Implementation of the Autokey Algorithm ASCII Text Cryptography Using JavaFX GUI

Ricky Dirgantara[1], Hans Valerian Lenice[2], L. Budi Handoko[3], Wise Herowati[4]
[1,2,3,4]Faculty of Computer Science, University of Dian Nuswantoro, Semarang, Indonesia

Article Info:

Keywords:

Autokey Cipher
Avalanche Effect
Character Error Rate
Cryptography
Entropy

ABSTRACT

The advance of information and communication technology has transformed the world in ways we could never have imagined. From the analog era to the digital age, technology has paved the way for new innovations and discoveries, changing the way we interact, communicate, work, and even play. Data breaches are one of the threats that frequently arise in this era of digital information since the data has great value in enhancing the security of the data we transmit, one of which is using code. There are numerous ways to encode data in cryptography, but one popular technique is Autokey Cipher cryptography. Autokey cipher is a development of vigenere cipher which is a solution to overcome the repetition of certain characters in keys. Autokey operations are performed based on the key length and calculated using the tabula recta of the vigenere cipher, the key from autokey is a combination of plaintext to produce a new key along the plaintext. In this research we built the implementation using Java library JavaFX. The GUI also shows how the autokey cipher works, which will give us an understanding about the encryption and decryption process that is built with ASCII text. And the other additions are the GUI that shows the calculations such as Avalanche Effect (AE), Bit Error Rate (BER), Character Error Rate (CER), and Entropy.

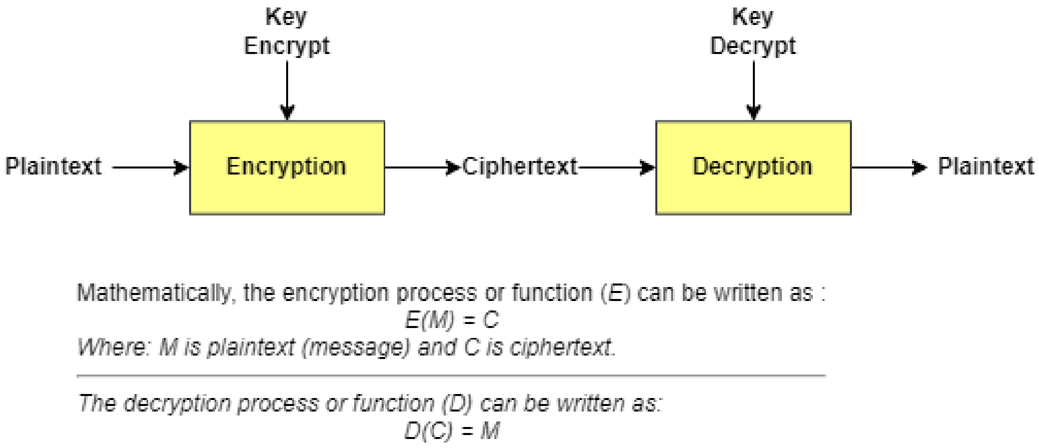
Author Correspondence:

L. Budi Handoko,
Faculty of Computer Science
University of Dian Nuswantoro
Email: handoko@dsn.dinus.ac.id

1. INTRODUCTION

Java is a programming language that is widely used for coding web applications. Java also supports a graphical user interface or GUI which is one of Java's main features. The main GUI we use is the Java library also known as JavaFX. GUI is a system of interactive visual components used in software, smartphones, and various other electronic devices [1], [2]. GUI are represented using several images which are referred to as GUI elements. With elements such as windows, icons and menus, computers can carry out commands such as opening, deleting, and moving files. One of the libraries provided by Java for building GUIs is JavaFX. JavaFX is a software platform for building Rich Internet Applications also known as RIA applications that can run on various devices. Such as desktop computers, web browsers on Windows, Linux and MacOS. Below is an overview of both processes in cryptography.

Cryptography is a process for disguising information so that it cannot be understood by other people, Cryptosystem is a method designed to carry out a cryptographic process while cryptanalysis is a process for re-disguising the results of a cryptosystem into understandable information [3], [4], [5], [6]. Plaintext can be said to be the original message/information that can still be understood or understood by humans, while ciphertext is a message that has been disguised so that it cannot be understood by humans [7], [8]. What about Encryption and Decryption? Encryption is the process of changing from plaintext to ciphertext while decryption is the process of changing ciphertext back to plaintext as shown in Figure 1.



Mathematically, the encryption process or function (*E*) can be written as :

E(M) = C

Where: M is plaintext (message) and C is ciphertext.

The decryption process or function (D) can be written as:

D(C) = M

Figure 1. Common Cryptography Scheme [9]

The main problem from this research that we made is to implement a graphical user interface that integrated with Autokey Algorithm to make it easy to use and understand for users [10], [11], [12], [13]. The GUI should support many text types and formats, and that why we used ASCII text to make it easier, and it will assist users that use our program to understand how an algorithm keyboard should operate automatically throughout a process. In addition, efficiency, and speed of the encrypt and decrypt processes are quite important for long and complex texts that are encoded with ASCII. In addition, the primary concern in this study is the security and privacy of user data during the encrypting and decrypting processes. The validation and verification process to ensure that the encrypt and decrypt functions are appropriate and crucial in this course are highly important [14], [15], [16]. In addition, the GUI needs to be simple to learn or modify to support other encrypt and decrypt algorithms in the future.

Here, we want to introduce to people about how the autokey cipher algorithm is implemented or works in a GUI using the Java programming language. In this journal we also explain the calculations, namely Avalanche Effect, Bit Error Rate, Character Error Rate and Entropy, so that people can better understand the performance of the autokey algorithm and examples of its use that we describe in the GUI implementation.

2. METHOD

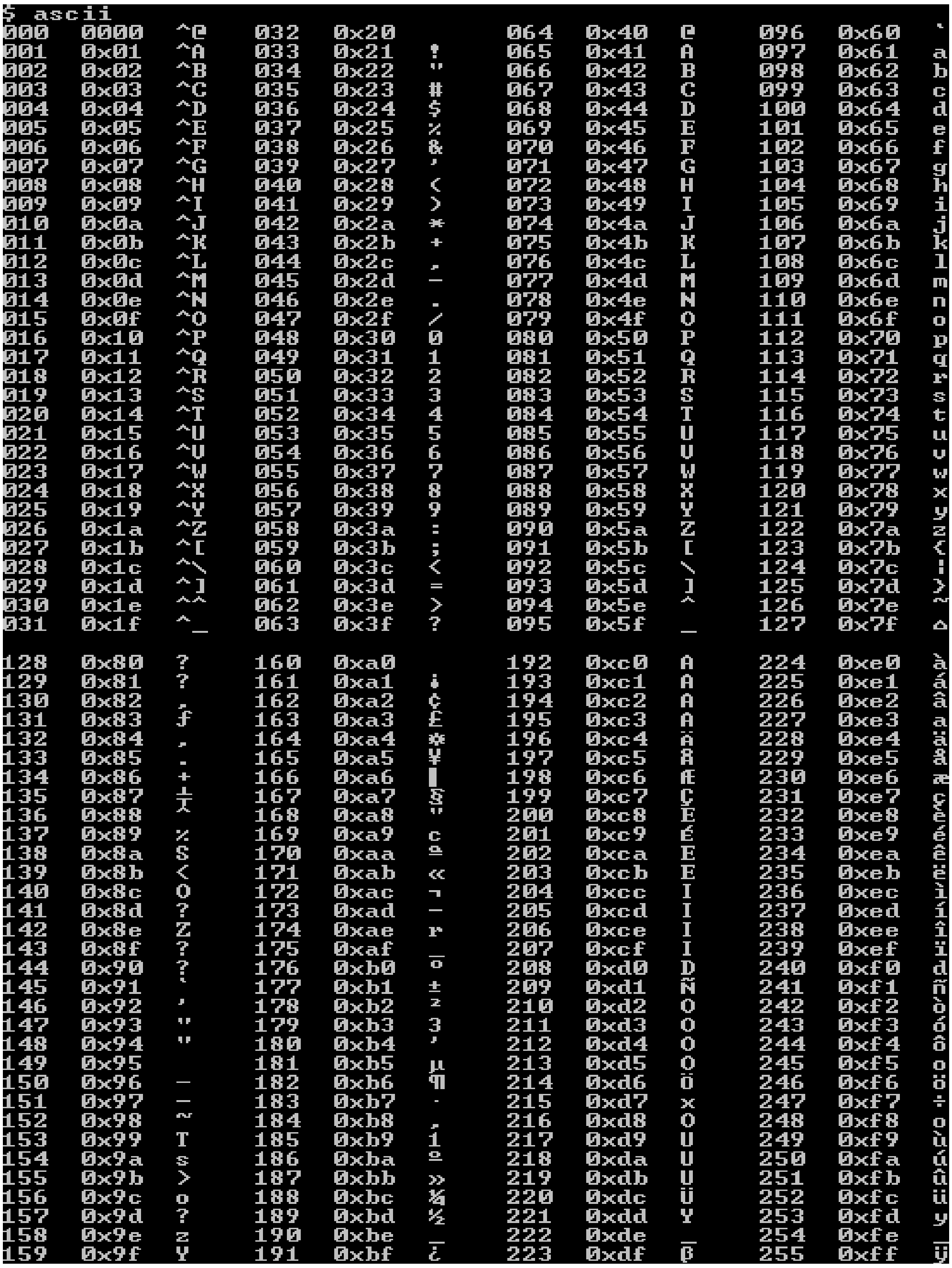
For this Research Method we started by searching for information about the AutoKey Cipher Algorithm and several tutorials for implementing it in a GUI using JavaFX. From these two things, we knew of the basic concepts of how AutoKey cryptography can be implemented in GUI-based applications. In this way we can carry out encryption and decryption as well as other calculations. After doing this, we also don't forget to carry out testing or tests regarding our application so that it is suitable and works correctly as we want.

2.1. Cryptography

Cryptography is an important component for maintaining data confidentiality. Cryptography plays an important role in the security and protection of network data because it allows the storage and transmission of sensitive data over unsecured networks, such as the Internet, so that it cannot be read by anyone except the person intended to receive it [3], [17], [18], [19], [20]. Cryptography Symmetric is a type of cryptography, where there is only one key to encrypt and decrypt from the information. Meanwhile, cryptography asymmetric is a type of cryptography that uses two keys: public key that can be shared to public and a private key that is only known by the data owner [21], [22], [23]. Further information about the public key is that it can be shared with anyone, but for the private key it must be kept confidential. If someone has your public key and they can encrypt your messages but only you with the private key can decrypt them.

2.2. ASCII

ASCII is based on the simple idea of using numbers to represent text, for example numbers that represent letters "A", "q", or "!", collectively letters, numbers and symbols are called characters [24]. By signing a number to each of characters in a block of text, the block of text can be represented entirely by numbers and that easily stored in the computer memories, for example letter "A" is always assigned the number '065' conversely '065' always reverse for letter "A". ASCII is an english language subsets of Unicode meaning that all of ascii exists within the Unicode. ASCII is also commonly used in computer science particularly in programming language, text files and data conversions as displayed in Figure 2.



```
$ ascii
000   0000   ^@    032   0x20         064   0x40   @     096   0x60   `
001   0x01   ^A    033   0x21   !     065   0x41   A     097   0x61   a
002   0x02   ^B    034   0x22   "     066   0x42   B     098   0x62   b
003   0x03   ^C    035   0x23   #     067   0x43   C     099   0x63   c
004   0x04   ^D    036   0x24   $     068   0x44   D     100   0x64   d
005   0x05   ^E    037   0x25   %     069   0x45   E     101   0x65   e
006   0x06   ^F    038   0x26   &     070   0x46   F     102   0x66   f
007   0x07   ^G    039   0x27   '     071   0x47   G     103   0x67   g
008   0x08   ^H    040   0x28   (     072   0x48   H     104   0x68   h
009   0x09   ^I    041   0x29   )     073   0x49   I     105   0x69   i
010   0x0a   ^J    042   0x2a   *     074   0x4a   J     106   0x6a   j
011   0x0b   ^K    043   0x2b   +     075   0x4b   K     107   0x6b   k
012   0x0c   ^L    044   0x2c   ,     076   0x4c   L     108   0x6c   l
013   0x0d   ^M    045   0x2d   -     077   0x4d   M     109   0x6d   m
014   0x0e   ^N    046   0x2e   .     078   0x4e   N     110   0x6e   n
015   0x0f   ^O    047   0x2f   /     079   0x4f   O     111   0x6f   o
016   0x10   ^P    048   0x30   0     080   0x50   P     112   0x70   p
017   0x11   ^Q    049   0x31   1     081   0x51   Q     113   0x71   q
018   0x12   ^R    050   0x32   2     082   0x52   R     114   0x72   r
019   0x13   ^S    051   0x33   3     083   0x53   S     115   0x73   s
020   0x14   ^T    052   0x34   4     084   0x54   T     116   0x74   t
021   0x15   ^U    053   0x35   5     085   0x55   U     117   0x75   u
022   0x16   ^V    054   0x36   6     086   0x56   V     118   0x76   v
023   0x17   ^W    055   0x37   7     087   0x57   W     119   0x77   w
024   0x18   ^X    056   0x38   8     088   0x58   X     120   0x78   x
025   0x19   ^Y    057   0x39   9     089   0x59   Y     121   0x79   y
026   0x1a   ^Z    058   0x3a   :     090   0x5a   Z     122   0x7a   z
027   0x1b   ^[    059   0x3b   ;     091   0x5b   [     123   0x7b   {
028   0x1c   ^\    060   0x3c   <     092   0x5c   \     124   0x7c   ¦
029   0x1d   ^]    061   0x3d   =     093   0x5d   ]     125   0x7d   }
030   0x1e   ^^    062   0x3e   >     094   0x5e   ^     126   0x7e   ~
031   0x1f   ^_    063   0x3f   ?     095   0x5f   _     127   0x7f   ⌂

128   0x80   ?     160   0xa0         192   0xc0   A     224   0xe0   à
129   0x81   ?     161   0xa1   ¡     193   0xc1   A     225   0xe1   á
130   0x82   ‚     162   0xa2   ¢     194   0xc2   A     226   0xe2   â
131   0x83   ƒ     163   0xa3   £     195   0xc3   A     227   0xe3   a
132   0x84   „     164   0xa4   ¤     196   0xc4   Ä     228   0xe4   ä
133   0x85   .     165   0xa5   ¥     197   0xc5   Å     229   0xe5   å
134   0x86   +     166   0xa6   ¦     198   0xc6   Æ     230   0xe6   æ
135   0x87   ‡     167   0xa7   §     199   0xc7   Ç     231   0xe7   ç
136   0x88   ˆ     168   0xa8   ¨     200   0xc8   Ë     232   0xe8   è
137   0x89   %     169   0xa9   c     201   0xc9   É     233   0xe9   é
138   0x8a   S     170   0xaa   ª     202   0xca   E     234   0xea   ê
139   0x8b   <     171   0xab   «     203   0xcb   E     235   0xeb   ë
140   0x8c   O     172   0xac   ¬     204   0xcc   I     236   0xec   ì
141   0x8d   ?     173   0xad   -     205   0xcd   I     237   0xed   í
142   0x8e   Z     174   0xae   r     206   0xce   I     238   0xee   î
143   0x8f   ?     175   0xaf   ¯     207   0xcf   I     239   0xef   ï
144   0x90   ?     176   0xb0   °     208   0xd0   D     240   0xf0   d
145   0x91   ‘     177   0xb1   ±     209   0xd1   Ñ     241   0xf1   ñ
146   0x92   ’     178   0xb2   ²     210   0xd2   O     242   0xf2   ò
147   0x93   "     179   0xb3   ³     211   0xd3   O     243   0xf3   ó
148   0x94   "     180   0xb4   ´     212   0xd4   O     244   0xf4   ô
149   0x95         181   0xb5   µ     213   0xd5   O     245   0xf5   o
150   0x96   –     182   0xb6   ¶     214   0xd6   Ö     246   0xf6   ö
151   0x97   —     183   0xb7   ·     215   0xd7   x     247   0xf7   ÷
152   0x98   ~     184   0xb8   ¸     216   0xd8   O     248   0xf8   o
153   0x99   T     185   0xb9   ¹     217   0xd9   U     249   0xf9   ù
154   0x9a   s     186   0xba   º     218   0xda   U     250   0xfa   ú
155   0x9b   >     187   0xbb   »     219   0xdb   U     251   0xfb   û
156   0x9c   o     188   0xbc   ¼     220   0xdc   Ü     252   0xfc   ü
157   0x9d   ?     189   0xbd   ½     221   0xdd   Y     253   0xfd   y
158   0x9e   z     190   0xbe   _     222   0xde   _     254   0xfe   _
159   0x9f   Y     191   0xbf   ¿     223   0xdf   ß     255   0xff   ÿ
```

Figure 2. ASCII Table

2.3. Autokey Cipher

Autokey in cryptography refers to a method of encryption where the key used in the encryption process is based on the original text. In the autokey method, the original text serves as the key, and this key is automatically or cyclically applied to the text to produce the encrypted text. This method falls under the category of symmetric-key algorithms as shown in equation (1) and (2).

$$Ci = (Pi + Ki)\ mod\ 26 \qquad (1)$$

Where :

C is Result of encrypted text (Ciphertext) from i
P is Plainteks from i
K is Key from i

The encryption process involves converting the text's original length (Pi) to the text's original length (Ki), and the result is expressed in modulo (M).

$$Pi = (Ci + Ki) \; mod \; 26 \qquad (2)$$

Where :
P is Plaintext from i
C is Result of encrypted text (Ciphertext) from i
K is Key from i

The Decryption process involves reversing by subtracting the value of the encrypted text (C_i) from the value of the key (Ki), and the result expressed in modulo (M).

2.4. Avalanche Effect (AE)

The Avalanche Effect is very important because it ensures that even small changes in the autokey algorithm will produce the same output, this ensures that attackers cannot easily predict the original text through statistical analysis. This happens because in an auto-key algorithm, a small change in the original text or key must result in a significant change in the encrypted text.

$$AE = \frac{Total\ different\ bits\ from\ plaintext\ and\ ciphertext}{total\ characters} x\ 100\% \qquad (3)$$

This equation above calculates the proportion of the encrypted text that changes when the original text or key is slightly changed.

2.5. Bit Error Rate (BER)

Bit Error Rate (BER) is a measure of telecommunication signal integrity based on the quantity or percentage of transmitted bits that are received incorrectly. Essentially, the more incorrect bits, the greater the impact on signal quality. Bit error rate is an effective indicator of full end-to-end performance because it encompasses the receiver and transmitter as well as the media between them.

$$BER = \frac{Total\ different\ bits}{total\ bits} x\ 100\% \qquad (4)$$

The equation above shows us how to calculate BER by the percentage difference in bits between the encrypted text and the plaintext.

2.6. Character Error Rate (CER)

Character Error Rate is to calculate percentage levels the accuracy of an encryption results by matching and comparing characters (letters, numbers, symbols). plaintext has compared to plaintext that is added or changed with a lower presentation.

$$CER = \frac{Total\ different\ bits\ from\ plaintext\ and\ ciphertext}{total\ characters} x\ 100\% \qquad (5)$$

The equation above shows us how to calculate CER by the percentage of character differences between the generated encrypted text and the expected text.

2.7. Entropy

Entropy is a measure of the amount of information in a message or text and is expressed in units of bits. Entropy is used to estimate the average number of bits to encode each element of the message. The higher the number of bits of entropy, the greater the complexity and the level of security in a cryptographic system. In general, entropy can be calculated using equation (6).

$$H(X) = -\sum_{i=i}^{n} a_i{}^2 log(p(S_i)) \quad (6)$$

Where :
X is message
Si is the ith symbol in the message p(S) = the chance of occurrence of S i
a is number of occurrences of S i

2.8. Proposed Method

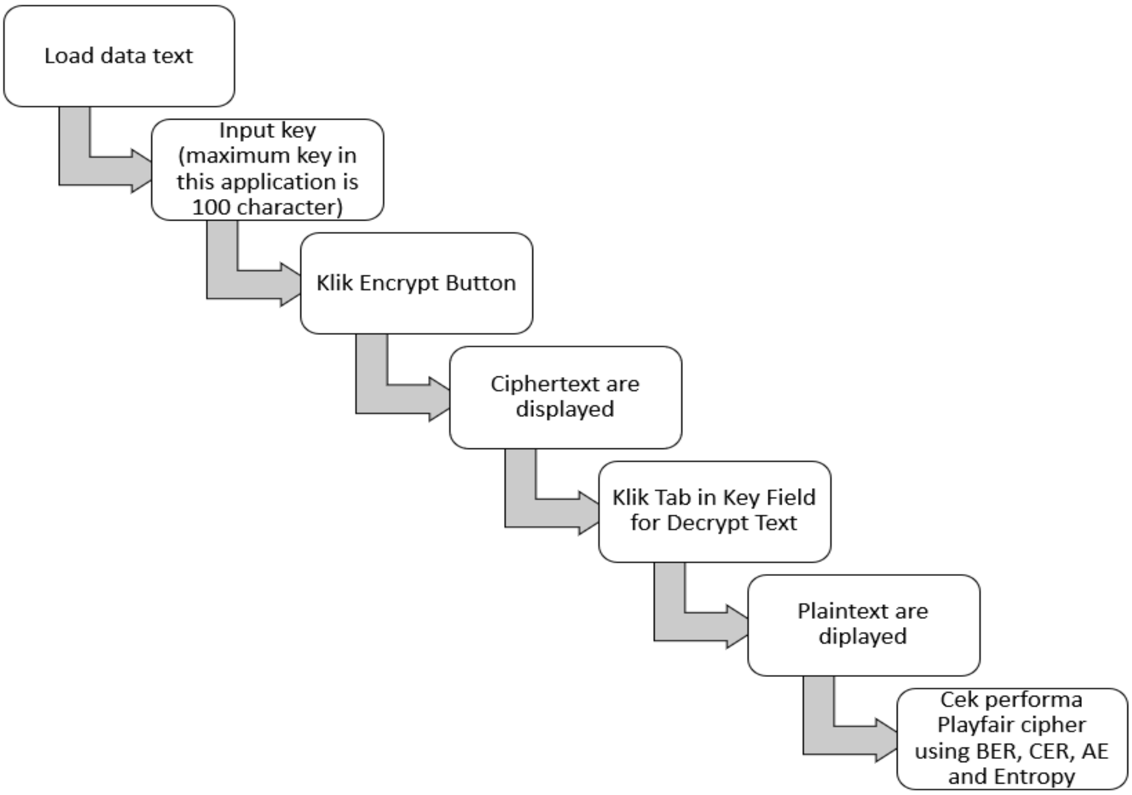

Load data text
Input key (maximum key in this application is 100 character)
Klik Encrypt Button
Ciphertext are displayed
Klik Tab in Key Field for Decrypt Text
Plaintext are diplayed
Cek performa Playfair cipher using BER, CER, AE and Entropy

Figure 3. Proposed method based on Playfair Cipher

3. RESULTS AND DISCUSSION

This research is implemented using the Java programming language with JavaFX as its user interface, and it is executed using the IntelliJ IDEA software. The following is the GUI display when it is run, as shown in Figure 4.

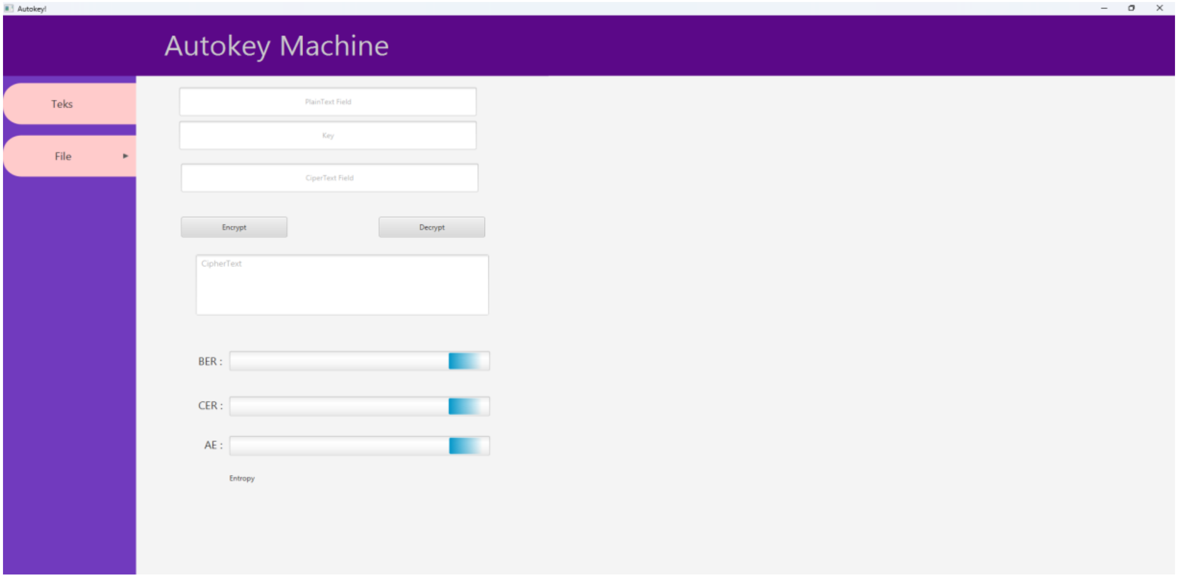
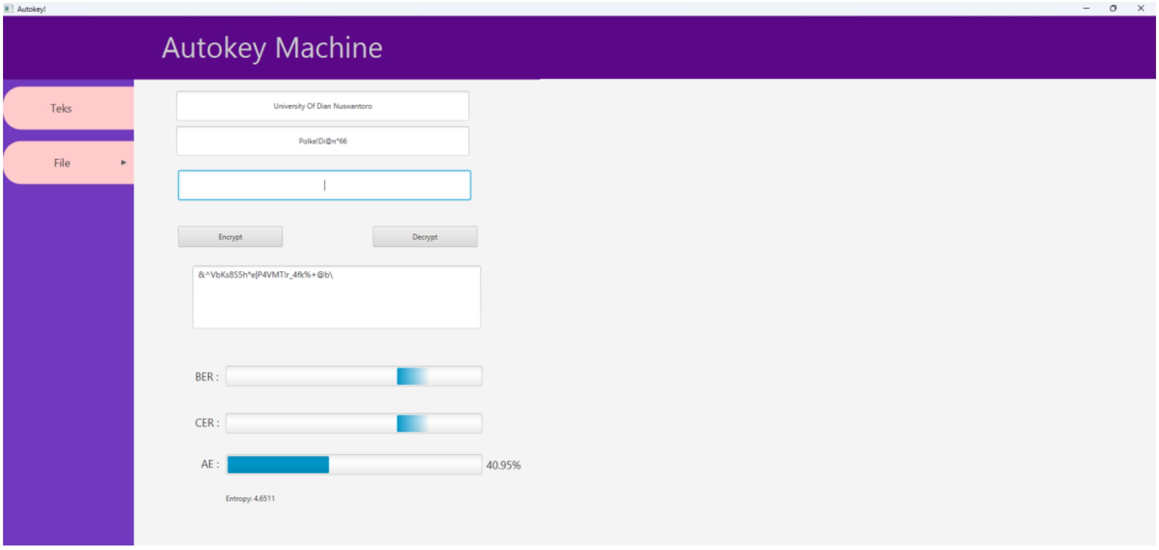


Figure 4. Display result

Figure 4 shows the initial view before inputting the plaintext and key; the Avalanche Effect, Character Error Rate, Bit Error Rate and Entropy item is not yet given results because the empty fields in the plaintext, key, and ciphertext and need to be filled in and entered first. After entering the plaintext and key/ciphertext in the provided columns, as shown in the display in Figure 4, and pressing the encrypt/decrypt button, the display will be complete. Once data security has been implemented by entering the plaintext and key/ciphertext, it is ensured that when executed by pressing the encryption/decryption button, the output results will be displayed as shown in Figure 5.



University Of Dian Nuswantoro

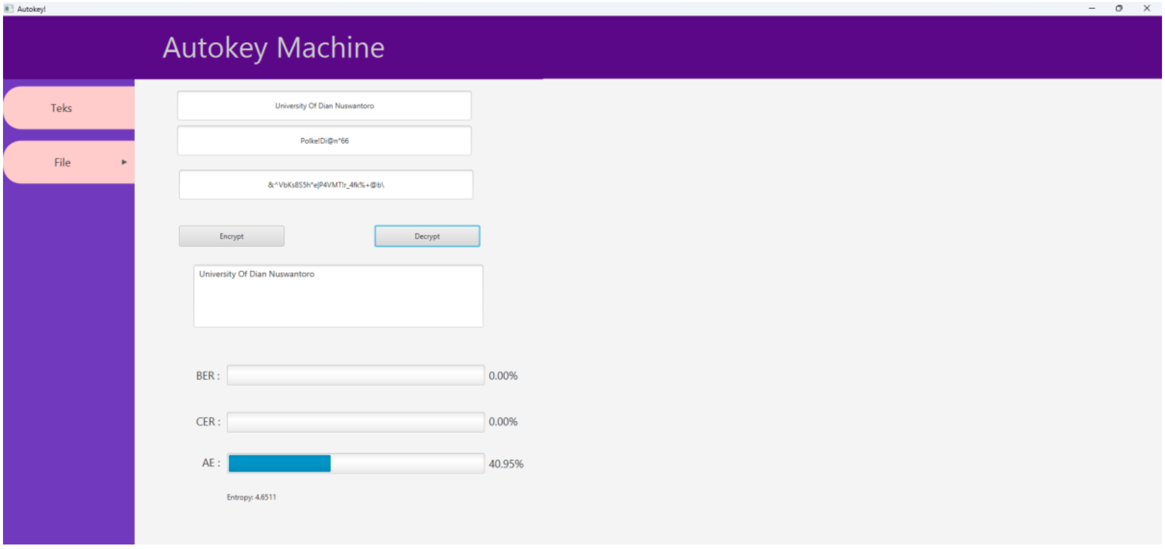
Polke!Di@n*66

&^VbKs8S5h*e|P4VMT!r_4fk%+@b\

40.95%

Entropy: 4.6511

Figure 5. Encrypt Output Result



University Of Dian Nuswantoro

Polke!Di@n*66

&^VbKs8S5h*eJP4VMT!r_4fk%+@b\

University Of Dian Nuswantoro

BER : 0.00%

CER : 0.00%

AE : 40.95%

Entropy: 4.6511

Figure 6. Decrypt Output Result

Figure 5 shows the encrypt output results when entering the plaintext and key. Meanwhile Figure 6, the decryption output displays the results when entering the same key and alo the results of the ciphertext from Figure 5, including the original text, encrypted text, decrypted text, Bit Error Rate (BER), Character Error Rate (CER), Avalanche Effect (AE), and Entropy.

Table 1. Test Results with Autokey Method

Plaintext	Key	Ciphertext
Universitas Dian Nuswantoro	Polke!Di@n*66	&^VbKs8S5P}6Z:Q[k4v8a"]~&)@
Testing number 45 with Autokey Method	Polke!Di@n*66	%U``Oo,i/dwx{Co!!ex.^)nK,+@[ReeN*^)^n
RickyDirgantara@mhs	Polke!Di@n*66	#YPW_E.\(Px+wCQ-YNt
HansVal3r1An_Lenice	Polke!Di@n*66	xQ[_<b1|3 K%u|U[UIf
Autokey_Ricky_hans@mhs.dinus!ac	Polke!Di@n*66	qea[Qf>IrXm"00XNZYA2R4|n %FcmMI
<?R1ckY_W1th_H4nS??	Polke!Di@n*66	l/?|Il}Iw ~~ux$[?%@

Based on Table 1, it had been seen that the autokey method can be applied to ASCII text with a combination of letters, numbers, and symbols, resulting in encrypted text or ciphertext in the form of symbols based on the ASCII table. Next, tests were conducted for Avalanche Effect, BER, CER, and Entropy with results as shown in Table 2, Table 3, and Figure 7.

Table 2. Avalanche Test Results

Plaintext	Key	Avalanche Effect %
Universitas Dian Nuswantoro	Polke!Di@n*66	43%
Testing number 45 with Autokey Method	Polke!Di@n*66	39%
RickyDirgantara@mhs	Polke!Di@n*66	42%
HansVal3r1An_Lenice	Polke!Di@n*66	37%
Autokey_Ricky_hans@mhs.dinus!ac	Polke!Di@n*66	38%
<?R1ckY_W1th_H4nS??	Polke!Di@n*66	36%

Based on Table 2 above, the average Avalanche Effect value from the 6 conducted tests is 39%. This means that the Avalanche Effect generated using the Autokey method is considered quite good. The use of ASCII text with a range of letters, numbers, and symbols enhances the level of information security derived from the use of the Autokey method. However, the Avalanche Effect value is considered quite good depending on the context and desired security requirements. In cryptography, the higher the avalanche effect is the better it gets. That means a small change in the original text or key should result in a significant change in the encrypted text.

Table 3 Bit Error Rate (BER) and Character Error Rate (CER) Test Result

Plaintext	Key	Bit Error Rate %	Character Error Rate %
Universitas Dian Nuswantoro	Polke!Di@n*66	0%	0%
Testing number 45 with Autokey Method	Polke!Di@n*66	0%	0%
RickyDirgantara@mhs	Polke!Di@n*66	0%	0%
HansVal3r1An_Lenice	Polke!Di@n*66	0%	0%
Autokey_Ricky_hans@mhs.dinus!ac	Polke!Di@n*66	0%	0%
<?R1ckY_W1th_H4nS??	Polke!Di@n*66	0%	0%

Table 3 shows us the Bit Error Rate and Character Error Rate values. The BER and CER values for each test are 0, which indicates that the encryption test has a high level of accuracy. Tests with high BER and CER values have lower CER values which indicates that the resulting ciphertext is better along with lower bit and character error rates. When evaluating algorithms for encryption or decryption, it is very important to pay attention and understand about the Character Error Rate and Bit Error Rate values. These values will show us how well the algorithm can produce precise and accurate encrypted text.

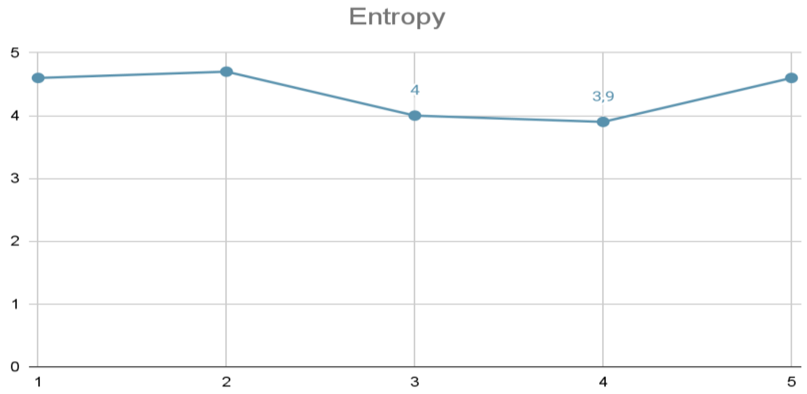


Figure 7. Entropy Test Result

Figure 7 shown that the average of the entropy test result obtained is 4 and if we look at the entropy value, it may be reasonable for randomness, but keep in mind that this value still depends on you or the final assessment set in the system. We now can understand if the high entropy results can be obtained if the text has a lot of long texts, numbers and symbols or complex randomness, and for low entropy results are text that has few short letters, numbers, and symbols.

4. CONCLUSION

This research was created using the Autokey Algorithm and the programming language used was Java and the library used from Java was JavaFX. The GUI that we created in this research has several features that have been created such as input plaintext fields, key fields, and ciphertext fields, other features are the results of input created such as ciphertext (results from plaintext encryption) and calculations that include things like Avalanche Effect, Character Error rate, Bit Error Rate, and Entropy Calculation. Apart from that, the IDE used to help run the GUI is Intellij IDE from Jetbrains. The purpose of this research is that the cryptography implementation created can be easily understood by users who use it. We hope that by making this research we can understand that cryptography is no longer used only for writing or completing code, but has developed into various functions such as encryption or decryption for data security, identity and message authentication, digital signatures and certificates, key exchange protocols, digital money, and so on.

REFERENCES

[1] Md. S. Hossain Biswas *et al.*, “A systematic study on classical cryptographic cypher in order to design a smallest cipher,” *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 12, p. p9662, Dec. 2019, doi: 10.29322/ijsrp.9.12.2019.p9662.

[2] R. D. Ainul, S. Wibowo, and I. Zaini, “An Encrypted QR Code Using Layered Numeral Calculation for Low Powered Devices,” in *Proceedings of the 4th International Conference on Informatics, Technology andEngineering 2023 (InCITE 2023)*, 2023, pp. 607–617. doi: 10.2991/978-94-6463-288-0_50.

[3] S. M. Suhael, Z. A. Ahmed, and A. J. Hussain, “Proposed Hybrid Cryptosystems Based on Modifications of Playfair Cipher and RSA Cryptosystem,” *Baghdad Science Journal*, May 2023, doi: 10.21123/bsj.2023.8361.

[4] J. P. Sermeno, K. A. S. Secugal, and N. E. Mistio, “Modified Vigenere cryptosystem: An integrated data encryption module for learning management system,” *International Journal of Applied Science and Engineering*, vol. 18, no. 4(Special Issue), pp. 1–10, 2021, doi: 10.6703/IJASE.202106_18(4).003.

[5] O. F. AbdelWahab, A. I. Hussein, H. F. A. Hamed, H. M. Kelash, and A. A. M. Khalaf, “Efficient Combination of RSA Cryptography, Lossy, and Lossless Compression Steganography Techniques to Hide Data,” *Procedia Comput Sci*, vol. 182, pp. 5–12, 2021, doi: 10.1016/j.procs.2021.02.002.

[6] K. R. Ilaga and C. A. Sari, “Analysis of Secure Image Crypto-Stegano Based on Electronic Code Book and Least Significant Bit,” *Journal of Applied Intelligent System*, vol. 3, no. 1, pp. 28–38, 2018.

[7] K. Prasad and H. Mahato, “Cryptography using generalized Fibonacci matrices with Affine-Hill cipher,” *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 25, no. 8, pp. 2341–2352, Nov. 2022, doi: 10.1080/09720529.2020.1838744.

[8] C. A. Sari, M. H. Dzaki, E. H. Rachmawanto, R. R. Ali, and M. Doheir, “High PSNR Using Fibonacci Sequences in Classical Cryptography and Steganography Using LSB,” *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 4, pp. 568–580, 2023, doi: 10.22266/ijies2023.0831.46.

[9] I. Gede, A. Putra Dewangga, T. W. Purboyo, and R. A. Nugrahaeni, “A New Approach of Data Hiding in BMP Image Using LSB Steganography and Caesar Vigenere Cipher Cryptography,” 2017. [Online]. Available: http://www.ripublication.com

[10] L. B. Handoko and C. Umam, “Data Security Using Color Image Based on Beaufort Cipher, Column Transposition and Least Significant Bit (LSB)id 2 *Corresponding author,” 2023.

[11] T. M. Aung, H. H. Naing, and N. N. Hla, “A complex transformation of monoalphabetic cipher to polyalphabetic cipher: (Vigenère-Affine Cipher),” *Int J Mach Learn Comput*, vol. 9, no. 3, pp. 296–303, Jun. 2019, doi: 10.18178/ijmlc.2019.9.3.801.

[12] L. Budi Handoko and A. Rizqy, “File Cryptography Optimization Based on Vigenere Cipher and Advanced Encryption Standard (AES),” 2023.

[13] S. K. Lakshmanan, L. Shakkeera, and V. Pandimurugan, “Efficient Auto key based Encryption and Decryption using GICK and GDCK methods,” in *Proceedings of the 3rd International Conference on Intelligent Sustainable Systems, ICISS 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 1102–1106. doi: 10.1109/ICISS49785.2020.9316114.

[14] E. Y. Purba, S. Efendi, P. Sirait, and P. Sihombing, “Collaboration of RSA Algorithm Using EM2B Key with Word Auto Key Encryption Cryptography Method in Encryption of SQL Plaintext Database,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Sep. 2019. doi: 10.1088/1742-6596/1230/1/012009.

[15] B. Deepa, V. Maheswari, and V. Balaji, “An Efficient Cryptosystem Using Playfair Cipher Together with Graph Labeling Techniques,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jul. 2021. doi: 10.1088/1742-6596/1964/2/022016.

[16] P. Pristiwanto, H. Sunandar, and B. Nadeak, “Analysis and Implementation of PlayFair Chipper Algorithm in Text Data Encoding Process,” *Jurnal Info Sains : Informatikan dan Sains*, vol. 1, no. 1, pp. 19–23, 2020, [Online]. Available: http://ejournal.seaninstitute.or.id/index.php/InfoSains

[17] M. S. Yousif, R. K. Salih, and N. M. G. Alsaidi, “A new modified playfair cipher,” in *AIP Conference Proceedings*, American Institute of Physics Inc., Apr. 2019. doi: 10.1063/1.5095132.

[18] N. A. Kako, H. T. Sadeeq, and A. R. Abrahim, “New symmetric key cipher capable of digraph to single letter conversion utilizing binary system,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 2, pp. 1028–1034, May 2020, doi: 10.11591/IJEECS.V18.I2.PP1028-1034.

[19] A. C. Licayan, B. D. Gerardo, and A. A. Hernandez, “Enhancing Playfair Cipher using Seed Based Color Substitution,” in *Proceedings, 2020 16th IEEE International Colloquium on Signal Processing & its Application (CSPA 2020)*, 2020, pp. 242–246.

[20] P. Pristiwanto, H. Sunandar, and B. Nadeak, “Analysis and Implementation of PlayFair Chipper Algorithm in Text Data Encoding Process,” *Jurnal Info Sains : Informatikan dan Sains*, vol. 1, no. 2, pp. 19–23, 2020, [Online]. Available: http://ejournal.seaninstitute.or.id/index.php/InfoSains

[21] S. M. Suhael, Z. A. Ahmed, and A. J. Hussain, “Proposed Hybrid Cryptosystems Based on Modifications of Playfair Cipher and RSA Cryptosystem,” *Baghdad Science Journal*, May 2023, doi: 10.21123/bsj.2023.8361.

[22] D. Babuc, “An Authentic Algorithm for Ciphering and Deciphering Called Latin Djokovic,” *International Journal of Computer and Information Engineering*, pp. 17–10, 2023, [Online]. Available: https://www.researchgate.net/publication/374589454

[23] E. Hari Rachmawanto *et al.*, “Testing Data Security Using a Vigenere Cipher Based on the QR Code,” *Computer Network, Computing, Electronics, and Control Journal*, vol. 8, no. 4, pp. 701–708, 2023, doi: 10.22219/kinetik.v8i4.1734.

[24] Permana langgeng wicaksono ellwid putra, C. A. Sari, and F. O. Isinkaye, “Secure Text Encryption for Iot Communication Using Affine Cipher And Diffie-Hellman Key Distribution On Arduino Atmega2560 IotT Devices,” *Jurnal Teknik Informatika (Jutif)*, vol. 4, no. 4, pp. 849–855, Aug. 2023, doi: 10.52436/1.jutif.2023.4.4.1129.