Implementation of Message Encryption Using the Digraph Affine Cypher Algorithm with Python

Andre Aditya
Program Studi Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan, Indonesia
adityaandre425@gmail.com

Article Info

Article history:

Received September 29, 2025
Revised November 15, 2025
Accepted November 26, 2025

Keywords:

Cryptography
Classical cipher
Digraph transformation
Performance evaluation
Processing time
Python

ABSTRACT

The security of digital messages is a critical concern, necessitating robust encryption methods. While the classical Affine Cipher offers a foundational approach, its security can be enhanced. This study addresses a gap in the practical application of the Digraph Affine Cipher (DAC), a modification that encrypts character pairs (digraphs) to increase cryptographic complexity. We implemented the DAC algorithm using Python to encrypt and decrypt text files and evaluated its performance against standard Affine Cipher and ASCII-based variations. The system was tested on files ranging from 200 to 5000 characters, measuring processing time and final file size. Results showed that processing time is directly proportional to message length, with the standard ASCII Affine Cipher being the fastest and the classic DAC the slowest. Notably, the DAC produced an encrypted file size most consistent with the original plaintext, while ASCII-based versions demonstrated greater flexibility by successfully processing a wider range of symbols. The study successfully demonstrates a practical Python-based implementation of the DAC, providing a performance baseline and confirming its effectiveness in securing text data. This work serves as a foundation for developing stronger hybrid classical cryptographic techniques.

Corresponding Author:

Andre Aditya
University of Harapan Medan
Email: adityaandre425@gmail.com

1. INTRODUCTION

The security of transmitted messages is a paramount concern in modern long-distance communication. Protecting data exchange and online transactions from unauthorized access is critical, necessitating techniques to safeguard information confidentiality [1, 2]. Cryptography provides the essential methods for encoding messages, and among the classical algorithms is the Affine Cipher, a type of monoalphabetic substitution cipher that operates on a character-by-character basis using two distinct keys [3]. A significant enhancement to this method is the Digraph Affine Cipher (DAC). This algorithm modifies the standard Affine Cipher by encrypting pairs of characters (digraphs) simultaneously, effectively categorizing it as a classic block cipher. This approach substantially increases the cryptographic complexity by squaring the size of the character space, denoted as 'n' [4].

The word cryptography, derived from Greek for "secret writing" [5], is the study of mathematical techniques for information security aspects like confidentiality and data integrity [6-8]. It involves encryption and decryption, where security relies on the secrecy of keys rather than the algorithm itself [9], and is part of the broader field of cryptology [10, 11]. An example of a classical algorithm is the Affine Cipher, a pre-digital, symmetric-key substitution cipher that uses two keys for encryption [3, 4, 12]. To enhance its security, a digraph transformation can be applied, allowing the algorithm to process plaintext in pairs of characters and requiring padding for texts with an odd length [13].

Previous research has explored various facets of the Affine Cipher. For instance, Wibowo (2018) focused on implementing the standard Affine Cipher for text messaging on the Android platform [12]. Similarly, Sitompul et al. (2018) analyzed the performance of a hybrid algorithm combining RC4 and the Affine Cipher to secure SMS messages [3]. More directly, Ritonga et al. (2020) investigated the theoretical modification of the Affine Cipher using a digraph transformation to enhance text security [4]. However, a notable gap persists in the literature regarding the practical implementation of the Digraph Affine Cipher, especially within a versatile programming environment like Python that is not tied to a specific mobile operating system.

This study aims to address this gap by focusing on the implementation of message encryption and decryption using the Digraph Affine Cipher algorithm in Python. The primary research questions are: (1) How is the encryption process executed using the Digraph Affine Cipher to secure text messages? (2) How is the corresponding decryption process performed to recover the original text?

The scope of this research is confined to the analysis of string data from .txt files, which include a character set of letters, numbers, and common symbols. The performance of the algorithm will be evaluated based on key parameters, specifically the impact on file size and the processing time required for both encryption and decryption operations. The main objective is to practically demonstrate the enhanced security provided by the Digraph Affine Cipher. The findings are intended to deepen the understanding of this modified algorithm's performance and provide a valuable foundation for developers looking to implement stronger classical cryptographic techniques.

2. METHOD

The methodology employed in this study follows a structured and systematic approach, encompassing several distinct phases from foundational analysis to final implementation and evaluation. Each stage was conducted sequentially to ensure a thorough and coherent research process.

3.1 Literature Review

The initial phase consisted of a comprehensive literature review to gather, explore, and study information pertinent to the research topic. This involved consulting a wide range of sources, including academic textbooks, reference books, and scholarly articles obtained from the internet, to build a strong theoretical foundation for the study.

3.2 System Analysis and Design

Following the literature review, a system analysis was performed to thoroughly examine the core problem, the operational processes of the Affine Cipher, and the proposed solution using the digraph transformation technique. Based on the insights gained from this analysis, the system design phase was undertaken. This stage focused on architecting a solution to address the identified requirements and defining the logical flow of both the encryption and decryption processes.

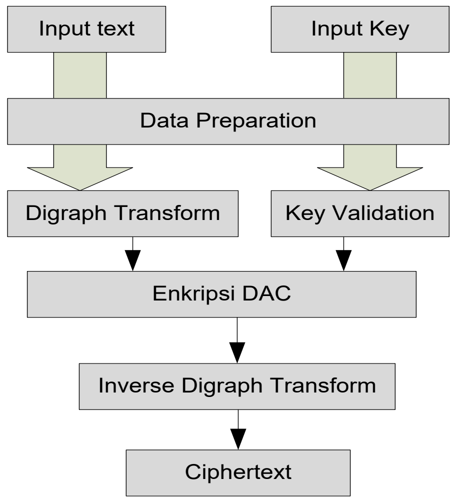


Figure 1. Block Diagram of the Affine and Vigenere Cipher Combination

3.3 Implementation

The design was brought to life during the program implementation phase. In this stage, the standard Affine Cipher algorithm was modified using the digraph technique for message encryption. The entire system was developed using the Python programming language.

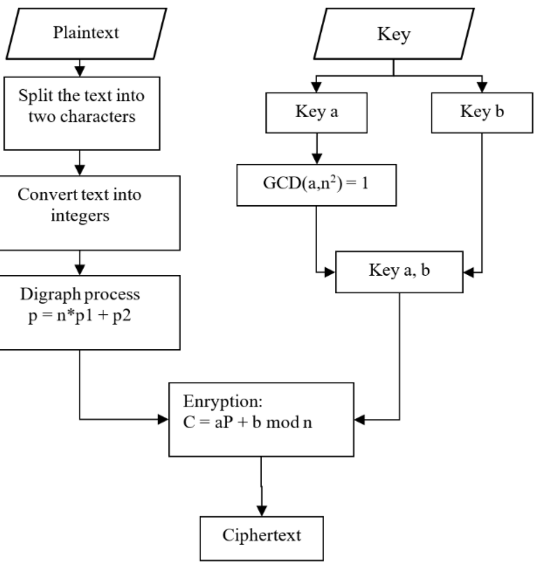

Plaintext
Key
Split the text into two characters
Key a
Key b
Convert text into integers
GCD(a,n²) = 1
Key a, b
Digraph process p = n*p1 + p2
Enryption: C = aP + b mod n
Ciphertext

Figure 2. Flowchart of the Affine Transform Digraph Encryption Process

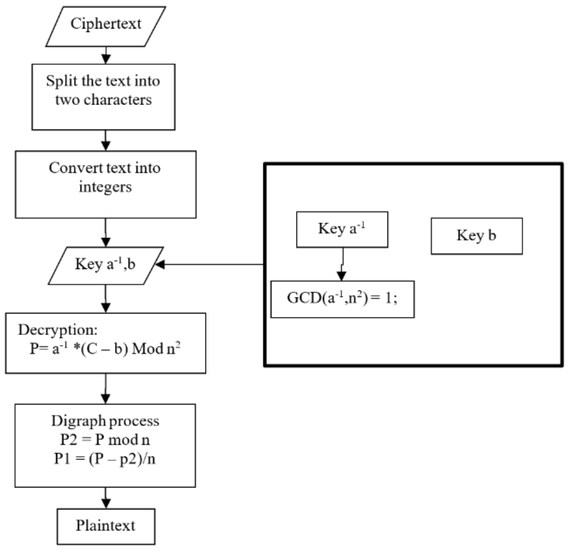

Ciphertext
Split the text into two characters
Convert text into integers
Key a⁻¹,b
Key a⁻¹
Key b
GCD(a⁻¹,n²) = 1;
Decryption: P= a⁻¹ *(C – b) Mod n²
Digraph process P2 = P mod n P1 = (P – p2)/n
Plaintext

Figure 3. Flowchart of the Affine Transform Digraph Decryption Process

3.4 Testing and Evaluation

To validate the system, a testing phase was conducted. The primary objective was to verify the successful encryption of plaintext messages into ciphertext and, subsequently, the accurate decryption of that ciphertext back into its original, readable form. The performance of the algorithm was also evaluated based on metrics such as processing time and the effect on file size, as outlined in the problem limitations.

3.5 Report Writing

The final stage of the methodology was the preparation of the research report. This involved documenting the entire process, including the theoretical background, system design, implementation details, and the results of the testing and evaluation, to form this comprehensive final project report.

3. RESULTS AND DISCUSSION

At this stage, the author will conduct research on the process of the Affine cipher algorithm with digraph transformation. In the next stage, the author will perform encryption on the Affine Cipher Classic, Affine Cipher ASCII, Digraph Affine Cipher, and Digraph Affine ASCII algorithms. Then, a comparison will be made on the time required to perform the encryption and decryption processes. The author will also compare the pair of keys ‘a’ and the value n possessed by each algorithm.

3.1. Affine Cipher Encryption Testing

Encryption testing using Affine Cipher was performed on a text consisting of 200 characters, namely “*Kriptografi merupakan seni atau ilmu yang mempelajari teknik-teknik matematika untuk aspek menjaga keamanan informasi seperti kerahasian data, keabsahan data, integritas data, serta autentifikasi data*”.

Table 1. Affine Cipher Encryption on 200 characters

No	Algorithm	Encryption	Decryption
1	Affine Cipher Classic	ETWLBUOTQXWMGTSLQEQDKGDWQBQSWVM SIQDOMGMLGVQNQTWBGEDWEOBGEDWEMQ BGMQBWEQSDBSEQKLGEMGDNQOQEGQMQD QDWDXUTMQKWKGLGTBWEGTQFQKWQQDP QBQXEGQHKQFQDPQBQXWDBGOTWBQKPQB QXKGTBQQSBGDBWXWEQKWPQBQ	KRIPTOGRAFIMERUPAKANSENIATAU ILMUYANGMEMPELAJARITEKNIKGTE KNIKMATEMATIKAUNTUKASPEKME NJAGAKEAMANANINFORMASISEPER TIKERAHASIAANDATAFKEABSAHAN DATAFINTEGRITASDATAFSERTAAUT ENTIFIKASIDATA
2	Affine Cipher ASCII	õŒój®YÑŒkÀó7¯Œ¿jkkH□¯Hók®k¿ó&7¿— kHÑ7¯7j¯&k— kŒó®¯Hó÷®¯Hó7k®¯7k®ók¿H®¿k□j¯7¯H— kÑk¯k7kHkHóHÀYŒ7k□ó□¯j¯Œ®ó¯Œkâk□ókkH□ k®kæ¯k|□kâkH□k®kæóH®¯ÑŒó®k□□k®kæ□¯Œ ®kk¿®¯H®óÀók□ó□k®k	Kriptografi merupakan seni atau ilmu yang mempelajari teknik-teknik matematika untuk aspek menjaga keamanan informasi seperti k erahasiaan data, keabsahan data, integritas da ta, serta autentifikasi data
3	Digraph Affine Cipher	A TTRCN ML R VNXVHRHDRUVTITHIHORRFJX HKRMTR VGTDFOZT RKVZMQPFGCPTIGRYHK VYHNIRHEX GEPTHATCP WEMAHDHZPXHYHO HDRSMWCIWUQ RUVXVNG RTVBHUHXIJHDR GHIHTRTVJYSHUHDRGHIHTRSMKVN WGUQV EVGXBDQG IHTHKGEMNISIRHXIVEVGZR	KRIPTOGRAFI MERUPAKAN SENI ATA U ILMU YANG MEMPELAJARI TEKNIK HTEKNIK MATEMATIKA UNTUK ASPE K MENJAGA KEAMANAN INFORMASI SEPERTI KERAHASIAAN DATAG KEA BSAHAN DATAG INTEGRITAS DAT G S ERTA AUTENTIFIKASI DATA
4	Digraph Affine ASCII	—Œj¼YßŒ xÀüD¯š¿wk"kQª¯Uó'k»kÈ&E¿)— yHÚD¯Ej½&y— yŒü»¯#H®½Uó-Dk»¯Dk»ó"k(¿V®Í'k«j½(7½HkÞk(¼ kDkUkQHÎYš7y□üª¯w¯š®ü"¯™kïkªóxkQ«k»kï"¯x|ªkï kQ«k»kïH»¯ßŒ®y□'□y®tæ(□½Œ»k'kÍ®½H»óÍó"kªó '□y®t	Kriptografi merupakan seni atau ilmu yang mempelajari teknik-teknik matematika untuk aspek menjaga keamanan informasi seperti k erahasiaan data, keabsahan data, integritas da ta, serta autentifikasi data

The results of testing 200 characters can be seen in Table 1. Affine Cipher Classic and Digraph Affine Cipher did not succeed in performing perfect decryption because in the 200-character test, there were several characters that were not included in the character vocabulary of the two affine algorithms, including the line and comma symbols.

3.2. Average testing time

Processing time testing was conducted to determine the effect of character length on the processing time of the affine cipher algorithm and the digraph transformation affine cipher algorithm, calculated in milliseconds (ms). The testing process was conducted on characters with a total of 200, 500, 1000, 2500, and 5000 characters. Each character was tested 5 times, and then the average time was obtained.

Table 2. Average testing time

Algorithm	Number of characters tested				
	200	500	1000	2500	5000
AFFINE CIPHER CLASSIC	1,2014	2,2026	2,6008	4,6008	6,6038
AFFINE CIPHER ASCII	1,5998	1,6008	2,4004	3,2182	5,8024
AFFINE CIPHER DIGRAPH	3,6028	5,0024	6,202	10,006	20,2108
AFFINE DIGRAPH ASCII	2,4012	3,0016	3,6028	8,6064	13,2096

Table 2 shows data compiled from the average processing time per number of characters tested. The fastest average processing time was for the affine cipher with ASCII code. The slowest was for the digraph affine cipher. The graph can be seen in Figure 4.

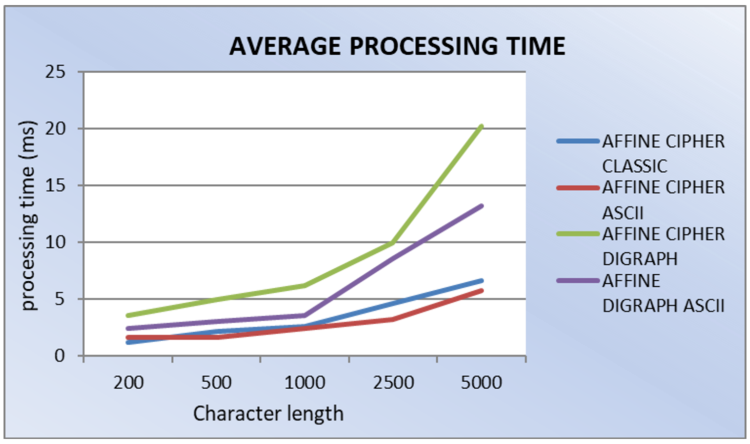
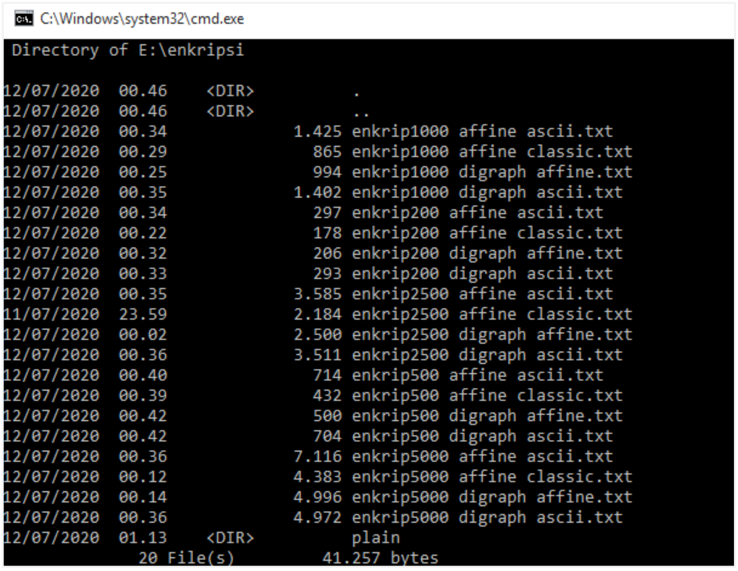


Figure 4. Character Length vs. Encryption Processing Time Graph

Figure 4 shows that character length is directly proportional to processing time. In this study, the more characters encrypted, the longer the processing time.

3.2. Encrypted File Size

Testing of file size was conducted to determine the effect of character length on the size of files encrypted using the affine cipher algorithm and the affine cipher digraph transformation algorithm, calculated in bytes. Testing was conducted on files containing 200, 500, 1000, 2500, and 5000 characters.



```
C:\Windows\system32\cmd.exe
 Directory of E:\enkripsi

12/07/2020  00.46    <DIR>          .
12/07/2020  00.46    <DIR>          ..
12/07/2020  00.34             1.425 enkrip1000 affine ascii.txt
12/07/2020  00.29               865 enkrip1000 affine classic.txt
12/07/2020  00.25               994 enkrip1000 digraph affine.txt
12/07/2020  00.35             1.402 enkrip1000 digraph ascii.txt
12/07/2020  00.34               297 enkrip200 affine ascii.txt
12/07/2020  00.22               178 enkrip200 affine classic.txt
12/07/2020  00.32               206 enkrip200 digraph affine.txt
12/07/2020  00.33               293 enkrip200 digraph ascii.txt
12/07/2020  00.35             3.585 enkrip2500 affine ascii.txt
11/07/2020  23.59             2.184 enkrip2500 affine classic.txt
12/07/2020  00.02             2.500 enkrip2500 digraph affine.txt
12/07/2020  00.36             3.511 enkrip2500 digraph ascii.txt
12/07/2020  00.40               714 enkrip500 affine ascii.txt
12/07/2020  00.39               432 enkrip500 affine classic.txt
12/07/2020  00.42               500 enkrip500 digraph affine.txt
12/07/2020  00.42               704 enkrip500 digraph ascii.txt
12/07/2020  00.36             7.116 enkrip5000 affine ascii.txt
12/07/2020  00.12             4.383 enkrip5000 affine classic.txt
12/07/2020  00.14             4.996 enkrip5000 digraph affine.txt
12/07/2020  00.36             4.972 enkrip5000 digraph ascii.txt
12/07/2020  01.13    <DIR>          plain
              20 File(s)         41.257 bytes
```

Figure 5. Command Prompt from the encryption results directory

The file size of the encryption results shown in Figure 5 produced the file size test data shown in Table 3 below.

Table 3. File size test results for the number of characters tested

Algorithm	**Number of characters tested (byte)**				
	200	**500**	**1000**	**2500**	**5000**
AFFINE CIPHER CLASSIC	178	432	865	2,184	4,383
AFFINE CIPHER ASCII	297	714	1,425	3,585	7,116
AFFINE CIPHER DIGRAPH	206	500	994	2,500	4,996
AFFINE DIGRAPH ASCII	293	704	1,402	3,511	4,972

Table 3 shows data compiled from the average processing time per number of characters tested. The fastest average processing time was for the affine cipher with ASCII code. The slowest was for the digraph affine cipher. The graph can be seen in Figure 6.

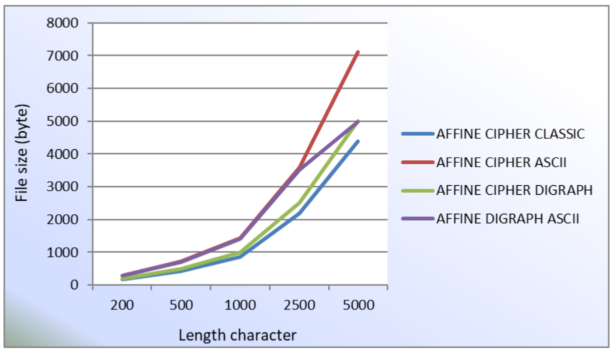


Figure 6. Graph of Character Length Against Ciphertext File Size

Figure 6 shows that character length is directly proportional to file size. In this study, the encryption algorithm that produced the most consistent file size close to the plain text file size was the Affine Cipher Digraph algorithm.

4. CONCLUSION

The message encryption process in the Digraph Affine Cipher algorithm produces a coded message (ciphertext). The encryption process (plaintext) is performed after converting the plaintext using an ASCII table limited to 127. The message encryption process in the Digraph Affine Cipher algorithm produces a coded message (ciphertext). The encryption process (plaintext) is performed after converting the plaintext using the ASCII table. The decryption process with the Digraph Affine Cipher algorithm can convert ciphertext into plaintext. The decryption process is the process of returning the ciphertext to its original plaintext. This is because the ciphertext obtained is the ciphertext of the message. Recommendations for future research include combining modern algorithms with modified affine cipher digraph transformations in key generation or encryption to make text messages more secure. Digraph transformations are also expected to be applicable in several other encryption algorithms.

REFERENCES

[1] A. Ray, A. Potnis, P. Dwivedy, S. Soofi, and U. Bhade, “Comparative study of AES, RSA, genetic, affine transform with XOR operation, and watermarking for image encryption,” in *2017 International Conference on Recent Innovations in Signal Processing and Embedded Systems (RISE)*, 2017, pp. 274–278.
[2] H. Siagian, “Analisis Kinerja Kombinasi Algoritma Transposisi Dan Rsa Dalam Pengamanan Data Pada Steganografi,” 2016.
[3] O. S. Sitompul, N. H. Pasaribu, and E. B. Nababan, “Hybrid RC4 and Affine Ciphers to Secure Short Message Service on Android,” in *2018 Third International Conference on Informatics and Computing (ICIC)*, 2018, pp. 1–6.
[4] R. P. Ritonga, M. Zarlis, and E. B. Nababan, “Modification affine cipher transform digraph to squared the value of ‘n’in text security,” in *2020 4th International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*, 2020, pp. 124–128.
[5] M. Rinaldi, *Kriptografi*. Bandung: Penerbit Informatika, 2006.
[6] E. S. Ompusunggu and E. B. Nababan, “Modification of variably modified permutation composition (vmpc) algorithm genetic key algorithm for data security,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 725, no. 1, p. 12123, 2020.
[7] Y. S. Santoso, “Pengamanan Pesan Menggunakan Kombinasi Algoritma Hill Cipher dan RSA,” 2016.
[8] G. T. Simbolon, O. S. Sitompul, and E. B. Nababan, “Data Security Using Multi-bit LSB and Modified Vernam Cipher,” 2019.
[9] S. Kromodimoeljo, *Teori dan aplikasi kriptografi*. SPK IT Consulting, 2009.
[10] D. Rachmawati and A. Candra, “Implementasi Kombinasi Caesar dan Affine Cipher untuk keamanan Data Teks,” *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, no. 2, pp. 60–63, 2015.
[11] A. I. Permana, “Kombinasi Algoritma Kriptografi One Time Pad dengan Generate Random Keys dan Vigenere Cipher dengan Kunci EM2B,” 2019.
[12] S. Wibowo, F. E. Nilawati, and S. Suharnawi, “Implementasi Enkripsi Dekripsi Algoritma Affine Cipher Berbasis Android,” *Techno. Com*, vol. 13, no. 4, pp. 215–221, 2014.
[13] R. P. Ritonga, “Super Enkripsi Digraph Affine Cipher (DAC) dan Advanced Encryption Standard (AES) dengan Kombinasi Kunci DAC-MD5,” 2021.