanya deterministik.

Bentuk iteratif ini (menggunakan indeks n) sangat penting, karena menunjukkan bahwa *Logistic Map* bukan hanya sekedar fungsi statis, melainkan model dinamis yang menggambarkan bagaimana nilai berubah dari satu iterasi ke iterasi berikutnya. Sifat inilah yang dimanfaatkan dalam kriptografi, kepekaan terhadap kondisi awal dan parameter r menghasilkan urutan *pseudo*-acak yang sulit diprediksi, tetapi tetap dapat direproduksi bila kunci awal diketahui (Ott, 2002).

2.3.1 *Logistic Map* sebagai Penghasil *Keystream*

Iterasi dari *Logistic Map* menghasilkan deret nilai real berupa:

$$x_1, x_2, x_3, \ldots$$

yang berada pada interval $[0, 1]$. Deret nilai tersebut mengahasilkan nilai x_n, setiap nilai x_n hasil iterasi dikonversi menjadi bit, dengan ($1, x_n > 0{,}5$ dan $0, x_n \leq 0,5$). Proses ini menjadikan *Logistic Map* berfungsi sebagai *pseudo-random number generator* yang mampu menghasilkan rangkaian kunci yang tampak acak, namun tetap deterministik dan dapat direproduksi ketika kondisi awalnya diketahui (Ott, 2002). Dalam konteks kriptografi *chaos*, keystream ini digunakan sebagai kunci utama pada tahap substitusi, khususnya pada operasi XOR. Pada penelitian (Arif et al., 2022) menunjukkan bahwa keluaran *Logistic Map* tidak hanya berperan sebagai sumber *keystream* untuk tahap substitusi, tetapi juga digunakan sebagai pengendali dalam tahap permutasi. Dengan demikian, nilai-nilai *chaos* yang dihasilkan *Logistic Map* memiliki dua fungsi sekaligus, yaitu sebagai kunci XOR dan sebagai parameter yang menentukan pola permutasi baris maupun kolom. Pendekatan ini memperkuat aspek *confusion* dan *diffusion* secara simultan, yang menjadi prinsip dasar keamanan kriptografi modern.

2.3.2 Operasi XOR Berbasis *Logistic Map*

Operasi *Exclusive OR* (XOR) adalah operasi logika biner yang mengambil dua nilai bit dan mengahasilkan *output* 1 jika kedua *input* berbeda, serta 0 jika keduanya sama (Reymond et al., 2024), seperti yang ditunjukkan dalam tabel kebenaran:

Tabel 2.1 Operasi XOR

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Dalam kriptografi, XOR sering digunakan sebagai bagian dari transformasi data karena kesederhanaannya yang efektif dalam membuat hasil enkripsi tampak acak dan tidak mudah ditebak. Ketika *plaintext* (P) di-XOR dengan kunci (K), proses dekripsi dapat dilakukan dengan sangat mudah, yaitu cukup meng-XOR kembali *ciphertext* tersebut dengan kunci yang sama untuk memperoleh kembali *plaintext* (P), karena:

$$(P \oplus K) \oplus K = P$$

Selain itu, XOR biasanya dikombinasikan dengan operasi lain seperti permutasi, substitusi, atau peta *chaos* untuk meningkatkan keamanan dan kompleksitas algoritma enkripsi. Contoh penerapan XOR dalam penelitian terbaru meliputi penggunaan dalam *One-Time Pad* (OTP) yang mengandalkan XOR sebagai operasi utama untuk enkripsi dan dekripsi, dengan syarat kunci harus unik dan sepanjang pesan agar sistem aman dan sulit dipecahkan (Reymond et al., 2024).

Dalam enkripsi gambar, operasi XOR bisa digabungkan dengan S-box nonlinier dan permutasi acak, di mana setiap lapisan gambar di-XOR dengan kunci unik, kemudian dipermutasi dan diproses melalui S-box untuk menghasilkan *ciphertext* (Artuğer, 2025). Selain itu, dalam pengembangan aplikasi chat berbasis android, XOR digunakan untuk mengamankan pesan, di

mana pengirim dan penerima harus memiliki kunci yang sama agar pesan yang di-XOR dapat didekripsi kembali (Saputro, 2023).

Operasi XOR merupakan mekanisme substitusi yang paling umum digunakan dalam kriptografi modern karena sifatnya yang *self-inverse*, yaitu operasi yang sama dapat digunakan untuk enkripsi maupun dekripsi. dalam konteks kriptografi *chaos*, *keystream* yang dihasilkan dari iterasi *Logistic Map* digunakan sebagai kunci XOR untuk setiap byte *plaintext*. Proses enkripsi dilakukan dengan rumus:

$$C = P \oplus K$$

sedangkan dekripsi dilakukan dengan operasi yang sama, yaitu:

$$P = C \oplus K$$

sehingga memungkinkan proses pembalikan sempurna ketika *keystream* yang sama digunakan. Penelitian (Arif et al., 2022) menunjukkan bahwa nilai-nilai *chaos* dari *Logistic Map* memiliki distribusi yang mendekati acak sehingga sangat efektif digunakan sebagai kunci substitusi dalam tahap *permutation-substitation* pada kriptografi berbasis citra. Selain itu, penelitian (Ignatius & Setiadi, 2023) menjelaskan bahwa operasi XOR yang dikombinasikan dengan *keystream chaotic* dari peta *Logistic* dua dimensi menghasilkan efek penyebaran (*confusion-diffusion*) yang kuat pada *ciphertext*, di mana perubahan kecil pada nilai awal atau parameter menyebabkan perubahan kecil pada nilai awal atau parameter menyebabkan perubahan besar pada hasil enkripsi. Mereka membuktikan bahwa *keystream* yang dibangkitkan oleh peta *chaos* memberikan sensitivitas kunci yang sangat tinggi dan meningkatkan keamanan sistem enkripsi berbasis permutasi-substitusi (Ignatius & Setiadi, 2023).

2.3.3 Permutasi Baris dan Kolom Berbasis *Logistic Map*

Selain digunakan sebagai sumber *keystream*, keluaran *Logistic Map* juga dimanfaatkan sebagai pengendali dalam proses permutasi (*diffusion phase*) pada sistem kriptografi berbasis *chaos*. Pada tahap ini, data direpresentasikan dalam bentuk matriks berukuran $N \times N$ agar proses transformasi dan permutasi data dapat dilakukan secara sistematis. Representasi matriks memungkinkan penerapan algoritma *Arnold's Cat Map* yang beroperasi dengan memanipulasi posisi elemen melalui pergeseran baris dan kolom, sehingga mendukung pencapaian tingkat *diffusion* yang tinggi dalam sistem enkripsi. Bentuk matriks persegi dipilih karena algoritma pengacakan termasuk operasi pergeseran melingkar (*circular shift*) bekerja paling efektif dan konsisten ketika jumlah baris dan kolomnya sama. Dengan menggunakan matriks $N \times N$, pola pergeseran dapat diterapkan secara simetris pada setiap baris maupun kolom, sehingga proses pengacakan lebih teratur, stabil, dan mudah dikembalikan (*reversible*) pada tahap dekripsi. Selain itu, banyak algoritma pengacakan seperti *Arnold's Cat Map* secara matematis dirancang untuk bekerja pada matriks persegi, sehingga data perlu disesuaikan ke dalam bentuk tersebut terlebih dahulu. Pada tahap merepresentasikan dalam bentuk matriks $N \times N$ atau matriks persegi, jika jumlah data tidak mencukupi untuk membentuk matriks persegi, maka dilakukan penentuan nilai N dengan syarat $N^2 \geq B$, dimana B adalah banyak data. Sisa ruang diisi menggunakan padding (proses menambahkan data tambahan, biasanya 0 atau pola tertentu, guna menyesuaikan ukuran data dengan persyaratan algoritma. Langkah ini memastikan bahwa seluruh elemen dalam matriks terisi penuh sehingga dapat diproses secara optimal tanpa ada posisi data yang tidak

terdefinisi). Dengan pendekatan ini, proses permutasi tetap dapat dilakukan meskipun banyaknya data tidak genap atau tidak sesuai bentuk persegi, berikut contoh pengimplementasiannya:

Jumlah data sebanyak 10

Cari N:

$$N^2 \geq 10 \Rightarrow N = 4$$

Maka, dibentuk matriks $4 \times 4 = 16$ posisi, artinya butuh padding 6 posisi.

Susunan matriks 4×4, sebagai berikut:

$$\begin{bmatrix} D_1 & D_2 & D_3 & D_4 \\ D_5 & D_6 & D_7 & D_8 \\ D_9 & D_{10} & p_1 & p_2 \\ p_3 & p_4 & p_5 & p_6 \end{bmatrix}$$

dengan p_i menyatakan padding untuk $i = 1, 2, 3, \ldots, 6$; setelah itu baru dilakukan permutasi baris dan kolom. Pengacakan ini bertujuan untuk memperkuat mekanisme *diffusion*, yaitu penyebaran perubahan satu elemen *plaintext* ke banyak posisi pada *ciphertext*. Banyaknya pergeseran (*shift*) diperoleh dari keluaran iterasi *Logistic Map* x_n yang dipetakan menjadi bilangan bulat melalui rumus:

$$k \equiv \lfloor x_n \times 10 \rfloor \; mod \; N$$

dengan:

x_n : Nilai *Logistic Map* pada iterasi ke-n

$x_n \times 10$: Memperbesar skala agar variasi pergeseran lebih dinamis

$\lfloor . \rfloor$: Fungsi *floor* untuk mengambil bilangan bulat terbesar yang lebih kecil atau sama dengan argumen

$mod \; N$: Memastikan nilai *shift* berada dalam rentang 0 hingga $N - 1$

k : Jumlah langkah pergeseran secara melingkar

Nilai *shift* tersebut kemudian digunakan untuk menggeser baris dan kolom. Pergeseran baris didefinisikan sebagai:

$$k_r \equiv \lfloor x_n \times 10 \rfloor \, mod \, N$$

dan pergeseran kolom sebagai berikut:

$$k_c \equiv \lfloor x_{n+1} \times 10 \rfloor \, mod \, N$$

dengan:

k_r : *Shift* untuk baris (menggunakan iterasi ke-n)

k_c : *Shift* untuk kolom (menggunakan iterasi ke-$(n+1)$ agar pola pergeserab berbeda)

Dengan *shift* tersebut, operasi permutasi baris dilakukan menggunakan rumus berikut:

$$Row'[i] \equiv Row[(i - k_{rb}) mod \, N]$$

dengan:

$Row[i]$: Baris ke-i sebelum permutasi

$Row'[i]$: Baris ke-i setelah permutasi

i : Indeks baris (0 sampai $N-1$)

k_{rb} : Jumlah geseran ke bawah

pada permutasi kolom dilakukan menggunakan rumus sebagai berikut:

$$Col'[j] \equiv Col[(j - k_{ck}) mod \, N]$$

dengan:

$Col[j]$: Kolom ke-i sebelum permutasi

$Col'[j]$: Kolom ke-i setelah permutasi

j : Indeks kolom (0 sampai $N-1$)

k_{ck} : Jumlah geseran ke kanan

Pada penelitian (Arif et al., 2022) menunjukkan bahwa penggunaan *Logistic Map* sebagai pengendali permutasi baris dan kolom mampu meningkatkan keacakan struktural pada *ciphertext*, karena nilai *shift* berubah setiap iterasi sehingga pola distribusi data tidak dapat diprediksi. Temuan ini diperkuat oleh penelitian (Ignatius & Setiadi, 2023) yang menjelaskan bahwa peta *chaos* menghasilkan efek *diffusion-confusion* yang kuat pada sistem enkripsi berbasis permutasi-substitusi. Kedua penelitian tersebut menegaskan bahwa permutasi yang dikendalikan oleh nilai *chaos* merupakan komponen penting dalam membangun algoritma enkripsi yang aman, karena dapat menyebarkan perubahan satu bit pada *plaintext* seluruh blok *ciphertext* secara signifikan.

2.4 Algoritma *Arnold's Cat Map*

Inti dari studi mekanika klasik adalah memprediksi kondisi masa depan suatu sistem dinamis berdasarkan kondisi awalnya. Salah satu kelas sistem dinamis yang relevan adalah diferomorfisme Anasov, yaitu fungsi dengan sifat *chaos* yang khas. Untuk memahami konsep yang abstrak dan kompleks seperti ini, matematika biasanya dimulai dari contoh sederhana yang lebih konkret. Salah satu contoh diferomorfisme Anosov yang terkenal adalah peta yang diperkenalkan oleh Vladmir Arnold pada tahun 1968, yang kemudian dikenal dengan nama *Arnold's Cat Map* (Dachlan, 2014). Nama ini berasal dari gambar kepala kucing yang digunakan Arnold untuk menjelaskan transformasi tersebut dalam bukunya bersama A.Avez (Arnol'd, 1968).

Secara matematis, *Arnold's Cat Map* didefinisikan dengan transformasi linier:

$$\begin{cases} x_{n+1} \equiv (x_n + y_n) \bmod N \\ y_{n+1} \equiv (x_n + 2y_n) \bmod N \end{cases}$$

dengan setiap posisi elemen pada matriks direpresentasikan sebagai entri matriks. Entri dinyatakan dengan pasangan indeks (x, y) yang menunjukkan baris ke-x dan kolom ke-y. Penomoran indeks dalam matriks menggunakan *zero-based indexing*, sehingga entri pertama berada pada posisi $(0,0)$. Pendekatan ini umum digunakan dalam implementasi algoritma komputer, termasuk *Arnold's Cat Map*, karena memudahkan proses perhitungan transformasi pada setiap entri. Dengan demikian, setiap entri dalam matriks berukuran $N \times N$ dinotasikan mulai dari $(0, 0)$ hingga $(N - 1, N - 1)$ untuk keperluan transformasi dan pengacakan data. Operasi modulo memastikan entri hasil tetap berada dalam batas matriks (Min et al., 2013).

Dalam konteks ini, operasi modulo diterapkan pada setiap entri (x, y) secara individual, bukan melakukan modulo pada keseluruhan matriks. Artinya, setelah entri baru dihitung, masing-masing nilai x_{n+1} dan y_{n+1} dirapatkan kembali ke dalam rentang valid indeks $[0, N - 1]$ melalui operasi $(mod\ N)$. Tidak ada konsep "matriks modulo" dalam teori ini, yang ada adalah entri modulo untuk menjaga agar letak hasil transformasi tetap berada dalam batas ukuran matriks. Pendekatan ini merupakan standar dalam implementasi *Arnold's Cat Map,* karena transformasinya memang bekerja pada koordinat atau entri, bukan pada objek matriks sebagai satu kesatuan. Dengan demikian, setiap entri pada matriks berukuran $N \times N$ dinotasikan mulai dari $(0,0)$ hingga $(N - 1, N - 1)$, dan operasi modulo memastikan bahwa proses transformasi dan pengacakan tetap berada dalam ruang indeks matriks yang valid.

Adapun beberapa karakteristik algoritma *Arnold's Cat Map* menurut (Min et al., 2013)diantaranya adalah:

1. Palestarian luas (Area *Preservation*)

   Pada *Arnold's Cat Map,* transformasinya biasanya dinyatakan dengan:

$$\begin{cases} x_{n+1} \equiv (x_n + y_n) \ mod \ N \\ y_{n+1} \equiv (x_n + 2y_n) \ mod \ N \end{cases}$$

   karena determinan matriks transformasi *Arnold's Cat Map* adalah 1, maka transformasi tidak mengubah luas bidang, artinya meskipun posisi titik-titik berubah, total area (misalnya pada gambar digital) tetap konstan. Hal ini disebut area-*preserving transformation*.

3. Periodisitas

   Walaupun transformasi ini tampak "mengacak", sebenarnya ia periodik dalam ruang diskret (karena bekerja dalam modulo N). Artinya setelah sejumlah iterasi tertentu, semua titik akan kembali ke posisi semula. Periode ini tergantung pada ukuran bidang N dan sifat aritmetika modularnya.

4. Sifat *chaotic*

   Arnold's Cat Map adalah deterministik tetapi tampak acak dan merupakan sifat umum sistem *chaos*. Ciri-cirinya sebagai berikut:

   a. Sangat sensitif terhadap kondisi awal (dua titik yang sangat dekat bisa menyebar jauh setelah beberapa iterasi)
   b. Dapat dibalik (*reversible*), karena determinannya adalah 1

5. Hubungan dengan deret *Fibonacci*

   Matriks transformasi pada *Arnold's Cat Map* memiliki sifat khusus:

   jika menghitung pangkatnya:

$$A^n = \begin{bmatrix} F_{2n-1} & F_{2n} \\ F_{2n} & F_{2n+1} \end{bmatrix}$$

di mana F_n adalah bilangan *Fibbonacci* ke-n, artinya setiap kali menaikkan pangkat matriks transformasi pada *Arnold's Cat Map* , entri-entrinya membentuk pola *Fibonacci*. Ini artinya titik-titik pada peta *Arnold* secara tidak langsung mengikuti pola pertumbuhan *Fibonacci* ruang dua dimensi.

Dalam penelitian ini, *Arnold's Cat Map* berperan sebagai metode pengacakan posisi data teks. Awalnya peta ini digunakan untuk mengacak piksel pada citra digital, namun dapat diadaptasi untuk mengacak posisi karakter pada pesan teks. Dengan cara ini, *Arnold's Cat Map* memperkuat proses pengacakan dalam enkripsi sehingga *ciphertext* menjadi lebih sulit ditebak tanpa kunci yang benar.

2.4.1 Kongruensi Matriks

Pada himpunan bilangan bulat, gagasan aritmetika modulo memiliki hubungan yang sangat dekat dengan konsep kongruensi serta keterbagian. Kajian terkait keterbagian dan karakteristik-karakteristiknya bisa dieksplorasi secara lebih mendalam melalui pendekatan konsep kongruensi. Dengan sebab itu, kongruensi dapat dianggap sebagai metode alternatif untuk menelaah keterbagian di dalam kumpulan bilangan bulat, yang menjadi fondasi utama dalam pembentukan sistem aritmetika modulo (W. H. Irawan, 2013).

Definisi Misalkan terdapat sebuah bilangan bulat $m \neq 0$. Apabila m membagi selisih $a - b$, maka disebut bahwa a kongruen dengan b modulo m, dan dinyatakan dengan $a \equiv b(mod\ m)$ (W. H. Irawan, 2013).

Dari definisi di atas dapat dijelaskan, ada bilangan bulat t sehingga $a - b = mt$ atau bisa dituliskan ulang menjadi $a = b + mt$, sebaliknya, apabila $a - b$ tidak habis dibagi oleh m, maka disebut bahwa a tidak kongruen dengan b

modulo m, dan dinyatakan dengan $a \not\equiv b(mod\ m)$. Dengan demikian, dua bilangan disebut kongruen modulo m jika keduanya mempunyai sisa hasil bagi yang identik saat dibagi dengan m (W. H. Irawan, 2013).

Contoh:

1. $27 \equiv 2(mod\ 5)$, karena $27 - 2 = 25$ dan 25 habis dibagi 5.

   Bentuk seperti di atas bisa dituliskan juga seperti ini, $27 = 5 \times 5 + 2$.

2. $35 \not\equiv 6(mod\ 7)$, karena $35 - 6 = 29$ dan 29 tidak habis dibagi 7.

Berdasarkan konsep kongruensi, aritmetika modulo dapat dipandang sebagai operasi bilangan bulat yang dilakukan dengan memperhatikan sisa pembagian terhadap suatu bilangan tetap m yang disebut modulus. Dua bilangan a dan b dikatakan ekuivalen modulo m jika $a \equiv b\ (mod\ m)$, artinya keduanya memiliki sisa pembagian yang sama terhadap m. Kumpulan bilangan yang saling kongruen dengan a disebut kelas residu a modulo m. Sebagai contoh, untuk $m = 5$, terdapat lima kelas residu $[0], [1], [2], [3], [4]$ yang masing-masing mewakili bilangan dengan sisa pembagian 0 hingga 4. Dalam sistem ini, operasi penjumlahan, pengurangan, dan perkalian dilakukan terhadap sisa pembagian, mengikuti aturan $(a \pm b) mod\ m = [(a\ mod\ m) \pm (b\ mod\ m)] mod\ m$ dan $(a \times b) mod\ m = [(a\ mod\ m) \times (a\ mod\ b)] mod\ m$. Dengan demikian, hasil setiap operasi selalu berada dalam himpunan residu $\{0, 1, 2, \dots, m - 1\}$, membentuk struktur aljabar yang tertutup dalam aritmetika modulo (W. H. Irawan, 2013).

Konsep kongruensi tidak hanya berlaku pada bilangan bulat, tetapi juga dapat diperluas ke matriks. Adapun definisi yang menjelaskan tentang kongruensi matriks,

Definisi Misalkan A dan B adalah ,matriks $n \times k$ dengan unsur-unsurnya bilangan bulat, unsur ke (i, j) berturut-turut adalah a_{ij} dan b_{ij}. A dikatakan kongruensi dengan B modulo m, jika $a_{ij} \equiv b_{ij} \ (mod\ m)$ untuk setiap pasang (i, j) dengan $1 \leq i \leq n$ dan $i \leq j \leq k$ dan dinotasikan dengan $A \equiv B\ (mod\ m)$ (W. H. Irawan, 2013).

Contoh:

$$\begin{pmatrix} 15 & 3 \\ 8 & 12 \end{pmatrix} \equiv \begin{pmatrix} 4 & 3 \\ -3 & 1 \end{pmatrix} (mod\ 11)$$

Pernyataan $A \equiv B\ (mod\ m)$ tidak berarti seluruh matriks A dan B dimodulokan secara keseluruhan, melainkan setiap entri atau elemen dari kedua matriks tersebut dibandingkan berdasarkan kongruensi modulo m. Dengan kata lain, operasi modulo dilakukan per elemen, bukan terhadap keseluruhan struktur matriks.

2.5 Amanah dalam Islam

Kata *amanah* berasal dari akar huruf *alif-hamzah, mim*, dan *nun* yang mengandung dua makna pokok yang saling berkaitan. Pertama, *al-amanah* yang menjadi lawan dari *al-khiyanah*, bermakna *sukun al-qalb* atau ketenangan hati. Kedua, *al-tasdiq*, yang berarti mempercayakan sesuatu. Kedua makna ini memiliki kedekatan, karena amanah selalu berkaitan dengan kepercayaan yang menghadirkan rasa tenang. *Al-Khalil* menjelaskan bahwa *al-amanah* berasal dari *al-amn*, yang berarti memberikan rasa aman, dan merupakan kebalikan dari pengkhianatan (*al-khiyanah*) (Dalimunthe, 2018).

Dasar hukum amanah dalam Islam ditegaskan dalam Al-Qur'an surah Al-Mu'minun ayat (8) (Kemenag, 2022):

وَالَّذِيْنَ هُمْ لِاَمٰنٰتِهِمْ وَعَهْدِهِمْ رَاعُوْنَۙ ﴿٨﴾

Artinya: *"Dan (sungguh beruntung) orang-orang yang memelihara amanat-amanat dan janjinya."*

Ayat ini menekankan bahwa salah satu ciri utama orang beriman adalah kesanggupan menjaga dan menunaikan amanah. Dalam *Tafsir Al-Azhar*, Hamka menjelaskan bahwa iman seseorang tidak dapat dipisahkan dari tanggung jawab dalam memelihara amanah yang telah Allah pikulkan kepadanya, baik dalam urusan keagamaan maupun sosial. Beliau menekankan bahwa amanah adalah fondasi keteraturan masyarakat yang adil dan makmur (Hamka, 2003)

Menurut (Hamka, 2003), amanah terbagi ke dalam dua bentuk utama. Pertama, amanat raya, yaitu tanggung jawab seluruh umat manusia sebagai *Khalifatullah fil-ardh*. Amanat ini mencakup kewajiban menjalankan hukum Allah, menegakkan keadilan, dan menjaga keseimbangan kehidupan di muka bumi. Kedua, amanat pribadi, yakni amanah yang melekat pada setiap orang sesuai dengan kemampuan, profesi, dan kedudukannya. Seorang pemimpin negara, petani, sopir, pedagang kecil, atau ibu rumah tangga masing-masing memiliki amanah tersendiri. Hamka menegaskan bahwa semua peran ini bernilai sama mulianya apabila dijalankan dengan penuh kesetiaan dan ketakwaan.

Demikian pula, Quraish Shihab dalam *Tafsir Al-Misbah* (Shihab, 2016) mengartikan amanah sebagai sesuatu yang dipercayakan kepada seseorang untuk dipelihara dan dikembalikan secara sempurna kepada pemiliknya. Baginya, amanah meliputi empat aspek pokok, yaitu pertama relasi manusia dengan Allah (ibadah dan zakat), kedua dengan orang lain (titipan dan rahasia), ketiga dengan alam

sekitar (perawatan lingkungan agar tetap lestari), dan yang keempat adalah dengan dirinya sendiri (memelihara kesehatan dan kehormatan). Beliau menekankan bahwa amanah menjadi dasar keimanan dan elemen kunci dalam hubungan sosial yang menghasilkan kepercayaan serta kedamaian dalam kehidupan bermasyarakat (Shihab, 2016).

Dalam *Tafsir Ibnu Katsir* (Ghoffar & Abdurrahim Mu'thi, 2003), amanah dijelaskan sebagai segala bentuk kepercayaan yang harus dijaga dan ditunaikan. Ketika seseorang diberi amanah, ia tidak boleh mengkhianatinya, dan jika berjanji, ia wajib menepatinya. Orang yang menunaikan amanah dan menepati janji termasuk dalam golongan orang mukmin sejati, sedangkan yang berkhianat termasuk dalam ciri orang munafik. Pandangan ini menunjukkan bahwa amanah bukan sekadar moral individu, tetapi juga ukuran keimanan dan kematangan spiritual (Ghoffar & Abdurrahim Mu'thi, 2003).

Sementara itu, Ath-Thabari dalam *Jami' al-Bayan fi Ta'wil al-Qur'an* menjelaskan bahwa amanah meliputi seluruh tanggung jawab manusia, baik *duniawi* seperti harta dan pekerjaan, maupun *ukhrawi* seperti ibadah dan ketaatan kepada Allah SWT. Amanah dalam pandangan Ath-Thabari merupakan perwujudan dari kepercayaan Allah kepada manusia sebagai *khalifah* di bumi. Dengan menjaga manah, manusia menjalankan peran spiritualnya dalam memakmurkan dunia sesuai tuntutan ilahi (Ath-Thabari, 2007).

Dari seluruh pandangan para mufasir tersebut, dapat disimpulkan bahwa amanah mencakup aspek hukum, sosial, dan moral. Dalam Islam, amanah adalah kunci terciptanya masyarakat yang adil, tertib, dan makmur. Menjalankan amanah berarti melaksanakan tanggung jawab yang diberikan oleh Allah dan sesama

manusia dengan jujur, adil, serta penuh kesadaran spiritual. Cara mengemban amanah, sebagaimana ditegaskan oleh Hamka dan Quraish Shihab, adalah dengan melaksanakan setiap tugas secara adil, setia, dan proporsional. Amanah bukan sekadar ukuran kedudukan, tetapi wujud dari integrasi dan keikhlasan hati dalam bekerja dan berbuat.

Dalam konteks kehidupan modern, nilai amanah memiliki relevansi yang luas, termasuk dalam bidang keamanan informasi dan teknologi. Menjaga kerahasiaan data, melindungi privasi pengguna, dan menghindari penyalahgunaan teknologi merupakan wujud nyata dari pelaksanaan amanah. Tanggung jawab etis ini sejalan dengan ajaran Islam yang menuntut umatnya untuk bertindak amanah dalam setiap urusan, baik yang bersifat spiritual maupun profesional. Dengan demikian, konsep amanah menjadi jembatan anatara nilai keimanan dan penerapan etika dalam dunia modern, menjadikan teknologi sebagai sarana menegakkan nilai-nilai kejujuran, keadilan, dan kepercayaan.

2.6 Kajian Topik dengan Teori Pendukung

Berdasarkan teori-teori yang telah dijelaskan sebelumnya, penelitian ini menggunakan algoritma *Logistic Map* dan *Arnold's Cat Map* sebagai dasar utama dalam rancangan sistem pengamanan pesan teks. *Logistic Map* berfungsi untuk menghasilkan deret *pseudo*-acak yang digunakan sebagai *keystream*, sehingga memperkuat proses substitusi dalam enkripsi, sedangkan *Arnold's Cat Map*, yang diadaptasi dari bidang pengolahan citra digital, digunakan untuk mengacak posisi karakter dalam pesan teks, sehingga meningkatkan difusi dan menambah kompleksitas *ciphertext* yang dihasilkan. Kombinasi kedua algoritma ini memungkinkan sistem enkripsi memiliki dua lapisan perlindungan, yaitu

pengacakan berbasis bilangan *chaos*, yang secara bersama-sama meningkatkan ketahanan terhadap serangan kriptografi modern seperti *brute force*, serangan diferensial, dan analisis statistik. Dengan demikian, kerangka teori ini akan menjadi landasan dalam pembahasan implementasi algoritma pada bab berikutnya, sekaligus mengisi kekosongan penelitian yang selama ini lebih banyak berfokus pada data citra dan jarang diterapkan secara khusus pada pesan teks digital.

BAB III

METODE PENELITIAN

3.1 Jenis Penelitian

Penelitian ini menggunakan pendekatan kualitatif dengan tujuan untuk memahami dan mengimplementasikan algoritma *Logistic Map* dan *Arnold's Cat Map* dalam konteks pengamanan pesan. Metode kualitatif memungkinkan penelitian untuk menggali secara detail bagaimana kedua algoritma tersebut dapat diaplikasikan dalam sistem enkripsi simetris dan dekripsi, serta memahami karakteristik, kelebihan, dan keterbatasan masing-masing algoritma. Proses penelitian berupa perancangan dan implementasi algoritma dengan menggunakan metode kualitatif, sehingga penelitian ini diharapkan dapat memberi pemahaman yang menyuluruh mengenai penerapan kedua algoritma sebagai alternatif teknik enkripsi simetris dan dekripsi berbasis teori *chaos*, sekaligus memberikan rekomendasi pengembangan lebih lanjut dalam bidang kriptografi.

3.2 Tahapan Penelitian

Berikut adalah tahapan-tahapan yang akan dilakukan pada penelitian ini:

3.2.1 Proses Pembentukan Kunci (*Keystream*)

1. Tentukan pesan teks yang akan di enkripsi
2. Konversi pesan ke ASCII (byte)
3. Tentukan parameter *Logistic Map*
    a. Pilih parameter *chaos* r
    b. Pilih nilai awal x_0
    c. Tentukan banyaknya iterasi pemanasan (*warm-up*)

d. Hitung total bit *keystream* yang dibutuhkan, dengan cara

$$banyak\ karakter\ pesan\ \times 8\ bit$$

4. Iterasi *Logistic Map* untuk menghasilkan *keystream*

a. Melanjutkan iterasi *Logistic Map* sebanyak total bit *keystream* yang dibutuhkan

b. Setiap nilai x_n hasil iterasi dikonversi menjadi bit :

$$\text{bit } keystream = \begin{cases} 1, & jika\ x_n > 0{,}5 \\ 0, & jika\ x_n \leq 0{,}5 \end{cases}$$

5. Gabungkan bit-bit menjadi byte *keystream*

3.2.2 Proses Enkripsi Pesan Teks

1. Tentukan suatu pesan (M)
2. Mengubah M (*plaintext*) menjadi bilangan biner dengan 8 bit menggunakan kode ASCII
3. Melakukan permutasi baris dan kolom pada *keystream* yang telah dibangkitkan dari *Logistic Map*.

   Keystream yang diperoleh dari iterasi *Logistic Map* disusun ke dalam bentuk matriks persegi, kemudian dilakukan proses permutasi baris dan kolom menggunakan pergeseran melingkar (*circular shift*). Nilai pergeseran ditentukan dari keluaran *Logistic Map* melalui operasi $\lfloor x_n \times 10 \rfloor mod\, N$. Permutasi ini bertujuan untuk mengacak posisi elemen-elemen *keystream* sehingga distribusi kuncinya menjadi lebih acak dan memperkuat tingkat *confusion* sebelum digunakan dalam operasi XOR
4. Melakukan XOR antara setiap byte *plaintext* dengan byte *keystream* yang sesuai:

$$Ciphertext\ (i) = plaintext\ (M)\ \oplus keystream\ (K)$$

5. Melakukan XOR antara setiap byte *plaintext* dengan byte *keystream* yang sesuai:

$$Ciphertext\ (i) = plaintext\ (M)\ \oplus keystream\ (K)$$

6. Susun *ciphertext* hasil XOR ke dalam matriks persegi
   a. Tentukan ukuran matriks $N \times N$ yang cukup menampung semua byte *ciphertext*
   b. Jika jumlah byte *ciphertext* bukan kuadrat sempurna, tambahkan *padding* (misal byte 0) sampai ukuran matriks terpenuhi
7. Definisikan *Arnold's Cat Map* untuk setiap entri (i, j) dalam matriks ukuran $N \times N$, posisi baru (i', j') dihitung dengan :

$$\begin{cases} x_{n+1} \equiv (x_n + y_n)\ mod\ N \\ y_{n+1} \equiv (x_n + 2y_n)\ mod\ N \end{cases}$$

8. Lakukan pengacakan (ulangi proses ini sebanyak K kali atau jumlah iterasi pengacakan)
9. Ambil kembali *ciphertext* dari matriks yang sudah diacak dan pesan siap dikirim.

3.2.3 Proses Dekripsi Pesan Teks

1. Terima *Ciphertext* Teracak

   Ciphertext yang diterima sudah dalam bentuk matriks atau array yang posisi byte-nya sudah diacak menggunakan algoritma *Arnold's Cat Map*

2. Bentuk Matriks *Ciphertext* Teracak

   Susun *ciphertext* teracak menjadi matriks persegi $N \times N$ sesuai ukuran yang digunakan saat enkripsi

3. Terapkan Invers Matriks Transformasi *Arnold's Cat Map*

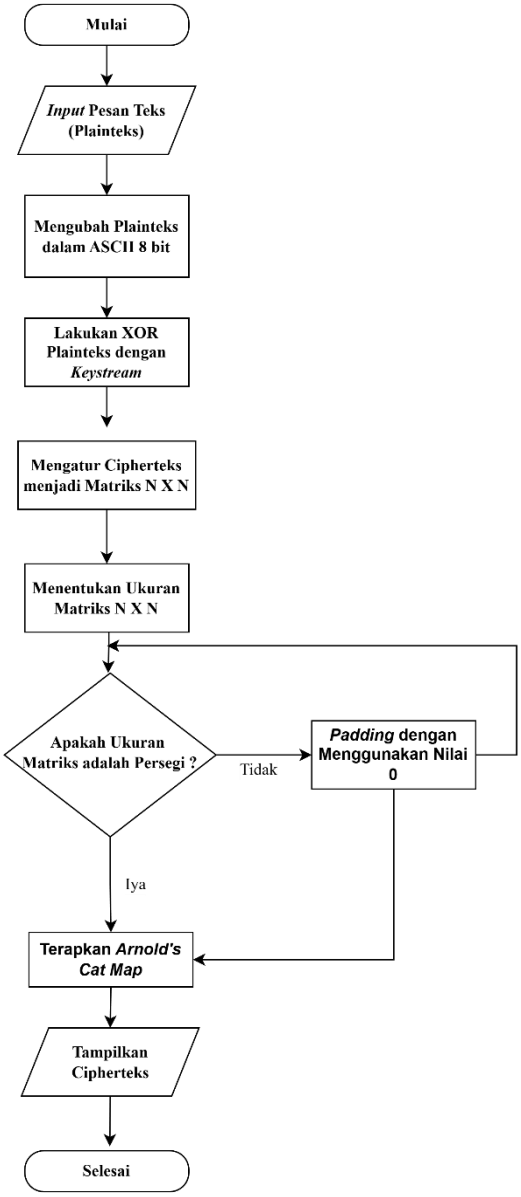
Untuk mengembalikan data atau pesan, digunakan invers matriks transformasi *Arnold's Cat Map*, sebagai berikut:

$$\begin{cases} x_n \equiv (2x' - y') \ mod \ N \\ y_n \equiv (-x' + y') \ mod \ N \end{cases}$$

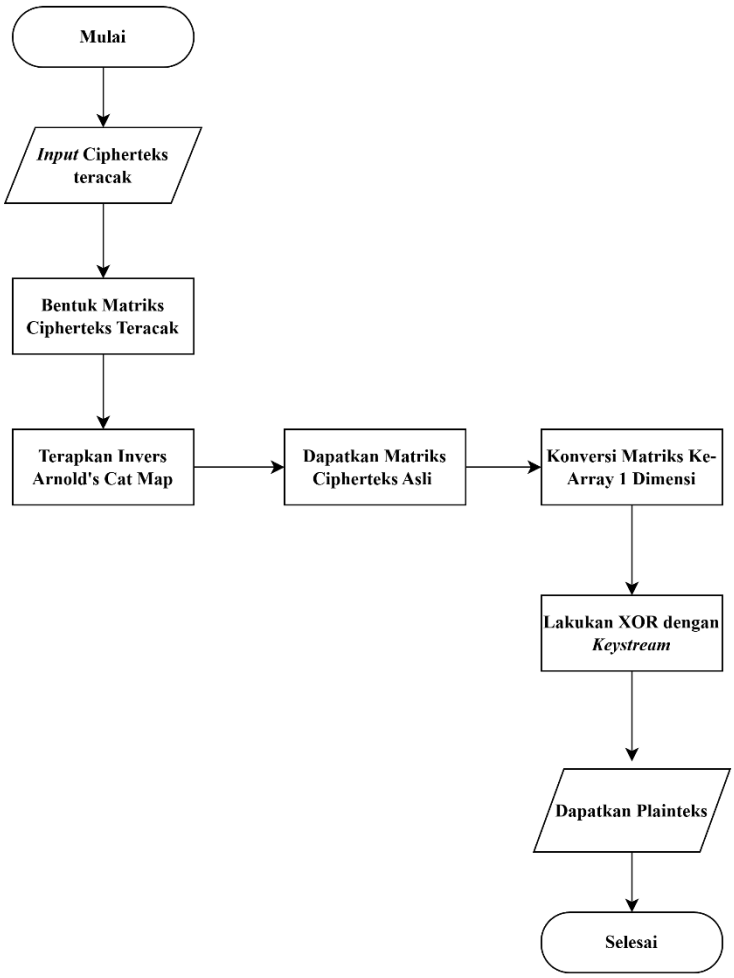
4. Dapatkan Matriks *Ciphertext* Asli
5. Konversi Matriks ke Array 1D (1 dimensi)

   Dengan mengubah bentuk matriks $N \times N$ hasil transformasi *Arnold's Cat Map* menjadi 1 baris, untuk mempermudah proses selanjutnya
6. Lakukan XOR dengan *keystream*
7. Dapatkan *Plaintext*

3.3 *Flowchart*



Gambar 3.1 *Flowchart* Tahapan Enkripsi



Gambar 3.2 *Flowchart* Tahapan Dekripsi

Keterangan :

: Mulai/Selesai

: *Input/ Output*

: Proses

: Pemilihan Kondisi

BAB IV

HASIL DAN PEMBAHASAN

4.1 Gambaran Umum Implementasi

Sistem enkripsi-dekripsi yang dibuat dalam penelitian ini bertujuan untuk melindungi pesan teks dengan menggunakan dasar-dasar kriptografi yang berbasis *chaos*, yang mengintegrasikan pembuatan *keystream pseudo*-acak menggunakan *Logistic Map*, operasi XOR untuk substitusi, serta transformasi *Arnold's Cat Map* untuk proses difusi. Secara umum, mekanisme ini berfungsi dengan mengubah teks biasa menjadi teks rahasia melalui beberapa langkah yang melibatkan unsur-unsur *chaos* untuk memperkuat keamanan, serta proses dekripsi yang merupakan kebalikan dari penyandian untuk mengembalikan pesan yang asli (*plaintext*).

Langkah-langkah implementasi dimulai dengan penghasilan *keystream* melalui *Logistic Map*, suatu peta *chaos* yang menciptakan rangkaian angka *pseudo*-acak berdasar pada parameter awal seperti nilai r dan x_0, yang selanjutnya digunakannya sebagai kunci dalam operasi XOR bersama *plaintext*. Namun sebelum digunakan untuk proses XOR, *keystream* tersebut terlebih dahulu mengalami tahap permutasi baris dan kolom. Pada tahap ini, *keystream* disusun dalam bentuk matriks berukuran $N \times N$, lalu dilakukan pengacakan posisi melalui teknik *circular shift* pada baris dan kolom. Nilai pergeseran (*shift*) dihitung dari keluaran iterasi *Logistic Map* menggunakan:

$$k \equiv \lfloor x_n \times 10 \rfloor mod\ N$$

nilai ini digunakan untuk menentukan banyaknya perpindahan pada baris (*row rotation*) maupun kolom (*column rotation*), sehingga struktur matriks *keystream* berubah pada setiap iterasi. Proses permutasi ini bertujuan meningkatkan *diffusion*,

sebab *keystream* yang telah diacak akan menghasilkan pola XOR yang sangat sulit diprediksi meskipun penyerang mengetahui sebagian hasil enkripsi. Dengan demikian, integrasi permutasi baris dan kolom berbasis *Logistic Map* memberikan lapisan acakan tambahan sebelum proses substitusi dilakukan.

Setelah *keystream* dipermutasi, proses dilanjutkan dengan operasi XOR anatara *plaintext* dan *keystream* tersebut. XOR dipilih karena bersifat sederhana, cepat, dan *self-inverse*, sehingga proses dekripsi dapat dilakukan dengan algoritma yang sama. Hasil XOR kemudian diproses lebih lanjut menggunakan algoritma *Arnold's Cat Map*, yang bertugas untuk mendistribusikan posisi karakter hasil XOR ke lokasi lain dalam matriks secara berulang. Transformasi ini menghasilkan efek difusi yang kuat dengan menyebarkan perubahan kecil pada *plaintext* ke seluruh *ciphertext*, menjadikan pola pesan asli sulit dilacak.

Pada tahap dekripsi, operasi dilakukan dengan cara kebalikan dari proses enkripsi, *ciphertext* terlebih dahulu dikenakan invers *Arnold's Cat Map* untuk mengembalikan posisi karakter ke pengaturan semula, kemudian dilakukan XOR menggunakan *keystream* yang sama (yang juga melalui proses permutasi baris dan kolom) sehingga diperoleh kembali *plaintext* asli. Kombinasi *Logistic Map*, permutasi baris dan kolom, XOR, dan *Arnold's Cat Map* memberikan dua lapis mekanisme keamanan *confusion* dan *diffusion* yang bekerja bersama memanfaatkan sifat ergodik, sensitivitas terhadap kondisi awal, dan ketidakterdugaan perilaku *chaos*. Dengan demikian, sistem ini mampu menghasilkan enkripsi yang kuat, cepat, dan tahan terhadap berbagai jenis serangan analisis kripto.

4.2 Implementasi Pembangkitan Kunci (*Keystream*)

Proses pembangkitan kunci (*keystream*) dilakukan dengan menggunakan algoritma *Logistic Map*, yang merupakan sistem dinamis satu dimensi yang dapat menghasilkan urutan bilangan *pseudo*-acak dengan sensitivitas tinggi terhadap kondisi awal dan parameter kontrol. Sifat *chaotic* ini membuat *Logistic Map* sangat cocok sebagai generator kunci kriptografi karena *output*nya sulit diprediksi tetapi masih dapat direproduksi jika nilai awal sama. Secara matematis, *Logistic Map* dirumuskan sebagai:

$$x_{n+1} = r.x_n(1 - x_n)$$

dengan x_0 sebagai kondisi awal dan r sebagai parameter kontrol dengan nilai $3{,}57 \leq r \leq 4{,}0$ untuk menghasilkan perilaku *chaos*.

Dalam implementasi penelitian ini, nilai x_0 dan r ditetapkan sebagai kunci utama. Pada awal dilakukan 100 iterasi awal (*warm-up iteration*) untuk menstabilkan sistem *Logistic Map* dan menghilangkan pengaruh kondisi transien, sehingga nilai-nilai berikutnya benar-benar merepresentasikan perilaku *chaos* yang digunakan sebagai pembangkit *keystream*. Hasil *keystream* yang didapatkan akan digunakan pada proses enkripsi dan dekripsi pesan teks. Berdasarkan prinsip kerja algoritma *Logistic Map* tersebut, dilakukan simulasi secara langsung untuk melihat bagaimana proses iterasi menghasilkan *keystream* yang bersifat *pseudo*-acak, dengan tahapan seperti yang terdapat pada BAB III:

1. Menentukan pesan teks yang akan di enkripsi.

   Plaintext yang akan digunakan dalam penelitian ini adalah "**Hai namaku Rofifah**".

2. Mengonversi pesan kedalam bentuk desimal (ASCII).

Tabel 4.1 Konversi Karakter Alfabet Menjadi Kode ASCII (desimal)

Karakter	Kode ASCII (desimal)
H	72
a	97
i	105
Space	32
n	110
a	97
m	109
a	97
k	107
u	117
Space	32
R	82
o	111
f	102
i	105
f	102
a	97
h	104

3. Menentukan parameter *Logistic Map*

Akan ditentukan parameter *Logistic Map* pada penelitian ini dengan memilih:

Nilai parameter *chaos* r $= 3{,}9$

Kondisi awal x_0 $= 0{,}2$

Jumlah iterasi pemanasan (*warm-up*) $= 100$

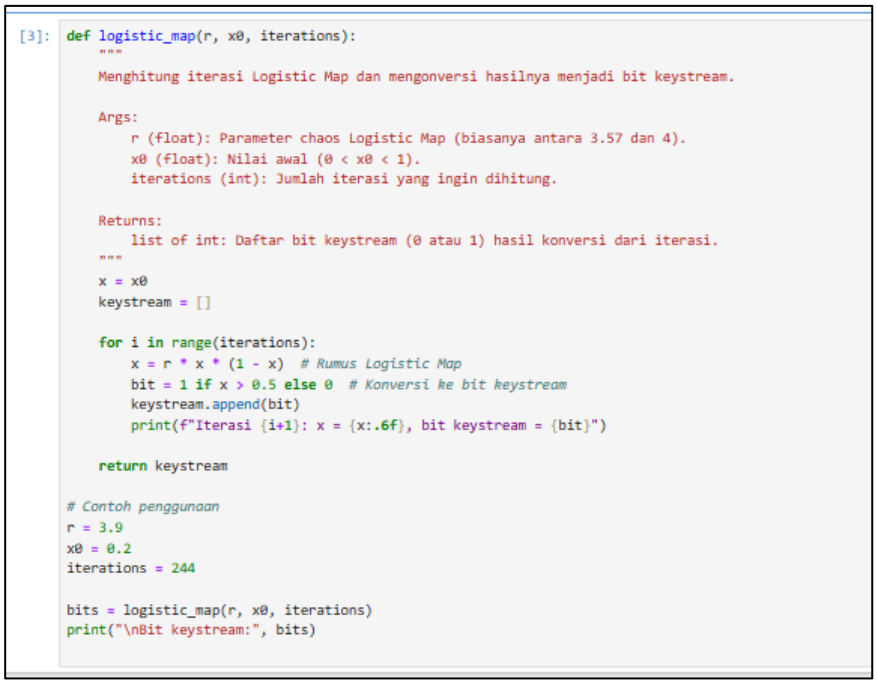
Total bit *keystream* yang dibutuhkan $= jumlah\ karakter\ pesan \times 8\ bit$

$= 18 \times 8 = 144$

4. Proses pembangkitan *keystream* dilakukan dengan menggunakan metode *Logistic Map* sebagai fungsi *chaos*. Parameter yang digunakan pada percobaan ini yaitu $r = 3{,}9$ dan nilai awal $x_0 = 0{,}2$. Jumlah iterasi yang dijalankan adalah 244 kali, di mana 100 iterasi pertama digunakan sebagai tahap *warm-up* untuk menstabilkan sistem dan menghilangkan pengaruh nilai awal. Dengan demikian, nilai hasil iterasi dari ke-101 hingga ke-244 digunakan sebagai *keystream* utama yang terdiri dari 144 bit. Hasil iterasi menunjukkan bahwa nilai x_n mengalami perubahan secara tidak teratur pada interval $(0, 1)$, yang menandakan bahwa sistem *Logistic Map* telah berada pada kondisi *chaotic*. Setiap nilai hasil iterasi dikonversi menjadi bit berdasarkan aturan:

$$K_i = \begin{cases} 1, & x_n > 0{,}5 \\ 0, & x_n \leq 0{,}5 \end{cases}$$

Perhitungan iterasi *Logistic Map* pada penelitian ini menggunakan bantuan program Python. Dengan parameter yang sudah ditentukan, diperoleh kode Python sebagai berikut:



```
[3]: def logistic_map(r, x0, iterations):
        """
        Menghitung iterasi Logistic Map dan mengonversi hasilnya menjadi bit keystream.

        Args:
            r (float): Parameter chaos Logistic Map (biasanya antara 3.57 dan 4).
            x0 (float): Nilai awal (0 < x0 < 1).
            iterations (int): Jumlah iterasi yang ingin dihitung.

        Returns:
            list of int: Daftar bit keystream (0 atau 1) hasil konversi dari iterasi.
        """
        x = x0
        keystream = []

        for i in range(iterations):
            x = r * x * (1 - x)  # Rumus Logistic Map
            bit = 1 if x > 0.5 else 0  # Konversi ke bit keystream
            keystream.append(bit)
            print(f"Iterasi {i+1}: x = {x:.6f}, bit keystream = {bit}")

        return keystream

    # Contoh penggunaan
    r = 3.9
    x0 = 0.2
    iterations = 244

    bits = logistic_map(r, x0, iterations)
    print("\nBit keystream:", bits)
```

Gambar 4.1 Kode Python Pembangkitan Kunci

Kode pada gambar 4.1 digunakan untuk menghitung iterasi fungsi *Logistic Map* berdasarkan parameter r dan x_0. Setiap hasil iterasi x_n dikonveri menjadi bit *keystream* menggunakan aturan, jika $x_n > 0{,}5$ maka bit $= 1$, dan $x_n \leq 0{,}5$ maka bit $= 0$. Hasilnya berupa deret bit yang bersifat acak sesuai karakteristik sistem *chaos Logistic Map*. Pada proses ini dapat dilakukan perhitungan secara manual terhadap beberapa iterasi pertama menggunakan rumus *Logistic Map*:

$$x_{n+1} = r.x_n(1 - x_n)$$

dengan $r = 3{,}9$ dan $x_0 = 0{,}2$, diperoleh hasil sebagai berikut:

Iterasi 1

$x_{n+1} = r.x_n(1 - x_n)$

$x_{0+1} = 3{,}9\,.\,0{,}2(1 - 0{,}2)$

$x_1 \quad = 0{,}624$

Iterasi 2

$x_{n+1} = r.x_n(1 - x_n)$

$x_{2+1} = 3{,}9\ .\ 0{,}9159(1 - 0{,}9159)$

$x_3 \quad = 0{,}30313$

Iterasi 3

$x_{n+1} = r.x_n(1 - x_n)$

$x_{2+1} = 3{,}9\ .\ 0{,}9159(1 - 0{,}9159)$

$x_3 \quad = 0{,}30313$

Iterasi 4

$x_{n+1} = r.x_n(1 - x_n)$

$x_{3+1} = 3{,}9\ .\ 0{,}30313(1 - 0{,}30313)$

$x_4 \quad = 0{,}8239$

Iterasi 5

$x_{n+1} = r.x_n(1 - x_n)$

$x_{4+1} = 3{,}9\ .\ 0{,}8239(1 - 0{,}8239)$

$x_5 \quad = 0{,}5657$

Selanjutnya akan ditampilakan hasil *output* dari kode Python pada gambar 4.1,

```
Iterasi 6: x = 0.958185, bit keystream = 1
Iterasi 7: x = 0.156258, bit keystream = 0
Iterasi 8: x = 0.514181, bit keystream = 1
Iterasi 9: x = 0.974216, bit keystream = 1
Iterasi 10: x = 0.097966, bit keystream = 0
Iterasi 11: x = 0.344638, bit keystream = 0
Iterasi 12: x = 0.880864, bit keystream = 1
Iterasi 13: x = 0.409276, bit keystream = 0
Iterasi 14: x = 0.942900, bit keystream = 1
Iterasi 15: x = 0.209975, bit keystream = 0
Iterasi 16: x = 0.646953, bit keystream = 1
Iterasi 17: x = 0.890778, bit keystream = 1
Iterasi 18: x = 0.379440, bit keystream = 0
Iterasi 19: x = 0.918314, bit keystream = 1
Iterasi 20: x = 0.292551, bit keystream = 0
```

Gambar 4.2 Hasil Iterasi Ke-6 Hingga Ke-20

Adapun hasil dari iterasi berikutnya dapat dilihat pada *Lampiran 1* sebagai bagian dokumentasi hasil. Setelah melakukan iterasi sebanyak 244 kali, maka akan didapatkan hasil *output* kode Python berupa bit *keystream* sebagai berikut:

Tabel 4.2 Output Kode Python

Iterasi	x_n	bit *keystream*
101	0,267664	0
102	0,764477	1
103	0,702202	1
104	0,815546	1
105	0,586680	1
106	0,945697	1
107	0,200280	0
108	0,624655	1
109	0,914398	1
110	0,305268	0

Hasil *output* kode Python untuk iterasi 101 hingga 110 ditunjukkan pada Tabel 4.2, sedangkan hasil *output* untuk iterasi selanjutnya secara lengkap

dapat dilihat pada *Lampiran 2*. Hasil *output* di atas menghasilkan *keystream* seperti berikut:

keystream pertama	$= 01111101$	*keystream* ke-10	$= 10111111$
keystream ke-2	$= 10111001$	*keystream* ke-11	$= 11011111$
keystream ke-3	$= 00101010$	*keystream* ke-12	$= 10110101$
keystream ke-4	$= 11100101$	*keystream* ke-13	$= 00101011$
keystream ke-5	$= 10101110$	*keystream* ke-14	$= 11011001$
keystream ke-6	$= 01010110$	*keystream* ke-15	$= 01011111$
keystream ke-7	$= 01010111$	*keystream* ke-16	$= 10110111$
keystream ke-8	$= 01001001$	*keystream* ke-17	$= 01110101$
keystream ke-9	$= 01100100$	*keystream* ke-18	$= 11001010$

berikut merupakan *keystream* yang akan dilakukan pada proses enkripsi dan dekripsi pesan teks.

4.3 Implementasi Proses Enkripsi Pesan Teks

Proses enkripsi diterapkan pada pesan teks dengan memanfaatkan algoritma *Logistic Map* dan *Arnold's Cat Map*. Dalam tahap ini, *plaintext* yang sudah diubah ke kode ASCII dikonversi menjadi format biner 8 bit, kemudian digabungkan dengan *keystream* yang dihasilkan dari *Logistic Map* melalui operasi XOR untuk menciptakan *ciphertext* awal. Berikutnya, posisi *ciphertxt* tersebut diacak dengan menggunakan transformasi *Arnold's Cap Map* agar difusi diperkuat dan pola karakter asli tersembunyi. Pada akhir tahap ini, diperoleh *ciphertext* yang tidak memperlihatkan pola teratur dan sudah siap untuk dikirimkan dengan aman ke penerima.

Pada tahap ini, proses enkripsi pesan teks dilakukan secara manual sesuai tahapan algoritma yang telah dirancang. Pesan teks (*plaintext*) terlebih dahulu dikonversi menjadi kode ASCII dan direpresentasikan dalam bentuk biner 8 bit. Setiap bit hasil konversi dienkripsi menggunakan *keystream* dari iterasi *Logistic Map* melalui operasi XOR, sebelum dilakukan operasi XOR dilakukan permutasi baris dan kolom untuk keystream yang dihasilkan melalui iterasi *Logistic Map*, *keystream* diatur dalam format matriks persegi, diikuti oleh proses permutasi pada baris dan kolomnya dengan menerapkan pergeseran melingkar (*circular shift*). Besaran pergeseran tersebut ditetapkan berdasarkan output *Logistic Map* melalui operasi $\lfloor x_n \times 10 \rfloor mod N$. Permutasi ini dimaksudkan untuk mengacak susunan elemen-elemen *keystream*, sehingga distribusi kunci menjadi lebih acak dan meningkatkan derajat *confusion* sebelum diterapkan dalam operasi XOR. Setelah itu dilakukan operasi XOR dengan *keystream* hasil permutasi baris dan kolom dan menghasilkan deret biner *ciphertext* awal. Selanjutnya, bit-bit tersebut diacak menggunakan transformasi *Arnold's Cat Map* untuk memperkuat keamanan dan menyamarkan pola karakter asli. Seluruh proses dihitung secara manual untuk menunjukkan keakuratan langkah enkripsi sesuai teori. Berikut implementasi langsung enkripsi pesan teks (*plaintext*):

1. Menentukan suatu pesan (M)

   Pada penelitian ini menggunakan pesan teks (*plaintext*) berupa "**Hai namaku Rofifah**"

2. Mengubah/mengonversi *plaintext* menjadi bilangan biner 8 bit

Tabel 4.3 Konversi Kode ASCII (desimal) Ke Biner 8 Bit

Kode ASCII (desimal)	Biner 8 bit
72	01001000
97	01100001
105	01101001
32	00100000
110	01101110
97	01100001
109	01101100
97	01100001
107	01101011
117	01110101
32	00100000
82	01010010
111	01101111
102	01100110
105	01101001
102	01100110
97	01100001
104	01101000

Tabel 4.3 menunjukkan hasil konversi dari kode ASCII (desimal) menjadi biner 8 bit.

3. Melakukan permutasi baris dan kolom pada *keystream* yang telah dibangkitkan dari *Logistic Map*.

   Sebelum dilakukan operasi XOR, dilakukan permutasi baris dan kolom untuk *keystream*. Sebelum melakukan permutasi baris dan kolom susun *keystream* yang diperoleh dari iterasi *Logistic Map* ke dalam bentuk matriks persegi, sebagai berikut:

$$\begin{bmatrix} 01111101 & 10111001 & 00101010 & 11100101 & 10101110 \\ 01010110 & 01010111 & 01001001 & 01100100 & 10111111 \\ 11011111 & 10110101 & 00101011 & 11011001 & 01011111 \\ 10110111 & 01110101 & 11001010 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 & 00000000 \end{bmatrix}$$

untuk menentukan banyaknya pergeseran (*shift*) menggunakan rumus sebagai berikut:

$$k \equiv \lfloor x_n \times M \rfloor mod\ N$$

dengan:

x_n : Nilai iterasi Logistic Map ke-n

N : Ukuran matriks (misal 5 untuk matriks 5×5)

k : Langkah geser atau besar pergeseran (*shift*)

M : Konstanta skala (sering digunakan $= 10, 100, 1000$

pada kali ini menggunakan konstanta skala sebesar 10 dan $N = 5$, karena besar matriks adalah 5×5, maka rumus yang akan digunakan adalah sebagai berikut:

$$k \equiv \lfloor x_n \times 10 \rfloor mod\ 5$$

akan di mulai dengan iterasi ke-101, diperoleh nilai:

$$x_{101} = 0{,}267664$$

dalam algoritma enkripsi berbasis *chaos*, nilai *chaos* digunakan untuk menentukan indeks permutasi (*shift*), maka akan didapatkan nilai untuk menentukan besar pergeseran baris sebagai berikut:

$$k_r \equiv \lfloor x_{101} \times 10 \rfloor\ mod\ 5$$
$$\equiv \lfloor 0{,}267664 \times 10 \rfloor\ mod\ 5$$
$$\equiv \lfloor 2{,}67664 \rfloor\ mod\ 5$$
$$\equiv 2\ mod\ 5$$
$$\equiv 2$$

dari perhitungan diatas didapatkan banyak pergeseran (*shift*) pada baris, yaitu 2 kali.

Selanjutnya, lakukan proses permutasi baris menggunakan rumus sebagai berikut:

$$Row'[i] \equiv Row[(i - k_{rb}) mod\ N]$$

karena nilai k_r sudah diketahui, yakni 2. Maka perhitungan permutasi baris sebagai berikut:

Baris 1

$$Row'[i] \equiv Row[(i - k_r) mod\ N]$$

$$Row'[0] \equiv Row[(0 - 2) mod\ 5]$$

$$\equiv Row[-2\ mod\ 5]$$

$$\equiv Row[3]$$

Baris 2

$$Row'[i] \equiv Row[(i - k_r) mod\ N]$$

$$Row'[1] \equiv Row[(1 - 2) mod\ 5]$$

$$\equiv Row[-1\ mod\ 5]$$

$$\equiv Row[4]$$

Baris 3

$$Row'[i] \equiv Row[(i - k_r) mod\ N]$$

$$Row'[2] \equiv Row[(2 - 2) mod\ 5]$$

$$\equiv Row[0\ mod\ 5]$$

$$\equiv Row[0]$$

Baris 4

$$Row'[i] \equiv Row[(i - k_r) mod\ N]$$

$$Row'[3] \equiv Row[(3 - 2) mod\ 5]$$

$$\equiv Row[1\ mod\ 5]$$

$$\equiv Row[1]$$

Baris 5

$$Row'[i] \equiv Row[(i - k_r) mod\ N]$$

$$Row'[4] \equiv Row[(4 - 2) mod\ 5]$$

$$\equiv Row[2\ mod\ 5]$$

$$\equiv Row[2]$$

dari perhitungan diatas didapatkan:

Baris 1 berpindah ke-baris 3

Baris 2 berpindah ke-baris 4

Baris 3 berpindah ke-baris 5

Baris 4 berpindah ke-baris 1

Baris 5 berpindah ke-baris 2

jika dalam bentuk matriks, maka akan seperti ini:

$$\begin{bmatrix} 10110111 & 01110101 & 11001010 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 & 00000000 \\ 01111101 & 10111001 & 00101010 & 11100101 & 10101110 \\ 01010110 & 01010111 & 01001001 & 01100100 & 10111111 \\ 11011111 & 10110101 & 00101011 & 11011001 & 01011111 \end{bmatrix}$$

Tahapan selanjutnya adalah permutasi kolom, tetapi sebelum melakukan permutasi kolom akan dilakukan perhitungan untuk mengetahui banyak pergeseran (*shift*) kolom dengan rumus sebagai berikut:

$$k_c \equiv \lfloor x_{n+1} \times 10 \rfloor\ mod\ 5$$

akan di lakukan dengan iterasi ke-102, diperoleh nilai:

$$x_{102} = 0{,}764477$$

maka akan didapatkan nilai untuk menentukan banyak pergeseran kolom sebagai berikut:

$k_c \equiv \lfloor x_{102} \times 10 \rfloor \; mod \; 5$

$\equiv \lfloor 0{,}764477 \times 10 \rfloor \; mod \; 5$

$\equiv \lfloor 7{,}64477 \rfloor \; mod \; 5$

$\equiv 7 \; mod \; 5$

$\equiv 2$

dari perhitungan diatas didapatkan besar pergeseran (*shift*) pada kolom, yaitu 2 kali. Selanjutnya, lakukan proses permutasi baris menggunakan rumus sebagai berikut:

$$Col'[i] \equiv Col[(i - k_{ck}) mod \; N]$$

karena nilai k_c sudah diketahui, yakni 2. Maka perhitungan permutasi kolom sebagai berikut:

Kolom 1

$Col'[j] \equiv Col\,[(j - k_r) mod \; N]$

$Col'[0] \equiv Col\,[(0 - 2) mod \; 5]$

$\equiv Col\,[-2 \; mod \; 5]$

$\equiv Col\,[3]$

Kolom 2

$Col'[j] \equiv Col\,[(j - k_r) mod\ N]$

$Col'[1] \equiv Col\,[(1 - 2) mod\ 5]$

$\equiv Col\,[-1\ mod\ 5]$

$\equiv Col\,[4]$

Kolom 3

$Col'[j] \equiv Col\,[(j - k_r) mod\ N]$

$Col'[2] \equiv Col\,[(2 - 2) mod\ 5]$

$\equiv Col\,[0\ mod\ 5]$

$\equiv Col\,[0]$

Kolom 4

$Col'[j] \equiv Col\,[(j - k_r) mod\ N]$

$Col'[3] \equiv Col\,[(3 - 2) mod\ 5]$

$\equiv Col\,[1\ mod\ 5]$

$\equiv Col\,[1]$

Kolom 5

$Col'[j] \equiv Col\,[(j - k_r) mod\ N]$

$Col'[4] \equiv Col\,[(4 - 2) mod\ 5]$

$\equiv Col\,[2\ mod\ 5]$

$\equiv Col\,[2]$

dari perhitungan diatas didapatkan:

Kolom 1 berpindah ke-kolom 3

Kolom 2 berpindah ke-kolom 4

Kolom 3 berpindah ke-kolom 5

Kolom 4 berpindah ke-kolom 1

Kolom 5 berpindah ke-kolom 2

jika dalam bentuk matriks, maka akan seperti ini:

$$\begin{bmatrix} 00000000 & 00000000 & 10110111 & 01110101 & 11001010 \\ 00000000 & 00000000 & 00000000 & 00000000 & 00000000 \\ 11100101 & 10101110 & 01111101 & 10111001 & 00101010 \\ 01100100 & 10111111 & 01010110 & 01010111 & 01001001 \\ 11011001 & 01011111 & 11011111 & 10110101 & 00101011 \end{bmatrix}$$

di atas merupakan matriks final untuk proses permutasi baris dan kolom, dan untuk 18 byte awal akan diambil untuk proses XOR dengan *plaintext*. Dimulai dari yang paling kiri kearah kanan. *Keystream* yang akan digunakan pada operasi XOR, yaitu:

$$[00000000, 00000000, 10110111, 01110101, 11001010, 00000000,$$
$$00000000, 00000000, 00000000, 00000000, 11100101, 10101110,$$
$$01111101, 10111001, 00101010, 01100100, 10111111, 01010110]$$

4. Melakukan XOR antara setiap byte *plaintext* dengan byte *keystream* hasil permutasi baris dan kolom yang sesuai:

$$Ciphertext = plaintext\ (M) \oplus keystream\ (K)$$

sebelum dilakukan proses enkripsi terhadap *plaintext* secara keseluruhan, setiap byte dari *plaintext* dikombinasikan dengan byte *keystream* hasil permutasi baris dan kolom yang bersesuaian menggunakan operasi logika XOR. Operasi ini berfungsi untuk mengubah nilai biner dari *plaintext* menjadi *ciphertext* dengan memanfaatkan sifat XOR yang menghasilkan keluaran berbeda ketika dua bit inputnya tidak sama. Tahapan ini merupakan inti dari proses enkripsi karena menentukan hasil akhir *ciphertext* berdasarkan nilai *keystream* yang bersifat acak. Setelah prinsip operasi XOR dijelaskan,

tahapan selanjutnya adalah penerapan perhitungan XOR secara langsung terhadap data pesan teks untuk memperoleh hasil *ciphertext*. Berikut adalah tahapan perhitungan XOR:

karakter pertama $= 01001000 \oplus 00000000 = 01001000\ (72)$

karakter ke-2 $= 01100001 \oplus 00000000 = 11011110\ (97)$

karakter ke-3 $= 01101001 \oplus 10110111 = 11011110\ (222)$

karakter ke-4 $= 00100000 \oplus 01110101 = 01010101\ (85)$

karakter ke-5 $= 01101110 \oplus 11001010 = 10100100\ (164)$

karakter ke-6 $= 01100001 \oplus 00000000 = 01100001\ (97)$

karakter ke-7 $= 01101100 \oplus 00000000 = 01101100\ (109)$

karakter ke-8 $= 01100001 \oplus 00000000 = 01100001\ (97)$

karakter ke-9 $= 01101011 \oplus 00000000 = 01101011\ (107)$

karakter ke-10 $= 01110101 \oplus 00000000 = 01110101\ (117)$

karakter ke-11 $= 00100000 \oplus 11100101 = 11000101\ (197)$

karakter ke-12 $= 01010010 \oplus 10101110 = 11111100\ (252)$

karakter ke-13 $= 01101111 \oplus 01111101 = 00010010\ (18)$

karakter ke-14 $= 01100110 \oplus 10111001 = 11011111\ (223)$

karakter ke-15 $= 01101001 \oplus 00101010 = 01000011\ (67)$

karakter ke-16 $= 01100110 \oplus 01100100 = 00000010\ (2)$

karakter ke-17 $= 01100001 \oplus 10111111 = 11011110\ (222)$

karakter ke-18 $= 01101000 \oplus 01010110 = 00111110\ (62)$

diperoleh hasil dari perhitungan XOR *plaintext* dengan *keystream*.

5. Menentukan dan menyusun c*iphertext* hasil XOR ke dalam matriks persegi

setelah diperoleh hasil *ciphertext* dari proses XOR, tahap selanjutnya adalah menyususnnya ke dalam matriks persegi. Penyususnan ini bertujuan untuk mempermudah proses transformasi menggunakan *Arnold's Cat Map* pada tahap berikutnya. Dengan merepresentasikan *ciphertext* dalam bentuk matriks, setiap elemen dapat dipetakan dan diacak posisinya secara sistematis sehingga meningkatkan tingkat pengacakan dan keamanan pesan teks. Pada penentuan besar matriks $N \times N$, jika banyak karakter pada pesan teks tidak membentuk matriks persegi sempurna maka akan di tambah *padding* dengan nilai 0. Dari hasil *ciphertext* yang diperoleh dari proses XOR dengan *keystream* hasil permutasi baris dan kolom didapat 18 karakter dan besar matriks yang mendekati yaitu, matriks 5×5 dengan menambahkan beberapa *padding* berupa nilai 0, hasil matriks *ciphertext* sebagai berikut:

$$\begin{bmatrix} 72 & 97 & 222 & 85 & 164 \\ 97 & 109 & 97 & 107 & 117 \\ 197 & 252 & 18 & 223 & 67 \\ 2 & 222 & 62 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Pada matriks *ciphertext* di atas membutuhkan tujuh *padding* agar besar matriks menjadi matriks persegi sempurna, yakni matriks 5×5.

6. Implementasikan *Arnold's Cat Map* untuk setiap entri dalam matriks ukuran 5×5, dimana entri matriks dimulai dengan $(0,0)$.

   Setelah *ciphertext* disusun dalam bentuk matriks, tahapan selanjutnya adalah menerapkan transformasi *Arnold's Cat Map* pada setiap entri dalam matriks berukuran 5×5, dengan

$$\begin{cases} x_{n+1} \equiv (x_n + y_n) \ mod \ N \\ y_{n+1} \equiv (x_n + 2y_n) \ mod \ N \end{cases}$$

transformasi ini digunakan untuk mengacak posisi elemen-elemen *ciphertext* sehingga pola asli menjadi sulit dikenali. Dengan demikian, distribusi data menjadi lebih acak dan keamanan hasil enkripsi semakin meningkat sebelum proses dekripsi dilakukan pada tahap selanjutnya. Berikut implementasi langsung algoritma *Arnold's Cat Map* terhadap matriks *ciphertext*:

a. Entri awal $(0, 0)$

$$\begin{bmatrix} x'_0 \\ y'_0 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 0 + 1 \cdot 0 \\ 1 \cdot 0 + 2 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{0} \end{bmatrix}$$

$(x'_0, y'_0) = (\bar{0}, \bar{0})$

diperoleh bahwa entri baru tetapberada pada posisi $(\bar{0}, \bar{0})$

b. Entri awal $(0, 1)$

$$\begin{bmatrix} x'_0 \\ y'_1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} (1 \cdot 0 + 1 \cdot 1) \\ (1 \cdot 0 + 2 \cdot 1) \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{2} \end{bmatrix}$$

$(x'_0, y'_1) = (\bar{1}, \bar{2})$

diperoleh bahwa entri baru berada pada posisi $(\bar{1}, \bar{2})$

c. Entri awal $(0, 2)$

$$\begin{bmatrix} x'_0 \\ y'_2 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 0 + 1 \cdot 2 \\ 1 \cdot 0 + 2 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{4} \end{bmatrix}$$

$(x'_0, y'_2) = (\bar{2}, \bar{4})$

diperoleh bahwa entri baru berada pada posisi $(\bar{2}, \bar{4})$

d. Entri awal $(0, 3)$

$$\begin{bmatrix} x'_0 \\ y'_3 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 0 + 1 \cdot 3 \\ 1 \cdot 0 + 2 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{1} \end{bmatrix}$$

$(x'_0, y'_3) = (\bar{3}, \bar{1})$

diperoleh bahwa entri baru berada pada posisi $(\bar{3}, \bar{1})$

e. Entri awal $(0, 4)$

$$\begin{bmatrix} x'_0 \\ y'_4 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 4 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 0 + 1 \cdot 4 \\ 1 \cdot 0 + 2 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{3} \end{bmatrix}$$

$(x'_0, y'_4) = (\bar{4}, \bar{3})$

diperoleh bahwa entri baru berada pada posisi $(\bar{4}, \bar{3})$

f. Entri awal $(1, 0)$

$$\begin{bmatrix} x'_1 \\ y'_0 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 1 + 2 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{1} \end{bmatrix}$$

$(x'_1, y'_0) = (\bar{1}, \bar{1})$

diperoleh bahwa entri baru berada pada posisi $(\bar{1}, \bar{1})$

g. Entri awal $(1, 1)$

$$\begin{bmatrix} x'_1 \\ y'_1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 1 + 1 \cdot 1 \\ 1 \cdot 1 + 2 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{3} \end{bmatrix}$$

$(x'_1, y'_1) = (\bar{2}, \bar{3})$

diperoleh bahwa entri baru berada pada posisi $(\bar{2}, \bar{3})$

h. Entri awal $(1, 2)$

$$\begin{bmatrix} x'_1 \\ y'_2 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 1 + 1 \cdot 2 \\ 1 \cdot 1 + 2 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{0} \end{bmatrix}$$

$(x'_1, y'_2) = (\bar{3}, \bar{0})$

diperoleh bahwa entri baru berada pada posisi $(\bar{3}, \bar{0})$

i. Entri awal $(1, 3)$

$$\begin{bmatrix} x'_1 \\ y'_3 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 1 + 1 \cdot 3 \\ 1 \cdot 1 + 2 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{2} \end{bmatrix}$$

$(x'_1, y'_3) = (\bar{4}, \bar{2})$

diperoleh bahwa entri baru berada pada posisi $(\bar{4}, \bar{2})$

j. Entri awal $(1, 4)$

$$\begin{bmatrix} x'_1 \\ y'_4 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 1 + 1 \cdot 4 \\ 1 \cdot 1 + 2 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{4} \end{bmatrix}$$

$(x'_1, y'_4) = (\bar{0}, \bar{4})$

diperoleh bahwa entri baru berada pada posisi $(\bar{0}, \bar{4})$

k. Entri awal $(2, 0)$

$$\begin{bmatrix} x'_2 \\ y'_0 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 2 + 1 \cdot 0 \\ 1 \cdot 2 + 2 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{2} \end{bmatrix}$$

$(x'_2, y'_0) = (\bar{2}, \bar{2})$

diperoleh bahwa entri baru berada pada posisi $(\bar{2}, \bar{2})$

l. Entri awal $(2,1)$

$$\begin{bmatrix} x'_2 \\ y'_1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 2 + 1 \cdot 1 \\ 1 \cdot 2 + 2 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{4} \end{bmatrix}$$

$(x'_2, y'_1) = (\bar{3}, \bar{4})$

diperoleh bahwa entri baru berada pada posisi $(\bar{3}, \bar{4})$

m. Entri awal $(2,2)$

$$\begin{bmatrix} x'_2 \\ y'_2 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 2 + 1 \cdot 2 \\ 1 \cdot 2 + 2 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{1} \end{bmatrix}$$

$(x'_2, y'_2) = (\bar{4}, \bar{1})$

diperoleh bahwa entri baru berada pada posisi $(\bar{4}, \bar{1})$

n. Entri awal $(2,3)$

$$\begin{bmatrix} x'_2 \\ y'_3 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 2 + 1 \cdot 3 \\ 1 \cdot 2 + 2 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{3} \end{bmatrix}$$

$(x'_2, y'_3) = (\bar{0}, \bar{3})$

diperoleh bahwa entri baru berada pada posisi $(\bar{0}, \bar{3})$

o. Entri awal $(2,4)$

$$\begin{bmatrix} x'_2 \\ y'_4 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 2 + 1 \cdot 4 \\ 1 \cdot 2 + 2 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{0} \end{bmatrix}$$

$(x'_2, y'_4) = (\bar{1}, \bar{0})$

diperoleh bahwa entri baru berada pada posisi $(\bar{1}, \bar{0})$

p. Entri awal $(3, 0)$

$$\begin{bmatrix} x'_3 \\ y'_0 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 3 + 1 \cdot 0 \\ 1 \cdot 3 + 2 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{3} \end{bmatrix}$$

$(x'_3, y'_0) = (\bar{3}, \bar{3})$

diperoleh bahwa entri baru berada pada posisi $(\bar{3}, \bar{3})$

q. Entri awal $(3, 1)$

$$\begin{bmatrix} x'_3 \\ y'_1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 3 + 1 \cdot 1 \\ 1 \cdot 3 + 2 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{0} \end{bmatrix}$$

$(x'_3, y'_1) = (\bar{4}, \bar{0})$

diperoleh bahwa entri baru berada pada posisi $(\bar{4}, \bar{0})$

r. Entri awal $(3, 2)$

$$\begin{bmatrix} x'_3 \\ y'_2 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 3 + 1 \cdot 2 \\ 1 \cdot 3 + 2 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{2} \end{bmatrix}$$

$(x'_3, y'_2) = (\bar{0}, \bar{2})$

diperoleh bahwa entri baru berada pada posisi $(\bar{0}, \bar{2})$

s. Entri awal $(3, 3)$

$$\begin{bmatrix} x'_3 \\ y'_3 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 3 + 1 \cdot 3 \\ 1 \cdot 3 + 2 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{4} \end{bmatrix}$$

$(x'_3, y'_3) = (\bar{1}, \bar{4})$

diperoleh bahwa entri baru berada pada posisi $(\bar{1}, \bar{4})$

t. Entri awal $(3, 4)$

$$\begin{bmatrix} x'_3 \\ y'_4 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 3 + 1 \cdot 4 \\ 1 \cdot 3 + 2 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{1} \end{bmatrix}$$

$(x'_3, y'_4) = (\bar{2}, \bar{1})$

diperoleh bahwa entri baru berada pada posisi $(\bar{2}, \bar{1})$

u. Entri awal $(4, 0)$

$$\begin{bmatrix} x'_4 \\ y'_0 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 4 + 1 \cdot 0 \\ 1 \cdot 4 + 2 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{4} \end{bmatrix}$$

$(x'_4, y'_0) = (\bar{4}, \bar{4})$

diperoleh bahwa entri baru berada pada posisi $(\bar{4}, \bar{4})$

v. Entri awal $(4, 1)$

$$\begin{bmatrix} x'_4 \\ y'_1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 4 + 1 \cdot 1 \\ 1 \cdot 4 + 2 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{1} \end{bmatrix}$$

$(x'_4, y'_1) = (\bar{0}, \bar{1})$

diperoleh bahwa entri baru berada pada posisi $(\bar{0}, \bar{1})$

w. Entri awal $(4, 2)$

$$\begin{bmatrix} x'_4 \\ y'_2 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 4 + 1 \cdot 2 \\ 1 \cdot 4 + 2 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{3} \end{bmatrix}$$

$(x'_4, y'_2) = (\bar{1}, \bar{3})$

diperoleh bahwa entri baru berada pada posisi $(\bar{1}, \bar{3})$

x. Entri awal $(4, 3)$

$$\begin{bmatrix} x'_4 \\ y'_3 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 4 + 1 \cdot 3 \\ 1 \cdot 4 + 2 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{0} \end{bmatrix}$$

$(x'_4, y'_3) = (\bar{2}, \bar{0})$

diperoleh bahwa entri baru berada pada posisi $(\bar{2}, \bar{0})$

y. Entri awal $(4, 4)$

$$\begin{bmatrix} x'_4 \\ y'_4 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 1 \cdot 4 + 1 \cdot 4 \\ 1 \cdot 4 + 2 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{2} \end{bmatrix}$$

$(x'_4, y'_4) = (\bar{3}, \bar{2})$

diperoleh bahwa entri baru berada pada posisi $(\bar{3}, \bar{2})$

Setelah perhitungan semua karakter menggunakan algoritma *Arnold's Cat Map*, akan didapatkan hasil seperti di atas untuk entri baru, atau letak baru setelah pengacakan.

7. Setelah pengacakan selesai didapatkan matriks yang sudah acak sebagai berikut:

$$\begin{bmatrix} 72 & 0 & 62 & 223 & 117 \\ 67 & 97 & 97 & 0 & 0 \\ 0 & 0 & 197 & 109 & 222 \\ 97 & 85 & 0 & 2 & 252 \\ 222 & 18 & 107 & 164 & 0 \end{bmatrix}$$

8. Diambil kembali *ciphertext* dari matriks yang sudah diacak dan pesan siap dikirim.

Setelah semua proses enkripsi dilakukan maka didapatkanlah *ciphertext* yang siap untuk dikirimkan kepada penerima pesan teks, berikut adalah *ciphertext*: **“H > ß u C a a Å m Þ a U STX ü Þ DC2 k ¤”.**

4.4 Implementasi Proses Dekripsi Pesan Teks

Tahapan proses dekripsi pesan teks yang merupakan kebalikan dari proses enkripsi sebelumnya. Tujuan dari tahap ini adalah untuk mengembalikan *ciphertext* menjadi pesan asli (*plaintext*) dengan mengikuti urutan tahapan yang terbalik dari proses enkripsi. Proses diawali dengan menerapkan invers matriks transformasi *Arnold’s Cat Map* untuk mengembalikan posisi elemen-elemen matriks *ciphertext* ke posisi semula. Selanjutnya, hasil transformasi tersebut dikonversi ke bentuk deret biner dan dilakukan operasi XOR dengan deret *keystream* yang sama seperti pada tahap enkripsi. Hasil dari proses XOR ini kemudian diubah kembali ke kode ASCII dan dikonversi menjadi karakter teks asli. Dengan demikian, proses dekripsi ini membuktikan bahwa algoritma yang diterapkan mampu mengembalikan pesan terenkripsi secara utuh sesuai pesan awal. Berikut pengimplementasian secara langsung terhadap *ciphertext* yang sudah didapatkan dari proses enkripsi sebelumnya:

1. Menerima *ciphertext*.

   Ciphertext yang diterima dari proses enkripsi sebelumnya pada penelitian ini adalah **“H > ß u C a a Å m Þ a U STX ü Þ DC2 k ¤”.**

2. Membentuk *ciphertext* acak menjadi matriks $N \times N$ sesuai ukuran yang digunakan saat proses enkripsi.

   Pada proses enkripsi ukuran matriks yang digunakan adalah 5×5, maka didapatkan matriks ter-acak 5×5 sebagai berikut:

$$\begin{bmatrix} 72 & 0 & 62 & 223 & 117 \\ 67 & 97 & 97 & 0 & 0 \\ 0 & 0 & 197 & 109 & 222 \\ 97 & 85 & 0 & 2 & 252 \\ 222 & 18 & 107 & 164 & 0 \end{bmatrix}$$

3. Menerapkan invers matriks tranformasi *Arnold's Cat Map*
4. Didapatkan matriks *ciphertext* asli

   Proses ini dilakukan untuk mengembalikan posisi elemen-elemen matriks *ciphertext* yang sebelumnya telah diacak menggunakan transformasi *Arnold's Cat Map*. Invers matriks transformasi *Arnold's Cat Map* yang berbentuk sebagai berikut:

$$\begin{cases} x_n \equiv (2x' - y')mod\ N \\ y_n \equiv (-x' + y')mod\ N \end{cases}$$

   berfungsi mengembalikan proses pengacakan dengan cara menerapkan rumus transformasi kebalikannya pada setiap entri dalam matriks. Dengan demikian, susunan elemen *ciphertext* dapat kembali ke posisi semula sebelum dilakukan operasi XOR, sehingga pesan terenkripsi dapat didekripsi menjadi *plaintext* secara tepat. Akan ditampilkan perhitungan dengan invers matriks transformasi *Arnold's Cat Map*:

   a. Entri baru $(0, 0)$

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 0 + (-1) \cdot 0 \\ (-1) \cdot 0 + 1 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{0} \end{bmatrix}$$

$$(x_0, y_0) = (\bar{0}, \bar{0})$$

   diperoleh bahwa entri awal tetap berada pada posisi $(\bar{0}, \bar{0})$

b. Entri baru $(0,1)$

$$\begin{bmatrix} x_0 \\ y_1 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 0 + (-1) \cdot 1 \\ (-1) \cdot 0 + 1 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{1} \end{bmatrix}$$

$(x_0, y_0) = (\bar{4}, \bar{1})$

diperoleh bahwa entri awal berada pada posisi $(\bar{4}, \bar{1})$

c. Entri baru $(0,2)$

$$\begin{bmatrix} x_0 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 0 + (-1) \cdot 2 \\ (-1) \cdot 0 + 1 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{2} \end{bmatrix}$$

$(x_0, y_2) = (\bar{3}, \bar{2})$

diperoleh bahwa entri awal berada pada posisi $(\bar{3}, \bar{2})$

d. Entri baru $(0,3)$

$$\begin{bmatrix} x_0 \\ y_3 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$$\equiv \begin{bmatrix} 2 \cdot 0 + (-1) \cdot 3 \\ (-1) \cdot 0 + 1 \cdot 3 \end{bmatrix} \equiv \begin{bmatrix} \bar{2} \\ \bar{3} \end{bmatrix}$$

$(x_0, y_3) = (\bar{2}, \bar{3})$

diperoleh bahwa entri awal berada pada posisi $(\bar{2}, \bar{3})$

e. Entri baru $(0,4)$

$$\begin{bmatrix} x_0 \\ y_4 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 4 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 0 + (-1) \cdot 4 \\ (-1) \cdot 0 + 1 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{4} \end{bmatrix}$$

$(x_0, y_4) = (\bar{1}, \bar{4})$

diperoleh bahwa entri awal berada pada posisi $(\bar{1}, \bar{4})$

f. Entri baru $(1, 0)$

$$\begin{bmatrix} x_1 \\ y_0 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 1 + (-1) \cdot 0 \\ (-1) \cdot 1 + 1 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{4} \end{bmatrix}$$

$$(x_1, y_0) = (\bar{2}, \bar{4})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{2}, \bar{4})$

g. Entri baru $(1, 1)$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 1 + (-1) \cdot 1 \\ (-1) \cdot 1 + 1 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{0} \end{bmatrix}$$

$$(x_1, y_1) = (\bar{1}, \bar{0})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{1}, \bar{0})$

h. Entri baru $(1, 2)$

$$\begin{bmatrix} x_1 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 1 + (-1) \cdot 2 \\ (-1) \cdot 1 + 1 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{1} \end{bmatrix}$$

$$(x_1, y_2) = (\bar{0}, \bar{1})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{0}, \bar{1})$

i. Entri baru $(1, 3)$

$$\begin{bmatrix} x_1 \\ y_3 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 1 + (-1) \cdot 3 \\ (-1) \cdot 1 + 1 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{2} \end{bmatrix}$$

$$(x_1, y_3) = (\bar{4}, \bar{2})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{4}, \bar{2})$

j. Entri baru $(1, 4)$

$$\begin{bmatrix} x_1 \\ y_4 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 1 + (-1) \cdot 4 \\ (-1) \cdot 1 + 1 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{3} \end{bmatrix}$$

$(x_1, y_4) = (\bar{3}, \bar{3})$

diperoleh bahwa entri awal berada pada posisi $(\bar{3}, \bar{3})$

k. Entri baru $(2, 0)$

$$\begin{bmatrix} x_2 \\ y_0 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 2 + (-1) \cdot 0 \\ (-1) \cdot 2 + 1 \cdot 0 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{3} \end{bmatrix}$$

$(x_2, y_0) = (\bar{4}, \bar{3})$

diperoleh bahwa entri awal berada pada posisi $(\bar{4}, \bar{3})$

l. Entri baru $(2, 1)$

$$\begin{bmatrix} x_2 \\ y_1 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 2 + (-1) \cdot 1 \\ (-1) \cdot 2 + 1 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{4} \end{bmatrix}$$

$(x_2, y_1) = (\bar{3}, \bar{4})$

diperoleh bahwa entri awal berada pada posisi $(\bar{3}, \bar{4})$

m. Entri baru $(2, 2)$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 2 + (-1) \cdot 2 \\ (-1) \cdot 2 + 1 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{0} \end{bmatrix}$$

$(x_2, y_2) = (\bar{2}, \bar{0})$

diperoleh bahwa entri awal berada pada posisi $(\bar{2}, \bar{0})$

n. Entri baru $(2,3)$

$$\begin{bmatrix} x_2 \\ y_3 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} \pmod 5$$

$$\equiv \begin{bmatrix} 2 \cdot 2 + (-1) \cdot 3 \\ (-1) \cdot 2 + 1 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{1} \end{bmatrix}$$

$$(x_2, y_3) = (\bar{1}, \bar{1})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{1}, \bar{1})$

o. Entri baru $(2,4)$

$$\begin{bmatrix} x_2 \\ y_4 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} \pmod 5$$

$$\equiv \begin{bmatrix} 2 \cdot 2 + (-1) \cdot 4 \\ (-1) \cdot 2 + 1 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{2} \end{bmatrix}$$

$$(x_2, y_4) = (\bar{0}, \bar{2})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{0}, \bar{2})$

p. Entri baru $(3,0)$

$$\begin{bmatrix} x_3 \\ y_0 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} \pmod 5$$

$$\equiv \begin{bmatrix} 2 \cdot 3 + (-1) \cdot 0 \\ (-1) \cdot 3 + 1 \cdot 0 \end{bmatrix} (mod\ 5) = \begin{bmatrix} \bar{1} \\ \bar{2} \end{bmatrix}$$

$$(x_3, y_0) = (\bar{1}, \bar{2})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{1}, \bar{2})$

q. Entri baru $(3,1)$

$$\begin{bmatrix} x_3 \\ y_1 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \pmod 5$$

$$\equiv \begin{bmatrix} 2 \cdot 3 + (-1) \cdot 1 \\ (-1) \cdot 3 + 1 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{3} \end{bmatrix}$$

$$(x_3, y_1) = (\bar{0}, \bar{3})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{0}, \bar{3})$

r. Entri baru $(3, 2)$

$$\begin{bmatrix} x_3 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 3 + (-1) \cdot 2 \\ (-1) \cdot 3 + 1 \cdot 2 \end{bmatrix} \ (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{4} \end{bmatrix}$$

$$(x_3, y_2) = (\bar{4}, \bar{4})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{4}, \bar{4})$

s. Entri baru $(3, 3)$

$$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 3 + (-1) \cdot 3 \\ (-1) \cdot 3 + 1 \cdot 3 \end{bmatrix} \ (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{0} \end{bmatrix}$$

$$(x_3, y_3) = (\bar{3}, \bar{0})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{3}, \bar{0})$

t. Entri baru $(3, 4)$

$$\begin{bmatrix} x_3 \\ y_4 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 3 + (-1) \cdot 4 \\ (-1) \cdot 3 + 1 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{1} \end{bmatrix}$$

$$(x_3, y_4) = (\bar{2}, \bar{1})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{2}, \bar{1})$

u. Entri baru $(4, 0)$

$$\begin{bmatrix} x_4 \\ y_0 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix} \ (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 4 + (-1) \cdot 0 \\ (-1) \cdot 4 + 1 \cdot 0 \end{bmatrix} \ (mod\ 5) \equiv \begin{bmatrix} \bar{3} \\ \bar{1} \end{bmatrix}$$

$$(x_4, y_0) = (\bar{3}, \bar{1})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{3}, \bar{1})$

v. Entri baru $(4, 1)$

$$\begin{bmatrix} x_4 \\ y_1 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} \pmod 5$$

$$\equiv \begin{bmatrix} 2 \cdot 4 + (-1) \cdot 1 \\ (-1) \cdot 4 + 1 \cdot 1 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{2} \\ \bar{2} \end{bmatrix}$$

$$(x_4, y_1) = (\bar{2}, \bar{2})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{2}, \bar{2})$

w. Entri baru $(4, 2)$

$$\begin{bmatrix} x_4 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} \pmod 5$$

$$\equiv \begin{bmatrix} 2 \cdot 4 + (-1) \cdot 2 \\ (-1) \cdot 4 + 1 \cdot 2 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{1} \\ \bar{3} \end{bmatrix}$$

$$(x_4, y_2) = (\bar{1}, \bar{3})$$

diperoleh bahwa entri awal berada pada posisi $(1, 3)$

x. Entri baru $(4, 3)$

$$\begin{bmatrix} x_4 \\ y_3 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 4 + (-1) \cdot 3 \\ (-1) \cdot 4 + 1 \cdot 3 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{0} \\ \bar{4} \end{bmatrix}$$

$$(x_4, y_3) = (\bar{0}, \bar{4})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{0}, \bar{4})$

y. Entri baru $(4, 4)$

$$\begin{bmatrix} x_4 \\ y_4 \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix} (mod\ 5)$$

$$\equiv \begin{bmatrix} 2 \cdot 4 + (-1) \cdot 4 \\ (-1) \cdot 4 + 1 \cdot 4 \end{bmatrix} (mod\ 5) \equiv \begin{bmatrix} \bar{4} \\ \bar{0} \end{bmatrix}$$

$$(x_4, y_4) = (\bar{4}, \bar{0})$$

diperoleh bahwa entri awal berada pada posisi $(\bar{4}, \bar{0})$

Telah didapat entri awal menggunakan invers matriks transformasi *Arnold's Cat Map*, dengan hasil seperti di atas.

4. Didapatkan matriks *ciphertext* asli

   Setelah proses pengembalian menggunakan invers matriks transformasi *Arnold's Cat Map* didapatkan bentuk matriks *ciphertext* awal sebagai berikut:

$$\begin{bmatrix} 72 & 97 & 222 & 85 & 164 \\ 97 & 109 & 97 & 107 & 117 \\ 197 & 252 & 18 & 223 & 67 \\ 2 & 222 & 62 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

5. Mengonversi matriks ke dalam array 1D.

   Berdasarkan hasil konversi, matriks dua dimensi di atas diubah menjadi bentuk array satu dimensi (1D) dengan urutan elemen dari baris pertama hingga baris terakhir, sehingga diperoleh hasil sebagai berikut:

   $[72, 97, 222, 85, 164, 97, 109, 97, 107, 117, 197, 252, 18, 223, 67, 2, 222, 62]$.

   konversi ini dilakukan untuk memudahkan proses pengolahan data pada tahap dekripsi selanjutnya.

6. Melakukan XOR dengan keystream awal.

   Untuk mendapatkan *plaintext* kembali, setelah proses invers matriks transformasi *Arnold's Cat Map* dilakukan XOR kembali menggunakan *keystream* awal, berikut proses XOR menggunakan *keystream* awal:

   karakter pertama $= 01001000 \oplus 00000000 = 01001000\ (H)$

   karakter ke-2 $= 01100001 \oplus 00000000 = 01100001\ (a)$

   karakter ke-3 $= 11011110 \oplus 10110111 = 01101001\ (i)$

   karakter ke-4 $= 10100100 \oplus 01110101 = 00100000\ (space)$

karakter ke-5 $= 01101110 \oplus 11001010 = 01101110\ (n)$

karakter ke-6 $= 01100001 \oplus 00000000 = 01100001\ (a)$

karakter ke-7 $= 01101101 \oplus 00000000 = 01101101\ (m)$

karakter ke-8 $= 01100001 \oplus 00000000 = 01100001\ (a)$

karakter ke-9 $= 01101011 \oplus 00000000 = 01101011\ (k)$

karakter ke-10 $= 01110101 \oplus 00000000 = 01110101\ (u)$

karakter ke-11 $= 11000101 \oplus 11100101 = 00100000\ (space)$

karakter ke-12 $= 11111100 \oplus 10101110 = 01010010\ (R)$

karakter ke-13 $= 00010010 \oplus 01111101 = 01101111\ (o)$

karakter ke-14 $= 11011111 \oplus 10111001 = 01100110\ (f)$

karakter ke-15 $= 01000011 \oplus 00101010 = 01101001\ (i)$

karakter ke-16 $= 00000010 \oplus 01100100 = 01100110\ (f)$

karakter ke-17 $= 01101011 \oplus 10111111 = 01100001\ (a)$

karakter ke-18 $= 00111110 \oplus 01101000 = 01010110\ (h)$

7. Didapatkan *plaintext* awal.

   Didapat kembali *plaintext* awal menjadi **“Hai namaku Rofifah”.**

Selain memastikan kerahasiaan pesan melalui proses enkripsi dan dekripsi, penelitian ini juga berkaitan dengan dua aspek penting lain dalam kriptografi, yaitu integritas dan autentikasi. Integritas pesan terjaga karena mekanisme yang digunakan seperti transformasi *Arnold’s Cat Map* dan operasi XOR sangat sensitif terhadap perubahan isi pesan teks. Apabila terdapat satu bit saja yang dimodifikasi selama proses pengiriman, hasil invers matriks transformasi *Arnold’s Cat Map* tidak akan dapat menyusun kembali *ciphertext* ke bentuk semula, dan hasil XOR tidak akan menghasilkan *plaintext* yang valid. Oleh karena itu, keberhasilan proses

dekripsi dalam mengembalikan pesan “Hai namaku Rofifah” secara utuh menunjukkan bahwa pesan teks tidak mengalami modifikasi, sehingga integritasnya terjamin. Selain itu, penelitian ini juga mendukung konsep autentikasi secara implisit. Hal ini karena hanya pihak yang mengetahui parameter *Logistic Map*, pengaturan transformasi atau pengacakan pada *Arnold’s Cat Map*, serta *keystream* yang dihasilkan yang dapat dilakukan proses dekripsi dengan benar. Jika parameter tersebut tidak sesuai, pesan tidak dapat dikembalikan lagi atau kembali ke *plaintext*. Dengan demikian, proses dekripsi yang berhasil juga berfungsi sebagai bukti bahwa pengirim dan penerima memiliki kunci dan parameter yang sama, sehingga memenuhi konsep autentikasi sebagaimana yang dijelaskan dalam definisi kriptografi.

4.5 Integrasi Nilai Amanah dalam Pengamanan Pesan Teks

Nilai amanah dalam Islam menekankan pentingnya tanggung jawab, kejujuran, dan ketekunan dalam menjalankan setiap tugas yang dipercayakan. Dalam penelitian ini, nilai amanah diintegrasikan ke dalam seluruh proses pengamanan pesan teks menggunakan algoritma *Logistic Map* dan *Arnold’s Cat Map*. Penerapan nilai ini tidak hanya mencakup aspek teknis dalam pengembangan sistem keamanan informasi, tetapi juga aspek etis dan spiritual yang menegaskan komitmen peneliti untuk menjaga kerahasiaan, integritas, dan kepercayaan dalam setiap tahapan penelitian. Dengan demikian, penelitian ini menempatkan amanah sebagai dasar kemaslahatan umat manusia.

Amanah diterapkan sejak tahap perancangan algoritma, di mana proses desain dilakukan dengan penuh tanggung jawab dan kejujuran tanpa manipulasi data atau penyembunyian kekurangan metode. Pemilihan algoritma *Logistic Map*

dan *Arnold's Cat Map* didasarkan pada keunggulannya dalam menghasilkan sistem kemanan yang kuat dan deterministik, sekaligus mencerminkan kesungguhan peneliti untuk menghadirkan solusi kriptografi yang transparan dan dapat dipercaya.

Dalam tahap komputasi, amanah diwujudkan melalui ketekunan dan objektivitas peneliti dalam melaksanakan proses iterasi *Logistic Map*, operasi XOR, dan transformasi *Arnold's Cat Map* secara akurat tanpa rekayasa nilai. Hasil simulasi, termasuk pembangkitan *keystream* dan proses enkripsi-dekripsi, disajikan apa adanya sesuai keluaran program. Sikap jujur ini merupakan bentuk nyata tanggung jawab ilmiah dalam menjaga integritas hasil penelitian dan mencegah penyalahgunaan teknologi untuk tujuan yang tidak etis.

Selanjutnya, amanah juga terlihat pada penyajian hasil penelitian. Seluruh data seperti tabel konversi ASCII, matriks *ciphertext*, dan *output* program Python ditampilkan secara terbuka tanpa penghilangan atau perubahan nilai. Transparansi ini mencerminkan kejujuran dan tanggung jawab peneliti dalam menghormati prinsip kerahasiaan data dan privasi digital. Dalam konteks sosial, pengamanan pesan teks yang dilakukan melalui algoritma ini merupakan wujud nyata penerapan nilai amanah, karena teknologi yang dikembangkan berperan dalam melindungi informasi pribadi dari potensi penyalahgunaan atau ancaman eksternal.

Secara keseluruhan, integrasi nilai amanah dalam penelitian ini memperkaya aspek ilmiah dan spiritual dari hasil yang diperoleh. Penelitian tidak hanya menghasilkan pendekatan kriptografi yang efisien secara matematis, tetapi juga mencerminkan penerapan etika Islam dalam teknologi. Nilai amanah menjadi landasan moral bagi peneliti untuk bekerja dengan jujur, bertanggung jawab, dan

profesional, sehingga hasil penelitian ini tidak hanya berkontribusi pada pengembangan ilmu pengetahuan, tetapi juga memperkuat nilai keimanan dan etika dalam dunia sains.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan uraian rumusan masalah, hasil penelitian dan pembahasan yang telah dilakukan mengenai pengamanan pesan teks menggunakan algoritma *Logistic Map* dan *Arnold's Cat Map*, maka dapat disimpulkan hal-hal sebagai berikut:

1. Proses pembangkitan kunci (*keystream*) menggunakan algoritma *Logistic Map* berhasil dilakukan dengan menentukan parameter awal (nilai x_0 dan konstanta r) sebagai kunci rahasia. Nilai-nilai hasil iterasi *Logistic Map* kemudian digunakan sebagai pembangkitan bilangan acak yang berperan sebagai *keystream* dalam proses enkripsi. Proses ini mampu menghasilkan urutan nilai yang acak, sensitif terhadap perubahan nilai awal, serta tidak mudah diprediksi, sehingga meningkatkan keamanan sistem pengamanan pesan teks.
2. Mekanisme enkripsi pesan teks dilakukan dengan memanfaatkan *keystream* yang dihasilkan dari algoritma *Logistic Map* dan dikombinasikan dengan proses pengacakan menggunakan algoritma *Arnold's Cat Map*. Setiap karakter pada pesan teks diubah ke dalam bentuk ASCII, kemudian dienkripsi dengan operasi XOR menggunakan *keystream* yang sudah dipermutasi baris dan kolom yang dihasilkan. Selanjutnya, hasil enkripsi diacak posisinya dengan *Arnold's Cat Map* sehingga diperoleh *ciphertext* yang memiliki pola acak dan sulit dianalisis. Kombinasi kedua algoritma ini

memperkuat keamanan data karena menghasilkan *ciphertext* yang berbeda meskipun pesan asli hanya mengalami sedikit perubahan.

3. Proses dekripsi pesan teks dilakukan dengan langkah yang berlawanan dari proses enkripsi, yaitu mengembalikan hasil pengacakan *Arnold's Cat Map* ke posisi semula, kemudian melakukan operasi XOR dengan *keystream* yang sama untuk mendapatkan kembali pesan asli. Hasil pengujian menunjukkan bahwa proses dekripsi berhasil mengembalikan pesan teks secara akurat 100%, tanpa perubahan karakter maupun kehilangan data.

Dengan demikian, dapat disimpulkan bahwa penerapan algoritma *Logistic Map* dan *Arnold's Cat Map* dalam sistem pengamanan pesan teks mampu bekerja secara efektif, menghasilkan *keystream* yang acak, proses enkripsi yang kuat, dan dekripsi yang akurat. Sistem ini berpotensi digunakan sebagai metode alternatif dalam pengamanan informasi digital yang membutuhkan tingkat keamanan tinggi.

5.2 Saran

Pada penelitian ini, penulis fokus pada pengamanan pesan teks menggunakan algoritma *Logistic Map* dan *Arnlod's Cat Map* sebagai metode enkripsi dan dekripsi untuk menjaga kerahasiaan informasi. Hasil penelitian menunjukkan bahwa kombinasi kedua algoritma tersebut efektif dalam menghasilkan *keystream* yang acak, proses enkripsi yang kuat, serta dekripsi yang kuat. Untuk penelitian selanjutnya, diharapkan pengembangan lebih lanjut dapat mencakup pengamanan data dengan format yang lebih kompleks, seperti citra, audio (.mp3), maupun video (.mp4), agar metode ini dapat diterapkan secara luas pada berbagai jenis data digital. Selain itu, penelitian dapat dikembangkan dengan mengoptimalkan kinerja program, baik dari sisi efisiensi waktu proses maupun

penggunaan memori, sehingga metode ini dapat diimplementasikan dalam sistem komunikasi *real-time*. Penelitian berikutnya juga dapat mempertimbangkan penggabungan algoritma *chaos* lain atau integrasi dengan metode kriptografi modern guna meningkatkan tingkat keamanan dan ketahan terhadap serangan kriptanalisis.

DAFTAR PUSTAKA

Alfred, J. M. dkk. (2014). Applied Cryptography. In *Icassp* (Vol. 21, Issue 3).

Alkhonaini, M. A., Gemeay, E., Zeki Mahmood, F. M., Ayari, M., Alenizi, F. A., & Lee, S. (2024). A new encryption algorithm for image data based on two-way chaotic maps and iterative cellular automata. *Scientific Reports*, *14*(1), 1–15. https://doi.org/10.1038/s41598-024-64741-x

Arif, J., Khan, M. A., Ghaleb, B., Ahmad, J., Munir, A., Rashid, U., & Al-Dubai, A. Y. (2022). A Novel Chaotic Permutation-Substitution Image Encryption Scheme Based on Logistic Map and Random Substitution. *IEEE Access*, *10*, 12966–12982. https://doi.org/10.1109/ACCESS.2022.3146792

Arnol'd, V. I. (1968). Ergodic problems of classical mechanics. *ZAMM - Zeitschrift Für Angewandte Mathematik Und Mechanik*, *50*(7–9), 1–4. http://doi.wiley.com/10.1002/zamm.19700500721

Artuğer, F. (2025). Innovative image encryption approach based on bitwise XOR high nonlinear S-boxes and random permutation. *Soft Computing*, *29*(6), 2891–2903. https://doi.org/10.1007/s00500-025-10547-6

Ath-Thabari, I. J. (2007). *Tafsir Ath-Thabari*. https://archive.org/details/tafsir-1_202201/Tafsir 18/mode/2up

Aumasson, J.-P. (2017). Serious Cryptography A Practical Introduction to Modern Encryption. In *No Starch Press*. http://www.mypetskunk.com/uploads/1/0/6/1/106105481/seriouscrytography_ebook.pdf

Dachlan 2014:1. (2014). Arnold's Cat Map: An Exposition Geneva. *Angewandte Chemie International Edition, 6(11), 951–952.*, 22–31.

Dalimunthe, R. P. (2018). Amanah Dalam Perspektif Hadis. *Diroyah : Jurnal Studi Ilmu Hadis*, *1*(1), 7–16. https://doi.org/10.15575/diroyah.v1i1.2050

Ghoffar, M. A., & Abdurrahim Mu'thi. (2003). Tafsir Ibnu Katsir Jilid 5 (Terjemahan). *Tafsir*, 367.

Hamka. (2003a). Jilid 8 Mengandungi Surat-surat. In *Tafsir al-Azhar*.

Hamka, prof. D. (2003b). Tafsir al-azhar (jilid 6). *Pustaka Nasional Pte Ltd Singapura*. https://www.academia.edu/download/70191859/Tafsir_Al_Azhar_04.pdf

Ignatius, D. R., & Setiadi, M. (2023). *An Image Encryption Scheme Combining 2D Cascaded Logistic Map and Permutation-Substitution Operations*.

Irawan, C., & Rachmawanto, E. H. (2022). Implementasi Kriptografi dengan Menggunakan Algoritma Arnold's Cat Map dan Henon Map. *Jurnal Masyarakat Informatika*, *13*(1), 15–32. https://doi.org/10.14710/jmasif.13.1.43312

Irawan, W. H. (2013). *Pengantar Teori Bilangan*. Uin Maliki Press.

Jin, B., Fan, L., Zhang, B., Lei, R., & Liu, L. (2024). Image encryption hiding algorithm based on digital time-varying delay chaos model and compression sensing technique. *IScience*, *27*(9), 110717. https://doi.org/10.1016/j.isci.2024.110717

Katz, Jonathan and Lindell, Y. (2015). *Introduction to Modern Cryptography* (Second Edi). Taylor & Francis Group. chrome - extension://efaidnbmnnnibpcajpcglclefindmkaj/file:///E:/Download/[Jonathan_Katz,_Yehuda_Lindell]_Introduction_to_Mo(2nd).pdf

Kemenag, Q. (2022a). *Qur'an Kemenag*. 6–10.

Kemenag, Q. (2022b). *Qur'an Kemenag*. 1–7.

Lawnik, M., Moysis, L., & Volos, C. (2022). Chaos-Based Cryptography: Text Encryption Using Image Algorithms. *Electronics (Switzerland)*, *11*(19), 1–13. https://doi.org/10.3390/electronics11193156

Lesnussa, Y. A., & Tomasouw, B. P. (2024). *Pengamanan Pesan Teks Pada Citra Digital Menggunakan Kombinasi Algoritma RSA-Quasigroup Cipher dan Metode LSB Pola Zig-zag*. *2024*(Senada), 1090–1099.

Min, L., Ting, L., & Yu-jie, H. (2013). Arnold Transform Based Image Scrambling Method. *Proceedings of 3rd International Conference on Multimedia Technology(ICMT-13)*, *84*. https://doi.org/10.2991/icmt-13.2013.160

Mukherjee, S. (2023). New Insights Into Chaos Based Image Encryption & Its Application. *Journal of Mathematical Sciences & Computational Mathematics*, *4*(2), 241–264. https://doi.org/10.15864/jmscm.4208

Munir, R. (2008). Pengantar Ilmu Kriptografi. *Penerbit Andi*, 1–16. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Pengantar+Kriptografi#0

Munir, R. (2021). *01 - Pengantar Kriptografi*. 1–57.

Ott, E. (2002). *Chaos Dynamical Systems* (Vol. 17).

Ramakrishna, D., & Ali Shaik, M. (2025). A Comprehensive Analysis of Cryptographic Algorithms: Evaluating Security, Efficiency, and Future Challenges. *IEEE Access*, *13*(December 2024), 11576–11593. https://doi.org/10.1109/ACCESS.2024.3518533

Reymond, R., Johanes Manullang, Jhoische Tamba, Farel Parasian Sitohang, & Eikel Nioisha Ginting. (2024). Cryptography With One-Time Pad (OTP) Algorithm Xor Based. *Jurnal Teknik Indonesia*, *3*(02), 54–60. https://doi.org/10.58471/ju-ti.v3i02.664

Riaz, M., Dilpazir, H., Naseer, S., Mahmood, H., Anwar, A., Khan, J., Benitez, I. B., & Ahmad, T. (2024). Secure and Fast Image Encryption Algorithm Based on Modified Logistic Map. *Information (Switzerland)*, *15*(3), 1–20. https://doi.org/10.3390/info15030172

Sanjaya W.S, M. (2025). *Chaotic Circuits*. Bolabot.https://books.google.co.id/books?hl=id&lr=&id=88_EQAAQBAJ&oi=fnd&pg=PP1&dq=chaos+definisi&ots=4l0BNuZDMU&sig=eBOSXZGe5uHLjLmF0qWC_fa9Lqg&redir_esc=y#v=onepage&q=chaos definisi&f=false

Saputro, P. H. (2023). Implementasi Algoritma Exclusive OR (XOR) dalam Pengembangan Aplikasi Chat Berbasis Android. *Accident Analysis and Prevention*, *183*(2), 153–164.

Shihab, M. Q. (2016). Tafsir Ai-Misbah Jilid 9. In *Jakarta : Lentera Hati* (Vol. 4, Issue 1).

Taipale, S., & Farinosi, M. (2018). The big meaning of small messages: The use of WhatsApp in intergenerational family communication. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 10926 LNCS*. Springer International Publishing. https://doi.org/10.1007/978-3-319-92034-4_40

Tazi, F., Nandakumar, A., Dykstra, J., Rajivan, P., & Das, S. (2024). SoK: Analyzing Privacy and Security of Healthcare Data from the User Perspective. *ACM Transactions on Computing for Healthcare*, *5*(2). https://doi.org/10.1145/3650116

Zhang, B., & Liu, L. (2023). *Mathematics-11-02585-V2.Pdf*.

LAMPIRAN

Lampiran 1. Hasil Iterasi Ke-21 Hingga Terakhir

```
Iterasi 21: x = 0.807164, bit keystream = 1
Iterasi 22: x = 0.607036, bit keystream = 1
Iterasi 23: x = 0.930319, bit keystream = 1
Iterasi 24: x = 0.252821, bit keystream = 0
Iterasi 25: x = 0.736720, bit keystream = 1
Iterasi 26: x = 0.756458, bit keystream = 1
Iterasi 27: x = 0.718494, bit keystream = 1
Iterasi 28: x = 0.788815, bit keystream = 1
Iterasi 29: x = 0.649684, bit keystream = 1
Iterasi 30: x = 0.887619, bit keystream = 1
Iterasi 31: x = 0.389030, bit keystream = 0
Iterasi 32: x = 0.926974, bit keystream = 1
Iterasi 33: x = 0.264003, bit keystream = 0
Iterasi 34: x = 0.757791, bit keystream = 1
Iterasi 35: x = 0.715820, bit keystream = 1
Iterasi 36: x = 0.793345, bit keystream = 1
Iterasi 37: x = 0.639400, bit keystream = 1
Iterasi 38: x = 0.899214, bit keystream = 1
Iterasi 39: x = 0.353451, bit keystream = 0
Iterasi 40: x = 0.891241, bit keystream = 1
Iterasi 41: x = 0.378028, bit keystream = 0
Iterasi 42: x = 0.916979, bit keystream = 1
Iterasi 43: x = 0.296901, bit keystream = 0
Iterasi 44: x = 0.814129, bit keystream = 1
Iterasi 45: x = 0.590161, bit keystream = 1
Iterasi 46: x = 0.943297, bit keystream = 1
Iterasi 47: x = 0.208602, bit keystream = 0
Iterasi 48: x = 0.643840, bit keystream = 1
Iterasi 49: x = 0.894309, bit keystream = 1
Iterasi 50: x = 0.368628, bit keystream = 0
```

```
Iterasi 51: x = 0.907692, bit keystream = 1
Iterasi 52: x = 0.326771, bit keystream = 0
Iterasi 53: x = 0.857968, bit keystream = 1
Iterasi 54: x = 0.475251, bit keystream = 0
Iterasi 55: x = 0.972611, bit keystream = 1
Iterasi 56: x = 0.103891, bit keystream = 0
Iterasi 57: x = 0.363081, bit keystream = 0
Iterasi 58: x = 0.901887, bit keystream = 1
Iterasi 59: x = 0.345098, bit keystream = 0
Iterasi 60: x = 0.881421, bit keystream = 1
Iterasi 61: x = 0.407620, bit keystream = 0
Iterasi 62: x = 0.941717, bit keystream = 1
Iterasi 63: x = 0.214054, bit keystream = 0
Iterasi 64: x = 0.656117, bit keystream = 1
Iterasi 65: x = 0.879947, bit keystream = 1
Iterasi 66: x = 0.411997, bit keystream = 0
Iterasi 67: x = 0.944796, bit keystream = 1
Iterasi 68: x = 0.203410, bit keystream = 0
Iterasi 69: x = 0.631934, bit keystream = 1
Iterasi 70: x = 0.907114, bit keystream = 1
Iterasi 71: x = 0.328607, bit keystream = 0
Iterasi 72: x = 0.860435, bit keystream = 1
Iterasi 73: x = 0.468337, bit keystream = 0
Iterasi 74: x = 0.971090, bit keystream = 1
Iterasi 75: x = 0.109489, bit keystream = 0
Iterasi 76: x = 0.380254, bit keystream = 0
Iterasi 77: x = 0.919078, bit keystream = 1
Iterasi 78: x = 0.290058, bit keystream = 0
Iterasi 79: x = 0.803105, bit keystream = 1
Iterasi 80: x = 0.616696, bit keystream = 1
```

```
Iterasi 81: x = 0.921890, bit keystream = 1
Iterasi 82: x = 0.280835, bit keystream = 0
Iterasi 83: x = 0.787670, bit keystream = 1
Iterasi 84: x = 0.652258, bit keystream = 1
Iterasi 85: x = 0.884588, bit keystream = 1
Iterasi 86: x = 0.398159, bit keystream = 0
Iterasi 87: x = 0.934551, bit keystream = 1
Iterasi 88: x = 0.238545, bit keystream = 0
Iterasi 89: x = 0.708401, bit keystream = 1
Iterasi 90: x = 0.805619, bit keystream = 1
Iterasi 91: x = 0.610728, bit keystream = 1
Iterasi 92: x = 0.927184, bit keystream = 1
Iterasi 93: x = 0.263305, bit keystream = 0
Iterasi 94: x = 0.756505, bit keystream = 1
Iterasi 95: x = 0.718401, bit keystream = 1
Iterasi 96: x = 0.788974, bit keystream = 1
Iterasi 97: x = 0.649326, bit keystream = 1
Iterasi 98: x = 0.888037, bit keystream = 1
Iterasi 99: x = 0.387766, bit keystream = 0
Iterasi 100: x = 0.925874, bit keystream = 1
Iterasi 101: x = 0.267664, bit keystream = 0
Iterasi 102: x = 0.764477, bit keystream = 1
Iterasi 103: x = 0.702202, bit keystream = 1
Iterasi 104: x = 0.815546, bit keystream = 1
Iterasi 105: x = 0.586680, bit keystream = 1
Iterasi 106: x = 0.945697, bit keystream = 1
Iterasi 107: x = 0.200280, bit keystream = 0
Iterasi 108: x = 0.624655, bit keystream = 1
Iterasi 109: x = 0.914398, bit keystream = 1
Iterasi 110: x = 0.305268, bit keystream = 0
```

```
Iterasi 111: x = 0.827110, bit keystream = 1
Iterasi 112: x = 0.557696, bit keystream = 1
Iterasi 113: x = 0.962018, bit keystream = 1
Iterasi 114: x = 0.142505, bit keystream = 0
Iterasi 115: x = 0.476568, bit keystream = 0
Iterasi 116: x = 0.972859, bit keystream = 1
Iterasi 117: x = 0.102978, bit keystream = 0
Iterasi 118: x = 0.360257, bit keystream = 0
Iterasi 119: x = 0.898840, bit keystream = 1
Iterasi 120: x = 0.354613, bit keystream = 0
Iterasi 121: x = 0.892564, bit keystream = 1
Iterasi 122: x = 0.373984, bit keystream = 0
Iterasi 123: x = 0.913068, bit keystream = 1
Iterasi 124: x = 0.309561, bit keystream = 0
Iterasi 125: x = 0.833558, bit keystream = 1
Iterasi 126: x = 0.541081, bit keystream = 1
Iterasi 127: x = 0.968418, bit keystream = 1
Iterasi 128: x = 0.119280, bit keystream = 0
Iterasi 129: x = 0.409702, bit keystream = 0
Iterasi 130: x = 0.943201, bit keystream = 1
Iterasi 131: x = 0.208935, bit keystream = 0
Iterasi 132: x = 0.644596, bit keystream = 1
Iterasi 133: x = 0.893459, bit keystream = 1
Iterasi 134: x = 0.371243, bit keystream = 0
Iterasi 135: x = 0.910344, bit keystream = 1
Iterasi 136: x = 0.318310, bit keystream = 0
Iterasi 137: x = 0.846255, bit keystream = 1
Iterasi 138: x = 0.507418, bit keystream = 1
Iterasi 139: x = 0.974785, bit keystream = 1
Iterasi 140: x = 0.095857, bit keystream = 0
```

```
Iterasi 141: x = 0.338008, bit keystream = 0
Iterasi 142: x = 0.872659, bit keystream = 1
Iterasi 143: x = 0.433389, bit keystream = 0
Iterasi 144: x = 0.957696, bit keystream = 1
Iterasi 145: x = 0.158007, bit keystream = 0
Iterasi 146: x = 0.518859, bit keystream = 1
Iterasi 147: x = 0.973613, bit keystream = 1
Iterasi 148: x = 0.100194, bit keystream = 0
Iterasi 149: x = 0.351606, bit keystream = 0
Iterasi 150: x = 0.889119, bit keystream = 1
Iterasi 151: x = 0.384488, bit keystream = 0
Iterasi 152: x = 0.922962, bit keystream = 1
Iterasi 153: x = 0.277301, bit keystream = 0
Iterasi 154: x = 0.781580, bit keystream = 1
Iterasi 155: x = 0.665780, bit keystream = 1
Iterasi 156: x = 0.867816, bit keystream = 1
Iterasi 157: x = 0.447374, bit keystream = 0
Iterasi 158: x = 0.964199, bit keystream = 1
Iterasi 159: x = 0.134626, bit keystream = 0
Iterasi 160: x = 0.454356, bit keystream = 0
Iterasi 161: x = 0.966875, bit keystream = 1
Iterasi 162: x = 0.124909, bit keystream = 0
Iterasi 163: x = 0.426296, bit keystream = 0
Iterasi 164: x = 0.953814, bit keystream = 1
Iterasi 165: x = 0.171806, bit keystream = 0
Iterasi 166: x = 0.554926, bit keystream = 1
Iterasi 167: x = 0.963234, bit keystream = 1
Iterasi 168: x = 0.138115, bit keystream = 0
Iterasi 169: x = 0.464254, bit keystream = 0
Iterasi 170: x = 0.970017, bit keystream = 1
```

```
Iterasi 171: x = 0.113429, bit keystream = 0
Iterasi 172: x = 0.392196, bit keystream = 0
Iterasi 173: x = 0.929676, bit keystream = 1
Iterasi 174: x = 0.254978, bit keystream = 0
Iterasi 175: x = 0.740860, bit keystream = 1
Iterasi 176: x = 0.748747, bit keystream = 1
Iterasi 177: x = 0.733687, bit keystream = 1
Iterasi 178: x = 0.762022, bit keystream = 1
Iterasi 179: x = 0.707243, bit keystream = 1
Iterasi 180: x = 0.807496, bit keystream = 1
Iterasi 181: x = 0.606240, bit keystream = 1
Iterasi 182: x = 0.930981, bit keystream = 1
Iterasi 183: x = 0.250596, bit keystream = 0
Iterasi 184: x = 0.732412, bit keystream = 1
Iterasi 185: x = 0.764341, bit keystream = 1
Iterasi 186: x = 0.702483, bit keystream = 1
Iterasi 187: x = 0.815102, bit keystream = 1
Iterasi 188: x = 0.587772, bit keystream = 1
Iterasi 189: x = 0.944954, bit keystream = 1
Iterasi 190: x = 0.202861, bit keystream = 0
Iterasi 191: x = 0.630663, bit keystream = 1
Iterasi 192: x = 0.908416, bit keystream = 1
Iterasi 193: x = 0.324464, bit keystream = 0
Iterasi 194: x = 0.854830, bit keystream = 1
Iterasi 195: x = 0.483972, bit keystream = 0
Iterasi 196: x = 0.973998, bit keystream = 1
Iterasi 197: x = 0.098770, bit keystream = 0
Iterasi 198: x = 0.347158, bit keystream = 0
Iterasi 199: x = 0.883893, bit keystream = 1
Iterasi 200: x = 0.400241, bit keystream = 0
```

```
Iterasi 201: x = 0.936188, bit keystream = 1
Iterasi 202: x = 0.232987, bit keystream = 0
Iterasi 203: x = 0.696946, bit keystream = 1
Iterasi 204: x = 0.823727, bit keystream = 1
Iterasi 205: x = 0.566283, bit keystream = 1
Iterasi 206: x = 0.957866, bit keystream = 1
Iterasi 207: x = 0.157400, bit keystream = 0
Iterasi 208: x = 0.517237, bit keystream = 1
Iterasi 209: x = 0.973841, bit keystream = 1
Iterasi 210: x = 0.099351, bit keystream = 0
Iterasi 211: x = 0.348972, bit keystream = 0
Iterasi 212: x = 0.886043, bit keystream = 1
Iterasi 213: x = 0.393785, bit keystream = 0
Iterasi 214: x = 0.931002, bit keystream = 1
Iterasi 215: x = 0.250526, bit keystream = 0
Iterasi 216: x = 0.732274, bit keystream = 1
Iterasi 217: x = 0.764590, bit keystream = 1
Iterasi 218: x = 0.701969, bit keystream = 1
Iterasi 219: x = 0.815913, bit keystream = 1
Iterasi 220: x = 0.585776, bit keystream = 1
Iterasi 221: x = 0.946306, bit keystream = 1
Iterasi 222: x = 0.198164, bit keystream = 0
Iterasi 223: x = 0.619691, bit keystream = 1
Iterasi 224: x = 0.919129, bit keystream = 1
Iterasi 225: x = 0.289891, bit keystream = 0
Iterasi 226: x = 0.802832, bit keystream = 1
Iterasi 227: x = 0.617342, bit keystream = 1
Iterasi 228: x = 0.921300, bit keystream = 1
Iterasi 229: x = 0.282774, bit keystream = 0
Iterasi 230: x = 0.790970, bit keystream = 1
```

```
Iterasi 231: x = 0.644812, bit keystream = 1
Iterasi 232: x = 0.893214, bit keystream = 1
Iterasi 233: x = 0.371991, bit keystream = 0
Iterasi 234: x = 0.911094, bit keystream = 1
Iterasi 235: x = 0.315908, bit keystream = 0
Iterasi 236: x = 0.842829, bit keystream = 1
Iterasi 237: x = 0.516627, bit keystream = 1
Iterasi 238: x = 0.973922, bit keystream = 1
Iterasi 239: x = 0.099052, bit keystream = 0
Iterasi 240: x = 0.348040, bit keystream = 0
Iterasi 241: x = 0.884942, bit keystream = 1
Iterasi 242: x = 0.397097, bit keystream = 0
Iterasi 243: x = 0.933703, bit keystream = 1
Iterasi 244: x = 0.241418, bit keystream = 0
```

Lampiran 2. Tabel *Output* Kode Python

Iterasi	x_n	Bit *Keystream*
111	0,827110	1
112	0,557696	1
113	0,962018	1
114	0,142505	0
115	0,476568	0
116	0,972859	1
117	0,102978	0
118	0,360257	0
119	0,898840	1
120	0,354613	0
121	0,892564	1
122	0,373984	0
123	0,913068	1
124	0,309561	0
125	0,833558	1
126	0,541081	1
127	0,968418	1
128	0,119280	0
129	0,409702	0
130	0,943201	1
131	0,208935	0
132	0,644596	1
133	0,893459	1
134	0,371243	0
135	0,910344	1
136	0,318310	0
137	0,846255	1
138	0,507418	1
139	0,974785	1
140	0,095857	0
141	0,338008	0
142	0,872659	1
143	0,433389	0
144	0,957696	1
145	0,158007	0
146	0,518859	1
147	0,973613	1
148	0,100194	0

Iterasi	x_n	Bit *Keystream*
149	0,351606	0
150	0,889119	1
151	0,384488	0
152	0,922962	1
153	0,277301	0
154	0,781580	1
155	0,665780	1
156	0,867816	1
157	0,447374	0
158	0,964199	1
159	0,134626	0
160	0,454356	0
161	0,966875	1
162	0,124909	0
163	0,426296	0
164	0,953814	1
165	0,171806	0
166	0,171806	1
167	0,963234	1
168	0,138115	0
169	0,464254	0
170	0,970017	1
171	0,113429	0
172	0,392196	0
173	0,929676	1
174	0,254978	0
175	0,740860	1
176	0,748747	1
177	0,733687	1
178	0,762022	1
179	0,707243	1
180	0,807496	1
181	0,606240	1
182	0,930981	1
183	0,930981	0
184	0,732412	1
185	0,764341	1
186	0,702483	1
187	0,815102	1
188	0,587772	1

Iterasi	x_n	Bit *Keystream*
189	0,944954	1
190	0,202861	0
191	0,630663	1
192	0,908416	1
193	0,324464	0
194	0,854830	1
195	0,483972	0
196	0,973998	1
197	0,098770	0
198	0,347158	0
199	0,883893	1
200	0,400241	0
201	0,936188	1
202	0,232987	0
203	0,696946	1
204	0,823727	1
205	0,566283	1
206	0,957866	1
207	0,157400	0
208	0,517237	1
209	0,973841	1
210	0,099351	0
211	0,348972	0
212	0,886043	1
213	0,393785	0
214	0,931002	1
215	0,250526	0
216	0,732274	1
217	0,764590	1
218	0,701969	1
219	0,815913	1
220	0,585776	1
221	0,946306	1
222	0,198164	0
223	0,619691	1
224	0,919129	1
225	0,289891	0
226	0,802832	1
227	0,617342	1
228	0,921300	1

Iterasi	x_n	Bit *Keystream*
229	0,282774	0
230	0,790970	1
231	0,644812	1
232	0,893214	1
233	0,371991	0
234	0,911094	1
235	0,315908	0
236	0,842829	1
237	0,516627	1
238	0,973922	1
239	0,099052	0
240	0,348040	0
241	0,884942	1
242	0,397097	0
243	0,933703	1
244	0,241418	0

Lampiran 3. Tabel ASCII

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
0	0	0	0	NULL	Null character
1	1	1	1	SOH	Start of Header
2	10	2	2	STX	Start of Text
3	11	3	3	ETX	End of Text, hearts card suit
4	100	4	4	EOT	End of Transmission, diamonds card suit
5	101	5	5	ENQ	Enquiry, clubs card suit
6	110	6	6	ACK	Acknowledgement, spade card suit
7	111	7	7	BEL	Bell
8	1000	10	8	BS	Backspace
9	1001	11	9	HT	Horizontal Tab
10	1010	12	a	LF	Line feed

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
11	1011	13	b	VT	Vertical Tab, male symbol, symbol for Mars
12	1100	14	c	FF	Form feed, female symbol, symbol for Venus
13	1101	15	d	CR	Carriage return
14	1110	16	e	SO	Shift Out
15	1111	17	f	SI	Shift In
16	10000	20	10	DLE	Data link escape
17	10001	21	11	DC1	Device control 1
18	10010	22	12	DC2	Device control 2
19	10011	23	13	DC3	Device control 3
20	10100	24	14	DC4	Device control 4
21	10101	25	15	NAK	NAK Negative-acknowledge
22	10110	26	16	SYN	Synchronous idle
23	10111	27	17	ETB	End of trans. block
24	11000	30	18	CAN	Cancel
29	11101	35	1d	GS	Group separator
30	11110	36	1e	RS	Record separator
31	11111	37	1f	US	Unit separator
127	1111111	177	7f	DEL	Delete
128	10000000	200	80	Ç	Majuscule C-cedilla
129	10000001	201	81	ü	letter u with umlaut or diaeresis, u-umlaut
130	10000010	202	82	é	letter e with acute accent or e-acute
131	10000011	203	83	â	letter a with circumflex accent or a-circumflex
132	10000100	204	84	ä	letter a with umlaut or diaeresis, a-umlaut
133	10000101	205	85	à	letter a with grave accent
134	10000110	206	86	å	letter a with a ring
135	10000111	207	87	ç	Minuscule c-cedilla

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
136	10001000	210	88	ê	letter e with circumflex accent or e-circumflex
137	10001001	211	89	ë	letter e with umlaut or diaeresis ; e-umlauts
138	10001010	212	8a	è	letter e with grave accent
139	10001011	213	8b	ï	letter i with umlaut or diaeresis ; i-umlaut
140	10001100	214	8c	î	letter i with circumflex accent or i-circumflex
141	10001101	215	8d	ì	letter i with grave accent
142	10001110	216	8e	Ä	letter A with umlaut or diaeresis ; A-umlaut
143	10001111	217	8f	Å	Capital letter A with a ring
144	10010000	220	90	É	Capital letter E with acute accent or E-acute
145	10010001	221	91	æ	Latin diphthong ae in lowercase
146	10010010	222	92	Æ	Latin diphthong AE in uppercase
147	10010011	223	93	ô	letter o with circumflex accent or o-circumflex
148	10010100	224	94	ö	letter o with umlaut or diaeresis ; o-umlaut
149	10010101	225	95	ò	letter o with grave accent
150	10010110	226	96	û	letter u with circumflex accent or u-circumflex
151	10010111	227	97	ù	letter u with grave accent
152	10011000	230	98	ÿ	Lowercase letter y with diaeresis
153	10011001	231	99	Ö	Letter O with umlaut or diaeresis ; O-umlaut

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
154	10011010	232	9a	Ü	Letter U with umlaut or diaeresis ; U-umlaut
155	10011011	233	9b	ø	Lowercase slashed zero or empty set
156	10011100	234	9c	£	Pound sign ; symbol for the pound sterling
157	10011101	235	9d	Ø	Uppercase slashed zero or empty set
158	10011110	236	9e	×	Multiplication sign
159	10011111	237	9f	*ƒ*	Function sign ; f with hook sign ; florin sign
160	10100000	240	a0	á	Lowercase letter a with acute accent or a-acute
161	10100001	241	a1	í	Lowercase letter i with acute accent or i-acute
162	10100010	242	a2	ó	Lowercase letter o with acute accent or o-acute
163	10100011	243	a3	ú	Lowercase letter u with acute accent or u-acute
164	10100100	244	a4	ñ	eñe, enie, spanish letter enye, lowercase n with tilde
165	10100101	245	a5	Ñ	Spanish letter enye, uppercase N with tilde, EÑE, enie
166	10100110	246	a6	ª	feminine ordinal indicator
167	10100111	247	a7	º	masculine ordinal indicator
168	10101000	250	a8	¿	Inverted question marks
169	10101001	251	a9	®	Registered trademark symbol
170	10101010	252	aa	¬	Logical negation symbol
171	10101011	253	ab	½	One half
172	10101100	254	ac	¼	Quarter, one fourth

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
173	10101101	255	ad	¡	Inverted exclamation marks
174	10101110	256	ae	«	Angle quotes, guillemets, right-pointing quotation mark
175	10101111	257	af	»	Guillemets, angle quotes, left-pointing quotation marks
176	10110000	260	b0	░	Graphic character, low density dotted
177	10110001	261	b1	▒	Graphic character, medium density dotted
178	10110010	262	b2	▓	Graphic character, high density dotted
179	10110011	263	b3	│	Box drawing character single vertical line
180	10110100	264	b4	┤	Box drawing character single vertical and left line
181	10110101	265	b5	Á	Capital letter A with acute accent or A-acute
182	10110110	266	b6	Â	Letter A with circumflex accent or A-circumflex
183	10110111	267	b7	À	Letter A with grave accent
184	10111000	270	b8	©	Copyright symbol
185	10111001	271	b9	╣	Box drawing character double line vertical and left
186	10111010	272	ba	║	Box drawing character double vertical line
187	10111011	273	bb	╗	Box drawing character double line upper right corner
188	10111100	274	bc	╝	Box drawing character double line lower right corner
189	10111101	275	bd	¢	Cent symbol
190	10111110	276	be	¥	YEN and YUAN sign

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
191	10111111	277	bf	┐	Box drawing character single line upper right corner
192	11000000	300	c0	└	Box drawing character single line lower left corner
193	11000001	301	c1	┴	Box drawing character single line horizontal and up
194	11000010	302	c2	┬	Box drawing character single line horizontal down
195	11000011	303	c3	├	Box drawing character single line vertical and right
196	11000100	304	c4	─	Box drawing character single horizontal line
197	11000101	305	c5	┼	Box drawing character single line horizontal vertical
198	11000110	306	c6	ã	Lowercase letter a with tilde or a-tilde
199	11000111	307	c7	Ã	Capital letter A with tilde or A-tilde
200	11001000	310	c8	╚	Box drawing character double line lower left corner
201	11001001	311	c9	╔	Box drawing character double line upper left corner
202	11001010	312	ca	╩	Box drawing character double line horizontal and up
203	11001011	313	cb	╦	Box drawing character double line horizontal down
204	11001100	314	cc	╠	Box drawing character double line vertical and right
205	11001101	315	cd	═	Box drawing character double horizontal line
206	11001110	316	ce	╬	Box drawing character double line horizontal vertical

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
207	11001111	317	cf	¤	Generic currency sign
208	11010000	320	d0	ð	Lowercase letter eth
209	11010001	321	d1	Ð	Capital letter Eth
210	11010010	322	d2	Ê	Letter E with circumflex accent or E-circumflex
211	11010011	323	d3	Ë	Letter E with umlaut or diaeresis, E-umlaut
212	11010100	324	d4	È	Capital letter E with grave accent
213	11010101	325	d5	ı	Lowercase dot less i
214	11010110	326	d6	Í	Capital letter I with acute accent or I-acute
215	11010111	327	d7	Î	Letter I with circumflex accent or I-circumflex
216	11011000	330	d8	Ï	Letter I with umlaut or diaeresis ; I-umlaut
217	11011001	331	d9	┘	Box drawing character single line lower right corner
218	11011010	332	da	┌	Box drawing character single line upper left corner
219	11011011	333	db	▉	Block, graphic character
220	11011100	334	dc	▄	Bottom half block
221	11011101	335	dd	¦	Vertical broken bar
222	11011110	336	de	Ì	Capital letter I with grave accent
223	11011111	337	df	▀	Top half block
224	11100000	340	e0	Ó	Capital letter O with acute accent or O-acute
225	11100001	341	e1	ß	Letter Eszett ; scharfes S or sharp S
226	11100010	342	e2	Ô	Letter O with circumflex accent or O-circumflex
227	11100011	343	e3	Ò	Capital letter O with grave accent

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
228	11100100	344	e4	õ	Lowercase letter o with tilde or o-tilde
229	11100101	345	e5	Õ	Capital letter O with tilde or O-tilde
230	11100110	346	e6	µ	Lowercase letter Mu ; micro sign or micron
231	11100111	347	e7	þ	Lowercase letter Thorn
232	11101000	350	e8	Þ	Capital letter Thorn
233	11101001	351	e9	Ú	Capital letter U with acute accent or U-acute
234	11101010	352	ea	Û	Letter U with circumflex accent or U-circumflex
235	11101011	353	eb	Ù	Capital letter U with grave accent
236	11101100	354	ec	ý	Lowercase letter y with acute accent
237	11101101	355	ed	Ý	Capital letter Y with acute accent
238	11101110	356	ee	¯	Macron symbol
239	11101111	357	ef	´	Acute accent
240	11110000	360	f0	≡	Congruence relation symbol
241	11110001	361	f1	±	Plus-minus sign
242	11110010	362	f2	‗	underline or underscore
243	11110011	363	f3	¾	three quarters, three-fourths
244	11110100	364	f4	¶	Paragraph sign or pilcrow ; end paragraph mark
245	11110101	365	f5	§	Section sign
246	11110110	366	f6	÷	The division sign ; Obelus
247	11110111	367	f7	¸	cedilla
248	11111000	370	f8	°	Degree symbol
249	11111001	371	f9	¨	Diaresis
250	11111010	372	fa	·	Interpunct or space dot

Kode ASCII (desimal)	Kode ASCII (2-ary)	Kode ASCII (okta)	Kode ASCII (heksadesimal)	*Character* (karakter)	Deskripsi (Pendahuluan)
251	11111011	373	fb	¹	Superscript one, exponent 1, first power
252	11111100	374	fc	³	Superscript three, exponent 3, cube, third power
253	11111101	375	fd	²	Superscript two, exponent 2, square, second power
254	11111110	376	fe	■	black square
255	11111111	377	ff	nbsp	Non-breaking space or no-break space

RIWAYAT HIDUP

Penulis bernama Rofifah, biasa dipanggil ifah, ipeh, dan ofi, merupakan anak kedua sekaligus terakhir dari pasangan Siti Aminah dan Riawan Dardianto. Penulis lahir di Tulungagung pada tanggal 05 November 2003 dan bertempat di Desa Sukowiyono RT.01 RW.03, Kecamatan Karangrejo, Kabupaten Tulungagung.

Pendidikan dasar ditempuh di SDN 01 Bungur dan diselesaikan pada tahun 2016. Selanjutnya, penulis melanjutkan pendidikan di MTsN 01 Blitar dan lulus pada tahun 2019. Pendidikan menengah atas ditempuh di MAN 3 Blitar dan diselesaikan pada tahun 2022. Pada tahun yang sama, penulis diterima di Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang, pada Program Studi Matematika, Fakultas Sains dan Teknologi.

Selama masa perkuliahan, penulis tidak hanya berfokus pada bidang akademik, tetapi juga aktif dalam kegiatan organisasi kemahasiswaan. Penulis pernah bergabung dalam Himpunan Mahasiswa Program Studi (HMPS) Integral Matematika selama dua periode berturut-turut khususnya pada Divisi Penerbitan dan Jurnalistik, yang menjadi wadah bagi penulis untuk mengembangkan minat dan bakat di bidang penulisan serta publikasi ilmiah.

Dengan semangat belajar yang tinggi dan tekad yang kuat, penulis berusaha untuk terus mengembangkan diri baik dalam bidang akademik maupun non-akademik, hingga akhirnya dapat menyelesaikan studi di Program Studi Matematika UIN Maulana Malik Ibrahim Malang dengan sangat baik.

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 : Rofifah
NIM : 210601110059
Fakultas / Jurusan : Sains dan Teknologi / Matematika
Judul Skripsi : Implementasi Algoritma *Logistic Map* dan *Arnold's Cat Map* pada Pengamanan Pesan teks
Pembimbing I : Muhammad Khudzaifah, M.Si.
Pembimbing II : Dr. Ach. Nashichuddin, M.A.

No.	Tanggal	Hal	Tanda Tangan
1.	11 September 2025	Konsultasi Topik dan Data	1.
2.	17 September 2025	Konsultasi Bab I, II, dan III	2.
3.	24 September 2025	Konsultasi Bab I, II, dan III	3.
4.	01 Oktober 2025	Konsultasi Bab I, II, dan III	4.
5.	02 Oktober 2025	Konsultasi Kajian Agama Bab I dan II	5.
6.	03 Oktober 2025	ACC Kajian Agama Bab I dan II	6.
7.	08 Oktober 2025	Konsultasi Bab I, II, dan III	8.
8.	09 Oktober 2025	Konsultasi Bab I, II, dan III	8.
9.	10 Oktober 2025	ACC Bab I, II, dan III	9.
10.	13 Oktober 2025	ACC Seminar Proposal	10.
11.	29 Oktober 2025	Konsultasi Revisi Seminar Proposal	11.
12.	05 November 2025	Konsultasi Bab IV dan V	12.
13.	11 November 2025	Konsultasi Kajian Agama Bab IV	13.
14.	12 November 2025	ACC Kajian Agama Bab IV	14.

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

15.	14 November 2025	Konsultasi Bab IV dan V	15.
16.	19 November 2025	Konsultasi Bab IV dan V	16.
17.	26 November 2025	ACC BAB IV dan V	17.
18.	27 November 2025	ACC Seminar Hasil	18.
19.	04 Desember 2025	Konsultasi Revisi Seminar Hasil	19.
20.	16 Desember 2025	Sidang Skripsi	20.
21.	23 Desember 2025	ACC Keseluruhan	21.

Malang, 23 Desember 2025
Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M.Si.
NIP. 19800527 20080 1 012