Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026. Vol. 5. No. 3; e-ISSN: 2808-4519

Implementation of the Vigenere Cipher Algorithm in a Python GUI-Based Text Encryption App

Mardiyyah Alvita Amelia[1]*, Yusnia Budiarti[2], Siti Hujaemah[3], Alvianus Artha Desienda[4], Farhan Afrian Nanda[5], Muhammad Fakhri Husaini[6], Ryan Hartono[7]

[1,2,3,4,5,6, 7] Universitas Bina Sarana Informatika

alviameliaaa@gmail.com[1], yusnia.ybi@bsi.ac.id[2], sitihujaemah0102@gmail.com[3], alvianus2004@gmail.com[4], farhanafriannanda123@gmail.com[5], husainifahri287@gmail.com[6], hrtchill15@gmail.com[7]*

Abstract

Rapid advances in information technology have increased the risk of data theft and misuse in digital communications. Cryptography is a scientific solution for ensuring the confidentiality, integrity, and authenticity of data. The Vigenère Cipher is one of the best-known classical cryptographic algorithms, developed by Blaise de Vigenère in the 16th century, which uses a polyalphabetic substitution method based on a repeating key and falls under the category of symmetric cryptography. This research implements the Vigenère Cipher algorithm in a desktop application named VigCrypt, built using the Python 3 programming language and the Tkinter framework as the graphical user interface (GUI). Unlike previous research, which mostly implemented the Vigenère Cipher on web- or mobile-based platforms, this study focuses on a desktop application that serves not only as an encryption tool, but also as a medium for learning cryptography. The developed application features text encryption and decryption, visualization of the calculation steps for each character—displaying numerical values and modular calculations explicitly—a 26x26 Vigenère Square table as a visual reference, and the ability to export results to a .txt file. Functional testing was conducted on 10 plaintext samples covering various text types, including short text, sentence-length text, and alphanumeric text. The test results showed a 100% accuracy rate for both encryption and decryption, with all non-alphabetic characters—such as spaces, numbers, and punctuation marks—successfully preserved.

Keywords: *Vigenère Cipher, Symmetric Cryptography, Python, GUI, Tkinter, Text Encryption and Decryption*

1. Introduction

Rapid advances in information technology have had a significant impact on data security. Every day, billions of digital messages are sent through various communication platforms, thereby increasing the risk of information misuse. Cryptography offers a scientific solution to ensure the confidentiality, integrity, and authenticity of data.

Cryptography is a field of study that is still used primarily to protect confidential data. Various research and development efforts continue to be carried out in order to develop methods that are difficult for unauthorized entities to crack[1]. The confidentiality of information must be protected during remote data transmission. Therefore, such information can be protected by applying cryptography[2].

Some concepts related to cryptography. The sender is the entity that sends a message to the recipient, ensuring that the message can be transmitted securely. Plaintext is the original message that is fed into a cryptographic algorithm, which produces ciphertext—that is, an encrypted message that has undergone a series of processes to transform the plaintext into ciphertext [1].

One of the most robust classical ciphers is the Vigenère cipher, invented in 1553 by Giovan Battista Bellaso. It is extremely difficult to decrypt text encrypted with this algorithm without the key, which led to the discovery of the Kasiski method in 1863. However, since the discovery of this method, the Vigenère cipher has become very vulnerable to being broken for long texts [3].

This study aims to implement the Vigenère cipher algorithm in a Python-based desktop application using the Tkinter framework. The application was developed and designed not only as a functional encryption tool but also as a cryptography learning tool that transparently displays the calculation process, features an interactive Vigenère cipher table, and allows users to save results to a .txt file.

With this application, users are expected to gain a direct and visual understanding of how the Vigenère cipher algorithm works, as well as to apply it as a simple information security tool in a desktop environment.

2. Theoritical Review

2.1. Cryptography

Cryptography is both a science and an art that protects messages to keep them secure. “Crypto” means “secret,” while “graphy” refers to “writing.” People or entities that apply cryptography are known as cryptographers. A cipher is a cryptographic algorithm. A cipher is a mathematical formula applied during the encryption and decryption processes. Generally, the formulas for the encryption and decryption steps are closely related mathematically. (Wicida Journal) A cryptanalyst is someone who attempts to break an encrypted message, while a cryptologist is someone who studies cryptography[3].

In symmetric algorithms, the key used to encrypt data is the same. There are many symmetric algorithms, such as Blowfish, DES, AES, and so on. Each algorithm uses a different method to encrypt and decrypt data. The components of symmetric encryption are as follows:

1. Plaintext: is the original data to be sent to a specific recipient. This data serves as the input to the encryption algorithm.
2. Encryption algorithm: is a series of processes applied to the plaintext using a secret key to generate ciphertext.
3. Secret key: is the value used to process the plaintext and convert it into ciphertext.
4. Ciphertext: is the output of the original plaintext that is fed into the encryption algorithm.
5. Decryption algorithm: is a set of processes that will be executed on the ciphertext with the help of a secret key to produce the original text or plaintext[4].

2.2. Vigenere Cipher

The Vigenère cipher is one of the classical cryptographic algorithms introduced in the 16th century, or around 1553. This cryptographic algorithm was published by a French diplomat and cryptographer named Blaise de Vigenère, although it had actually been described earlier in the book *La Cifra del Sig. Giovan Battista Belaso, a book written by Giovan Battista Belaso in 1553[5].

The encryption and decryption formulas for the Vigenère cipher are defined as follows:

Enkripsi : C[i] = (P[i] + K[i]) mod 26
Dekripsi : P[i] = (C[i] – K[i] + 26) ,od 26

Notes:
P[i] = The numerical value of the i-th plaintext character (A=0, B=1, ..., Z=25)
K[i] = The numerical value of the i-th key character (repeated throughout the text)
C[i] = The numerical value of the i-th ciphertext character

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

2.3. Python

Python is a programming language that uses an interpreter to execute its code. The interpreter translates the code on the fly, and Python can run on various platforms such as Windows, Linux, and others. This highly popular programming language was first created by Guido van Rossum at the Centrumm voor Wiskundig Onderzoek (CWI) in Amsterdam in 1991. The development of Python was inspired by the ABC programming language, which was gaining traction at the time. One of the main differences between Python and other programming languages is that its development involves millions of programmers, researchers, and users from diverse backgrounds—not just the IT community—because Python is open-source.

There are several reasons why Python is the top choice, namely:

1. Python runs on various platforms, such as Windows, Linux, macOS, Android, Raspberry Pi, and others.
2. Python has a simple syntax that resembles English.
3. Python's syntax allows for more concise code compared to other programming languages.
4. Python uses an interpreter, so programs can be executed quickly once they are written.
5. Python supports procedural, object-oriented, and functional programming paradigms[6].

Python has 12 modules for creating Graphical User Interface (GUI) applications, one of which is Tkinter. Tkinter is the most commonly used method because it is easy enough for beginners to use to learn the basics of GUIs. A Graphical User Interface (GUI) is an interface that allows users to interact with the operating system through a set of elements, commonly known as widgets, such as list boxes, radio buttons, icons, and dialog boxes. One of the advantages of a GUI is that users can receive visual or graphical responses to questions they have previously entered. (Tkinter GUI)

3. Result and Discussion

Based on the design described in the previous chapter, the VigCrypt application was successfully implemented using the Python 3 programming language with the Tkinter framework as the graphical user interface (GUI).

3.1. Use case Diagram Modelling

A use case diagram is a visual representation of various components, such as actors, use cases, and the relationships between them. Various symbols and notations are used to depict a system's functionality in a use case diagram. Use case diagrams can aid in the analysis and formulation of system development requirements. They are used to explain the system design to users and to design all the features of the system to be built [7].

Fig .1: Usecase Diagram

The use case diagram for this cryptography application features a single actor the user who interacts directly with the system through seven main use cases, starting with selecting a mode (encryption or decryption), entering input text (plaintext or ciphertext), entering a key, and processing the text. When the process is executed, the system automatically invokes the "Input & Key Validation" use case via an include relationship, and if the key is invalid, the "Display Error Message" use case is triggered via an extend relationship. After the process succeeds, the user can view the output, trace the steps of the algorithm used, and save the results, which automatically invokes the "Export to .txt File" use case via an include relationship.

3.2. Activity Diagram Modelling

An activity diagram presents the flow of processes or activities within a system to be built, starting from the initial process, the decisions made within the system, to how a process ends. An activity diagram also visualizes the parallel processes occurring within the system. Each use case must include at least one activity diagram. An activity diagram is designed based on one or more use cases found in the use case diagram. An activity diagram represents the processes running within a system, while a use case represents how an actor uses the system to perform activities [7].

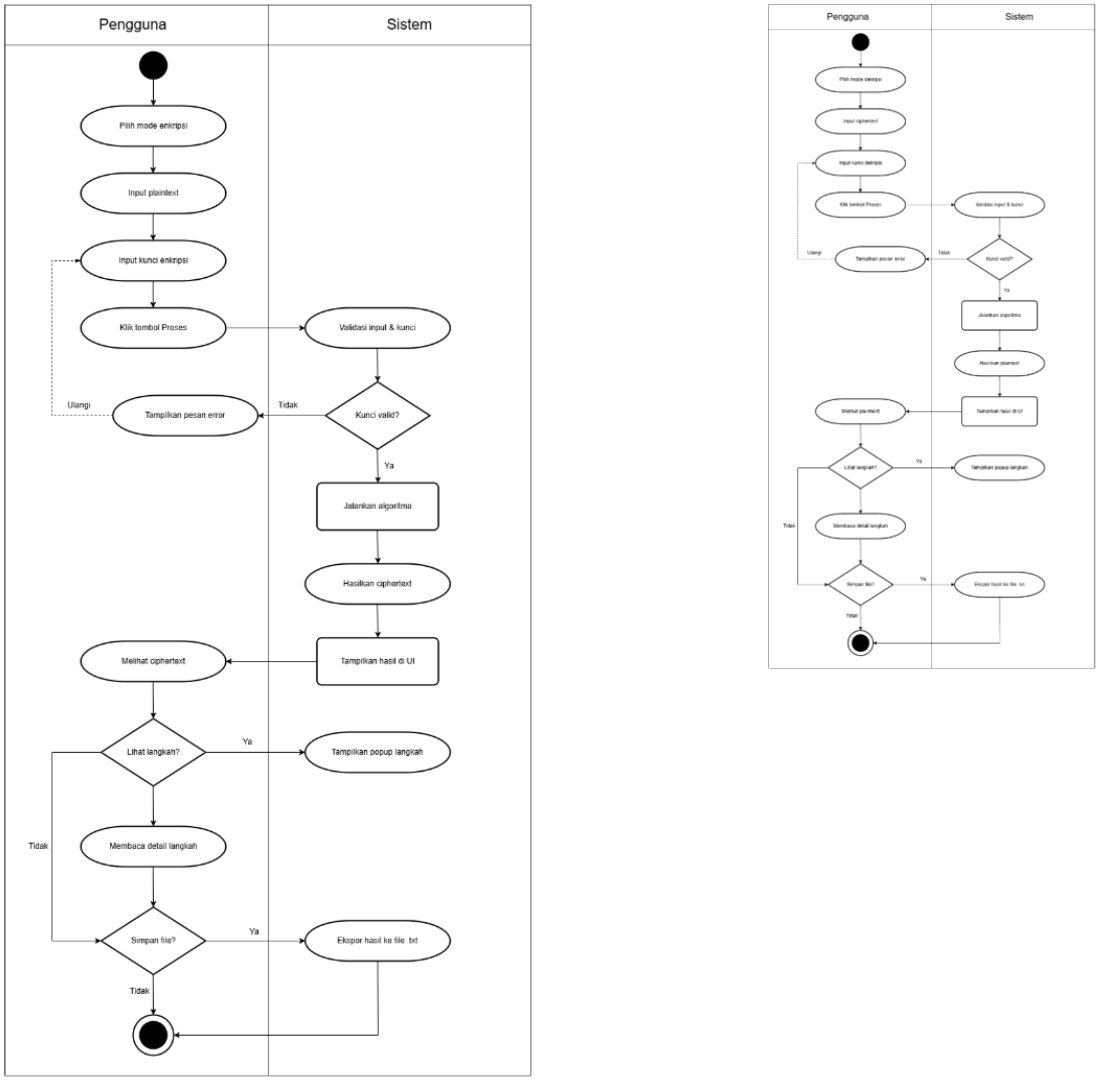

Fig 2: Activity Diagram decryption and encryption

The activity diagram for this cryptographic application consists of two main parallel workflows: the decryption workflow (left) and the encryption workflow (right), both of which involve two swimlanes: User and System. In the decryption flow, the user begins by selecting a decryption mode, then enters the ciphertext and decryption key, and clicks the Process button so that the system validates the input and key; if the key is invalid, the system displays an error message and the user is asked to re-enter the input, but if it is valid, the system runs the algorithm and generates plaintext that is displayed in the UI. Similarly, in the encryption flow, the user selects the encryption mode, enters the plaintext and encryption key, then clicks the Process button so that the system validates the input and key using the same branching logic; if valid, the system runs the algorithm and generates ciphertext, which is displayed on the UI. After the results are displayed, in both workflows the user is presented with two identical additional decisions: whether to view the algorithm steps, if yes the system displays a step popup and the user can read the details and whether to save the file, if yes the system exports the results to a .txt file, if not the workflow ends immediately.

3.3. Application Interface

The following results were obtained from the development of an application using the Python programming language and cryptographic methods based on the Vigenère cipher.

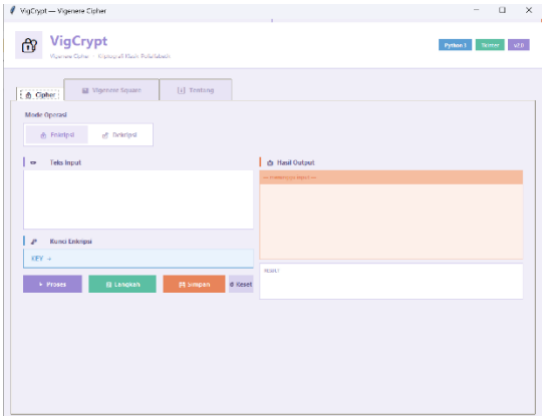

Fig3: VigCrypt App Interface

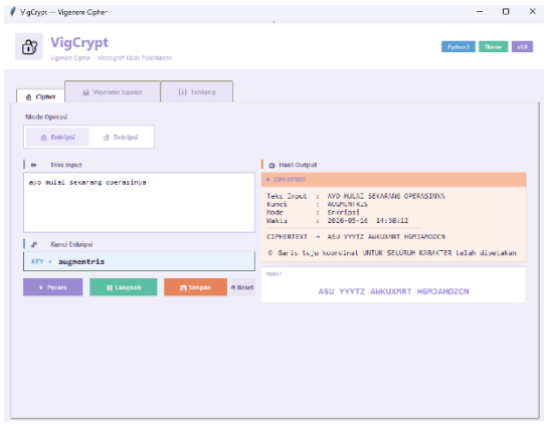

Fig .4: Encryption Results Display

The test shown in Figure 4, using the plaintext “let’s start the operation now” and the encryption key “augmentris,” produced the ciphertext: “ASU YYYTZ AWKUXMRT HGMJAMOZCN”.

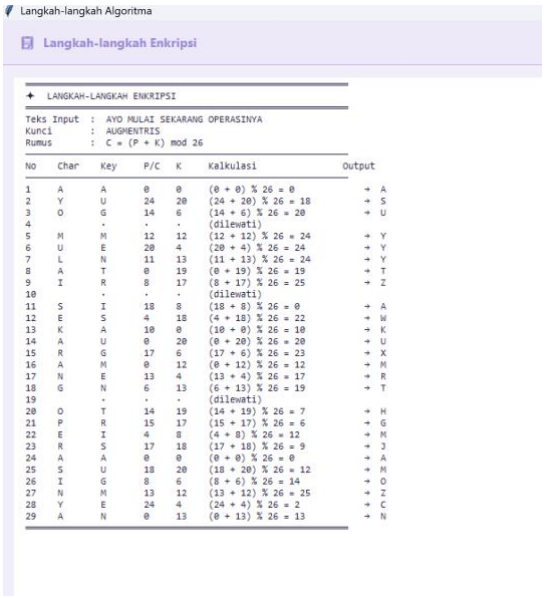

Fig .5: Overview of the Encryption Process

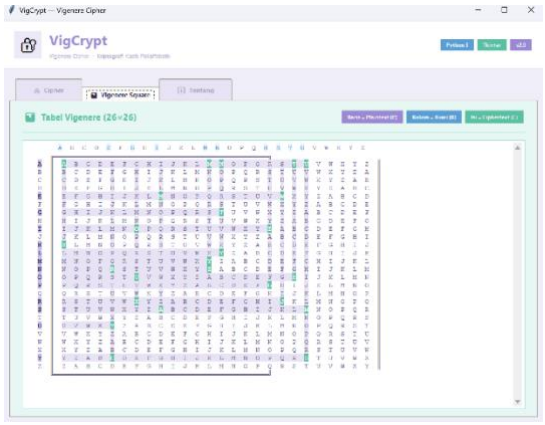

Fig.6: Encryption Table Display

Testing the application shown in Figures 4 through 6 using the plaintext “let’s start the operation now” and the encryption key “augmentris” produced the ciphertext: “ASU YYYTZ AWKUXMRT HGMJAMOZCN.” At the time of writing, this was conducted on May 16, 2026. After obtaining the ciphertext from the encryption result, the decryption process was performed, which involves converting the ciphertext back to plaintext using the same key. The first decryption test used the ciphertext: “ASU YYYTZ AWKUXMRT HGMJAMOZCN” and the key “augmentris”.

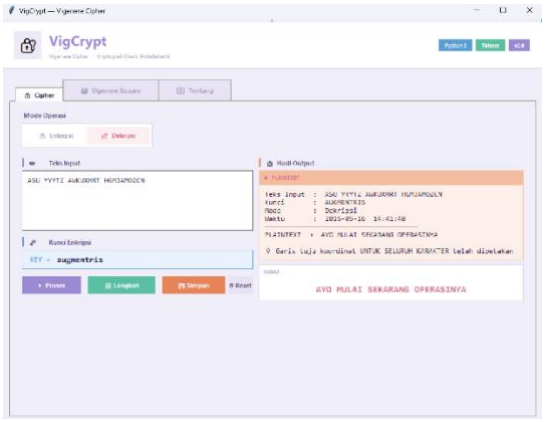

Fig.7: Decryption Results Display

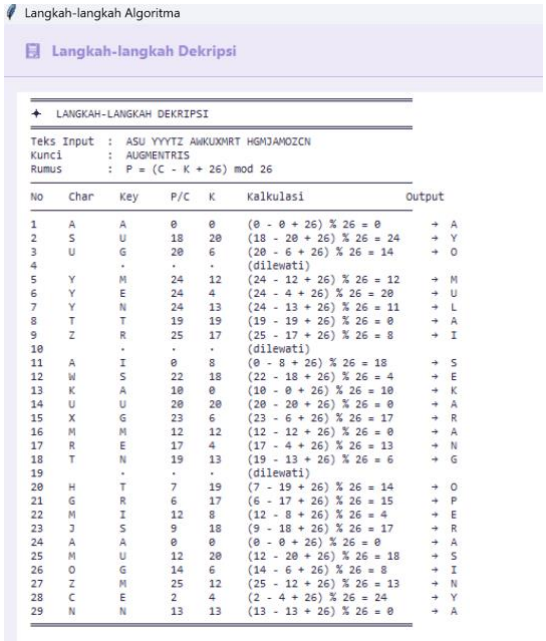

Fig.8: Decryption Process Overview

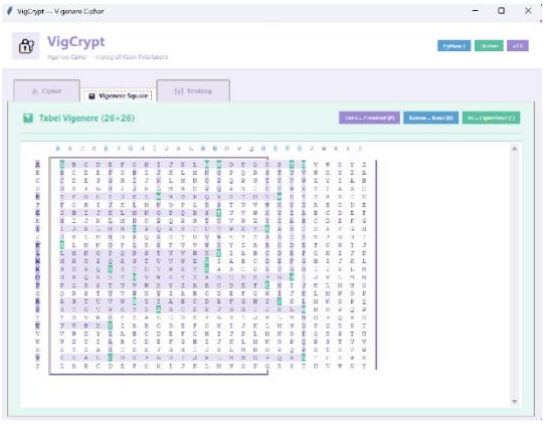

Fig .9: Decryption Table Display

The decryption results shown in Figures 7 through 9, using the ciphertext: “ASU YYYTZ AWKUXMRT HGMJAMOZCN” and the key “augmentris,” yield the plaintext “let’s start the operation now.”

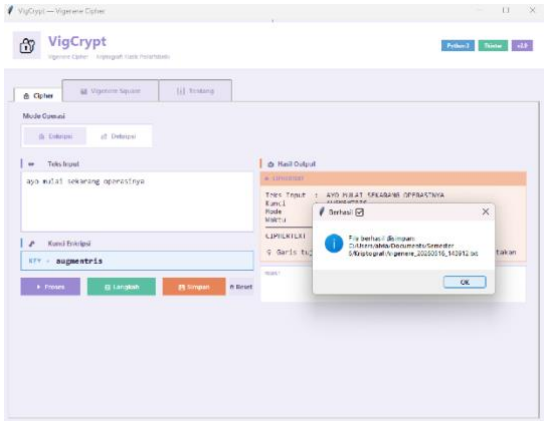

Fig. 10: File Saved Successfully Display

3.4. Test Results

The following shows some of the results of the tests conducted:

Table 1: Tests conducted

No.	Plaintext	Kunci	Ciphertext	Dekripsi	Status
1	hello	key	rijvs	hello	benar
2	belajar	cipher	dmahnrt	belajar	benar
3	vigenere cipher	secret	nmnxviv uihlxj	vigenere cipher	benar
4	selamat pagi	kunci	cgxcqcn zche	selamat pagi	benar
5	python gui	algo	prfqbv twu	python gui	benar
6	kriptografi	aman	kziplmkiafi	kriptografi	benar
7	data 123	key	nkxk 123	data 123	benar
8	ubsi	ti	njlq	ubsi	benar
9	enkripsi	rahasia	vilisaay	enkripsi	benar
10	hello world!	key	rijvs uyvjn!	hello world!	benar

4. Conclusion

This study successfully implemented the Vigenère Cipher algorithm in a desktop application called VigCrypt, built using the Python 3 programming language and the Tkinter framework as its graphical user interface (GUI). The developed application not only functions as a tool for encrypting and decrypting text but is also designed as an interactive and transparent medium for learning cryptography.

Based on functional testing conducted on 10 plaintext samples including short texts, sentence texts, and alphanumeric texts the VigCrypt application demonstrated 100% accuracy in both the encryption and decryption processes. All non-alphabetic characters, such as spaces, numbers, and punctuation marks, were successfully preserved without alteration, consistent with the characteristics of the Vigenère Cipher algorithm. The decryption process was also proven capable of perfectly restoring the ciphertext to its original plaintext form using the same key.

Features such as step-by-step character calculation visualization, a 26×26 Vigenère Square table, and the ability to export results to a .txt file give this application added value compared to similar implementations, particularly as a tool for teaching classical cryptography in a desktop environment.

5. Suggestion

Based on the research findings and existing limitations, the following recommendations for further development are as follows:

1. Development of cryptographic algorithms — The application can be enhanced by adding other cryptographic algorithms such as the Caesar Cipher, Hill Cipher, or modern algorithms like AES, allowing users to directly compare various encryption methods within a single platform.
2. Enhanced key security — Given that the Vigenère Cipher is vulnerable to Kasiski attacks and frequency analysis, particularly with long texts, it is recommended to add a key scrambling mechanism or combine the Vigenère Cipher with other cryptographic techniques to improve resistance to cryptanalysis.
3. Input/output file format support — Future development could include direct encryption of text files (.txt, .docx, .pdf), thereby expanding the application's scope and practicality.
4. Platform development — The application is currently only available on Python-based desktop platforms. It is recommended to develop web-based or mobile versions so that it can be accessed more widely by users on various devices.
5. Addition of cryptanalysis features — For educational purposes, the application can be equipped with attack simulation features such as character frequency analysis or key length detection, so that users can directly understand the weaknesses of the Vigenère cipher algorithm.
6. User interface improvements — The user interface can be further improved by adding interactive help and tutorials to make it more user-friendly for those new to cryptography.

References

[1] M. Arrijal, R. Efendi, and B. Susilo, "Penerapan Algoritma Kriptografi Kunci Simetris Dengan Modifikasi Vigenere Cipher Dalam Aplikasi Kriptografi Teks," Jurnal Pseudocode, vol. 3, no. 1, pp. 69–82, 2016.
[2] Aulansari, D. Sawitri, and A. Ikhwan, "PENERAPAN KRIPTOGRAFI VIGENERE CIPHER PADA KEAMANAN DATA PESAN TEKS BERBASIS WEBSITE," Jurnal Informatika Teknologi dan Sains, vol. 4, pp. 421–426, Nov. 2022, doi: 10.51401/jinteks.v4i4.2155.
[3] Ridho and C. R. Niani, "Implementasi Enkripsi Dengan Vigenere Cipher Dan Reverse Cipher Menggunakan Bahasa Pemrograman Python," Jurnal Teknologi Informasi, vol. 1, no. 1, pp. 9–15, 2022.
[4] Amrulloh and E. Ujianto, "Kriptografi Simetris Menggunakan Algoritma Vigenere Cipher," Jurnal CoreIT, vol. 5, no. 2, 2019, [Online]. Available: https://program.arfianhidayat.com/kriptografi/vig
[5] Efrandi and Y. Asnawati, "APLIKASI KRIPTOGRAFI PESAN MENGGUNAKAN ALGORITMA VIGENERE CIPHER," 2014.
[6] S. Rahman et al., "PYTHON : DASAR DAN PEMROGRAMAN BERORIENTASI OBJEK TAHTA MEDIA GROUP."
[7] S. Narulita, A. Nugroho, and M. Z. Abdillah, "Diagram Unified Modelling Language (UML) untuk Perancangan Sistem Informasi Manajemen Penelitian dan Pengabdian Masyarakat (SIMLITABMAS)," Bridge: Jurnal Publikasi Sistem Informasi Dan Telekomunikasi, vol. 2, no. 3, pp. 244–256, 2024.