ENHANCING COMPUTATIONAL TIME FOR A 16X16 PLAYFAIR MATRIX FOR UNICODE CHARACTERS

LAWAL, I.*, AHMAD, B.I. AND ABDULLAHI, F.B.
Department of Computer Science, Ahmadu Bello University, Zaria, Nigeria

ABSTRACT

Playfair cipher is one of the most popular poly-alphabetic ciphers that can be used suitably for wireless and mobile devices where security requirements are high and system resources are low. The existing work uses a 16x16 playfair matrix for Unicode characters. The 16x16 matrix is used for characters' representation and a shuffling technique is applied after each character encryption. However, the shuffling technique used and character search in the 16x16 matrix after each character encryption leads to a high computation time. In order to overcome the shuffling and character research of the existing 16 x 16 algorithm, this work used shifting keys to search for character sand shuffle the entire matrix without changing the position of each element in the matrix unlike in the existing work. This paper shows that a 17x17 matrix algorithm performed better in terms of encryption time by 1.29972 seconds and decryption time by 2.76146 seconds on an average. Therefore, an increase in memory to form a 17x17 matrix has a negligible impact on the enhanced algorithm. Also, the enhanced algorithm transforms the plaintext into Chinese characters. These Chinese characters (ciphertext) are seen as plaintext which translates into something meaningful in order to preserve the integrity of the original plaintext and to fend-off suspicious attacker sun like in the previous study where the ciphertext generated are random characters without any meaning. Hence, the enhanced algorithm provides a better way of hiding information.

Keywords: Chinese character, ciphertext, hexadecimal, plaintext.
***Correspondence:** lawibrahim3@gmail.com

INTRODUCTION

The Playfair cipher was developed for telegraph secrecy and was the first digraph substitution cipher. It was invented by Sir Charles Wheatstone in 1854, but he named it after his friend Lyon Playfair who was a scientist and a public figure of Victorian England. It was used by the British forces in both the Boer War and World War I and also by the Australians in World War II Bhattacharyya *et al*. [1]. The Playfair system in its easiest form uses a 5 by 5 alphabet matrix. Not only is playfair interesting, but it is also suitable for the security of wireless and mobile systems [2].

The advent of digital encryption devices has rendered the traditional Playfair unsecured. This is simply because modern computers could easily break the cipher within seconds using Brute Force and Frequency Analysis [3]. For this reason, several authors have modified the traditional playfaircipher algorithm by adding various techniques to make it strong and to be able to represent many characters.

Related works

Several authors have worked and improved on the traditional playfair to combat the issues facing it. Some of which include: Lahiri [4] who came up with a new approach which treats each file as a binary file and applies the playfair algorithm on each byte of the file. The nibbles of each byte are used to encrypt / decrypt with the help of a reduced 4×4 reference key matrix. A nibble consists of 4 bits having a value of range 0-15. On the other hand, the 4×4 reference key matrix also contains 16 values, in the range0-15. So, each pair of the nibbles of a byte is replaced with a new pair of nibbles and thus new byte is obtained. This algorithm encrypts / decrypts the file byte wise and odd length word problem does not arise. It implements rotation after each encryption of a byte. Some of the limitations includes: its inability to represent special characters and it also lacks a larger key matrix which can be used to enhance the security. It supports only 8 bit characters.

The paper Enhanced Cryptographic Scheme (NPSC) proposed by Masadehs *et al* [5], which is inspired by Playfair cipher encrypts alphanumeric messages. It adopts an elaborate key creation method and consists of two encryption/decryption algorithms and relies on modular arithmetic calculation for key generation and cryptographic processes. Key length must be either 4 or 9 or 16. Two 5x5 matrices were employed as the backbone for this scheme, one for the alphabetic characters and the other for numerals. Thei/j characters are considered the same. It could not represent special characters and can only support 25 characters.

The paper proposed by Ahnaf and Rabiul [6], extended the character support and additional features. Primarily, the 7x7 matrix supports 49 characters. But, this model uses 47 of them for general purpose and 2 for special purpose. The character set includes 26 lower-case letters, 10 numerals, 10 most frequently used punctuation marks and a whitespace character. The two remaining characters "!" and "~" serve exclusively as a filler character and a padding character. This two particular character are not eligible to participate in plaintext or keyword. During decryption they are omitted. They eliminate the existing ambiguity in playfair of using x character as a filler and for replacing double character in the plaintext. This cipher supports only 49 characters and the"!" and "~" characters cannot be used in a plaintext since its meaning has been redefined.

According to Ahmed *et al.* [7], this algorithm can support all language scripts in the world; for this Unicode is being used and it requires a 256X256 matrix. This is because the value of any character of any language around the world is between (0 - 65536). This way the user can encrypt any language including Kurdish language, space, symbols, special characters, etc. Next, a hash function of the key will be generated using Sha512 technique and then it will be merged with the cipher. This will be used to verify the Authenticity of the user and prevent the dictionary attack. The matrix size is too large for mobile devices. High computational time will be required which is not suitable for mobile devices.

Similarly, Nuhu *et al.* [8] proposed, the size of the matrix used for Unicode characters was reduced from 256x256 to 16x16. The identity of a particular language that is used for encryption was hidden. Also, it performed swapping of hexadecimal characters, shuffling of a 16x16 matrix after each encryption of a character and padding of zeros to the hexadecimal characters to be encrypted. High computation time in both encryption/decryption process and character search in the entire 16x16 matrix. Memory utilization was not fully taken into consideration by padding the hexadecimal characters with zeros and shuffling the 16x16 matrix. Therefore, this is not suitable for mobile devices with low system resources.

The enhanced method

The enhanced algorithm takes in the plaintext and user's key. A matrix of 16 x 16 Chinese randomly selected single characters is created. The user's key which is a top row of characters A-F and numbers 0-9 is added to the 16 x 16 matrix to form a 17x17 matrix. The top row is from 0-9 and A-F, it shifts from left to right by a number/alphabet each time after an encryption takes place. The first character shifts to the position of the last character and the second character becomes the first character and so on. Shifting of the key characters shuffles the entire 16x16 matrix of the Chinese characters. Also, searching of the hexadecimal characters will be done in the key row and column unlike in the existing work where the searching is done in the entire 16x16 matrix. The first column of the matrix which is the column key is arranged from 0-9 and A-Z from bottom up. These keys do not shift continuously like the top row keys but rather it is static at that position.

Enhanced Encryption algorithm

1. // input : plaintext P, keyword K
2. // output: ciphertext C
3. Begin
4. P = user plaintext
5. K= user keyword
6. Foreach k in K
7. get hexadecimal equivalent of k
8. end foreach
9. K^1 = modified K
10. Create 16x16 matrix of Chinese characters
11. Add a top row and first column of key characters (0 -9 and A –F) specified by the user.
12. Foreach p in P
13. get hexadecimal equivalent of p
14. End foreach
15. P^1 = modified P
16. If length of P^1 is even
17. get diagraphs of P^1
18. Else
19. Add null to P^1
20. get diagraphs of P^1
21. End if
22. End foreach
23. Foreach diagraph in P^1
24. Swap first nibble in first character with second nibble in second character
25. Swap second nibble in first character with first nibble in second character
26. End foreach
27. P^{11} = modified P^1
28. Foreach current byte character in P^{11}
29. Apply row – column substitution of nibbles on the 16x16 matrix to give ciphertext C.
30. Shift the first row key character.
31. End.

Enhanced Decryption algorithm

1. // input : ciphertext C, keyword K
2. // output: plaintext P
3. Begin
4. P = user plaintext
5. K= user keyword
6. Foreach k in K
7. get hexadecimal equivalent of k
8. end foreach
9. K^1 = modified K
10. Create 16x16 matrix of Chinese characters
11. Add a top row and first column of key characters (0 -9 and A –F) specified by the user.
12. foreach c in C
13. get hexadecimal equivalent of c
14. End foreach
15. C^1 = modified C
16. Foreach diagraph of characters in C^1
17. Swap first nibble in first character with second nibble in second character
18. Swap second nibble in first character with first nibble in second character
19. End foreach
20. C^{11} = modified C^1
21. Foreach byte character in C^{11}
22. Apply column – row substitution of nibbles on the 16x16 matrix to give plaintext P.
23. Shift the first row key character.
24. End of foreach
25. C^{111} = modified C^{11}
26. Foreach of c in C^{111}
27. Get the byte character equivalent of c to give the plaintext P
28. End foreach
29. End.

For 8 bit characters, the plaintext length must be even otherwise if it is odd a null value is added at the end of the plaintext. The plaintext is divided into two pairs and converted to its hexadecimal equivalent. For 16 bit characters, a character is converted to its hexadecimal equivalent. The first nibble of the first character is swapped with the last nibble of the second character and the last nibble of the first hexadecimal character is swapped with the first nibble of the second hexadecimal character. The process continues till all the characters are exhausted to form modified hexadecimal characters. Apply a substitution of a column and row intersection of the Chinese character. The Chinese character gotten as a result of the intersection becomes the ciphertext of that character. Shifting of keys will happen before the next encryption. This process continues till all the characters get encrypted. Note that the 16x16 Chinese characters are static while in the existing work it is shuffled after each encryption of a character. To decrypt the ciphertext, construct the 16x16 matrix Chinese matrix and place the top row and first column keys. Take each Chinese character at a time, locate the Chinese character in the matrix and get the corresponding row and column values (this values forms the modified hexa decimal characters). Divide the modified hexadecimal characters into pairs and perform the swapping of the higher nibble of the first character with the nibble of the second character and the last nibble of the first character with the first nibble of the second character. Finally, convert each byte (pair of nibble) to character and this gives the original plaintext.

Note that these ciphertext generated as a result of the row and column substitutions are Chinese characters which have meanings. A Chinese character can either mean a word or a phrase and also a word can comprise of two or three of the Chinese character e.g. spoon 勺子, chocolate 巧克力.

Enhanced algorithm

Section A shows the entire encryption of plaintext to ciphertext and decryption of ciphertext to plaintext of the enhanced algorithm. Illustration of the enhanced algorithm using English characters.

Section C illustrates how the input plaintext to the algorithm is encrypted. The plaintext: “demo” is divided into pairs, converted into their hexadecimal equivalent and then the nibbles are swapped. Table 1 shows the plaintext –hexadecimal mapping and swapping of nibbles.

Table 1: Plaintext-hexadecimal mapping

Plaintext	**d**	**e**	**m**	**o**
Hexadecimal Equivalent	44	65	6D	6F
Modified Hexadecimal decimal	56	44	F6	D6

For the encryption the first byte (56) which has two nibble is picked. The first nibble (5) is used for the column key while the second nibble (6) is used for the row key and the intersecting Chinese character becomes the ciphertext. Table 2 shows the entire 17x17 matrix and the first ciphertext obtained.

After the first encryption the row key digit shifts to the end of the row. This shifting of the keys changes the position of the entire table which means the entire table has been shuffled. Table 3 shows the intersection of the cipher character obtained.

For the encryption the next byte (F6) which has two nibble is picked. The first nibble (F) is used for the column key while the second nibble (6) is used for the row key and the intersecting Chinese character becomes the ciphertext. Table 4 shows the intersection of the cipher character obtained.

For the encryption the next byte (D6) which has two nibble is picked. The first nibble (D) is used for the column key while the second nibble (6) is used for the row key and the intersecting Chinese character becomes the ciphertext. Table 4 shows the intersection of the cipher character obtained.

Table 6: the ciphertext – plaintext mapping. Each of the ciphertext generated is converted back to its hexadecimal equivalent and to its original character.

These (矢耳冫走) are the Chinese ciphertext obtained with their various meanings: 矢 stands for arrow, 耳 stands for ear, 冫 stands forice/cold, and 走 stands for stamping on the earth

Table 6: Ciphertext- plaintext mapping

Ciphertext	**矢**	**而**	**冫**	**走**
Hexadecimal Equivalent	56	44	F6	D6
Plaintext	d	e	M	O

Table 6 shows the ciphertext generated from the plaintext “demo” with the modified hexadecimal after swapping a pair of byte as seen in table1.

RESULTS AND DISCUSSION

The enhanced algorithm was implemented using java. The ability of the enhanced algorithm to hide plaintexts in Chinese characters makes it look less than a cipher and more of a normal plaintext. This unique feature makes the cipher strong enough to withstand any form cryptanalysis and tends to fend-off any potential attacker. The meaning of the Chinese characters generated has no relationship with the plaintext encrypted. The cipher encrypt more faster than the one proposed by Nuhu *et al*. [8] because of the reduction in character search and matrix shuffling.

Figure 1 shows the interface of the enhanced algorithm with a plaintext of size 248 kb to be encrypted. Since the plaintext to be encrypted is in English so the English is selected. Figure 2 shows the generated Chinese ciphertext within a timeframe of 0.133 seconds. Figure 3 shows the plaintext area cleared for ciphertext to decrypted back to its original plaintext. Figure 4 shows that it took 0.099 seconds to decrypt the Chinese ciphertext to the original plaintext.

Brute force attack

Brute force seeks to find the possible combination of possible keys. Since the algorithm is made up of the whole languages, brute force will go through the entire languages of 65536x65536 characters. However, it is possible to keep ciphering the ciphertext as many times as possible with the same key or different keys which makes it even more difficult to decrypt. This enhanced work cannot be cracked by brute force.

For Nuhu *et al.* [8] work which used Unicode characters, the attacker has to find from a 65536x65536 digraphs, which is practically impossible. Therefore, it cannot be cracked by Brute Force Attack.

Figure 5 shows an encryption of a word demo with an encryption key consisting of 3 letters. Figure 6 shows a brute force approach which tries to guess the encryption key by using a different key of 8 characters on the ciphertext generated by the word "demo".

Known plaintext attack

The enhanced cipher is free from known plaintext attacked since the ciphertext does not leaves any trace of something hidden. The key matrix is being shuffled after encrypting each character by just shifting a key character. This eliminates the fact that a pair of character and its reverse will encrypt in a similar fashion. (i.e. if XY encrypt to AB then YX will encrypt to BA). In the enhanced cipher XY encrypt to 臣米 then YX will encrypt to 牛牙 using the keyword 12a.

Figure 7 shows the encryption of two characters "a" and "b" into two Chinese characters 臣 and 米 which means: servant and rice respectively. Figure 8 shows that there is no one-one mapping of characters. This kind of attack will keep repeating characters to see if there is any particular pattern it can get to come up with a strategy to break the cipher. But trying "b" and "a" will get another different Chinese character 牛 and 牙 which means: cow/ox and teeth respectively.

Frequency analysis

Frequency analysis seeks to uncover the message by studying the frequency of letters or groups of letters contained in the ciphertext. Frequency analysis will be of no good since there is no one-to- one mapping i.e. repetition of characters leads to different Chinese characters and also means a different thing entirely. Encrypting the same character leads to different Chinese character. According to Zhao and Zhang [9] Chinese characters have a total of 85000 different Chinese characters. The probability of a character reappearing in a ciphertext gives a pattern which makes it easy for an attacker to break. The probability of occurrence an attacker will consider in the enhanced cipher is 1/85000 = 0.0000111764.

For Nuhu *et al.* [8] the probability of occurrence that an attacker will consider is1/65536 = 0.0000152. Therefore, probability of occurrence of a character in the enhanced cipheris less compared to that of Nuhu et al [8] which makes it more secured.

Low time requirement

Table 7 shows the time for encryption and decryption it took the enhanced algorithm and that of Nuhu *et al*. [8] to encrypt and decrypt different size of text in kilobyte. The running time for both the enhanced algorithm and that of Nuhu *et al.* [8] are shown in Table 7. It can be seen that the enhanced algorithm runs faster when compared to that of Nuhu *et al*. [8] as shown in Table 7. Shuffling and searching for characters in a 16x16 matrix will give rise to high computation time.

Table 7: Encryption and decryption time

Size of text (KB)	2	4	6	8	10	12	14
Encryption time (sec) Enhanced algorithm	0.016	0.028	0.031	0.046	0.047	0.056	0.078
Decryption time (sec) enhanced algorithm	0.015	0.016	0.022	0.031	0.032	0.047	0.063
Encryption time (sec) Nuhu *et al.* [8]	0.046	0.062	0.063	0.078	0.109	0.125	0.141
Decryption time (sec) Nuhu *et al.*[8]	0.032	0.046	0.057	0.063	0.088	0.094	0.125

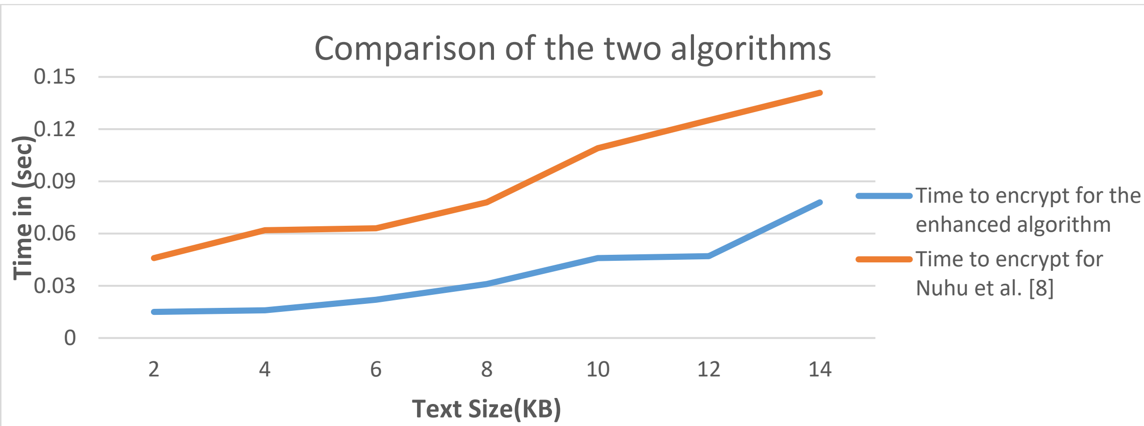

Figure 5: Encryption time for both ciphers

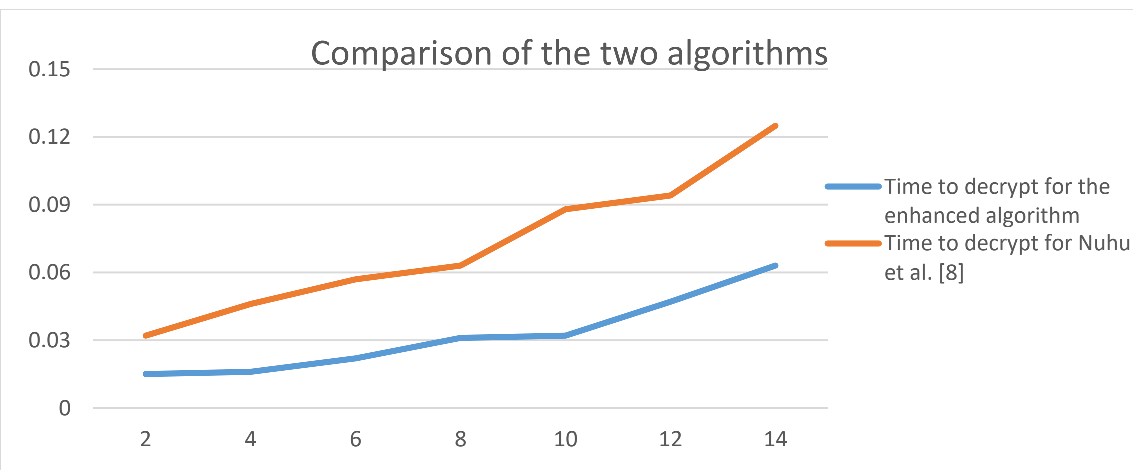

Figure 6: Decryption time for both ciphers

The encryption time for the enhanced algorithm is very low when compared with Nuhu *et al.* [8] as shown in Figure 5. This is because the enhanced algorithm searches for a nibble in the row and column key while Nuhu *et al.* [8] algorithm searches for two bytes in the matrix. Also, the entire matrix is not shuffled but shifts just the row key characters to give the entire matrix a new look.

The decryption time for the enhanced algorithm is very low when compared with Nuhu *et al.* [8] because only a byte (Chinese character) searched in the entire matrix and the resultant nibbles intersecting the character are obtained. Unlike Nuhu *et al.* [8] algorithm two bytes have to be searched in order to obtain the substituted bytes and the entire matrix is shuffled after each encryption.

CONCLUSION

It is possible to enhance the computation time and memory efficiency of a 16x16 playfair matrix for encrypting Unicode characters so it can be more suitable for mobile devices that have low memory and require low power consumption. Increasing the size of the matrix to 17x17 and shifting the row key characters instead of shuffling the entire matrix after each encryption has led to the significant reduction in the computation time as shown in Table 7. Also, swapping of nibbles carried out is to ensure that the strength of the algorithm is not compromised. It is possible to transform any language into Chinese characters which have meanings in order to fend-off suspicion by attackers.

REFERENCES

1. BHATTACHARYYA, S., CHAND, N. & CHAKRABORTY, S. (2014). A Modified Encryption Technique using Playfair Cipher 10 by 9 Matrix with Six Iteration Steps. *Computer Engineering & Technology*, **3**(2): 307 – 312.
2. KHAN, S.A. (2015). Design and Analysis of Playfair Ciphers with Different Matrix Sizes. *International Journal of Computing and Network Technology*. **3**(3): 2210-1519.
3. TUNGA, H. & MUKHERJEE, S., (2012). A New Modified Playfair Algorithm Based On Frequency Analysis. *International Journal of Emerging Technology and Advanced Engineering*, ISSN 2250-2459, Volume **2**, Issue 1, January 2012
4. LAHIRI, A., (2012). Design and Implementation and Enhanced Binary Playfair Algorithm Using a 4x4 Key Matrix. Jadavpur University, Kolkata, India, pp 1-59.
5. MASADEHS. R., AL_SEWADIH. A. & WADIM, A. (2016). A Novel Paradigm for Symmetric Cryptosystem. (IJACSA) *International Journal of Advanced Computer Science and Applications*, Vol. 7, No. 3
6. AHNAF, T.S. & RABIUL, I. (2014). An efficient modification to Playfair Cipher. Ulab *Journal of Science and Engineering* Vol. **5**, NO. 1, (ISSN: 2079-4398).
7. AHMED, O.H., AHMED, A.M., & AHMED, S. H. (2015). Improving Playfair Algorithm to Support User Verification and all the Languages in the World including Kurdish Language. *International Journal of Engineering and Computer Science*, **4**(8): 14058-14062.
8. NUHU *et al.* (2017). Reduced Playfair Matrix for Unicode characters; *Nigerian Journal of Scientific Research*, **16**(4): 407-411 July – August; njsr.abu.edu.ng
9. ZHAO, S. & ZHANG, D. (2008). The Totality of Chinese Characters – A Digital Perspective. *Journal of Chinese Language and Computing* **17**(2): 107-125
10. http://www.chinaknowledge.de/Literature/radicals.htm. Retrieved November, 2018.
11. INDEPENDENT SCHOOLS EXAMINATIONS BOARD (ISEB) Mandarin_ Chinese_ Common_Entrance_Level_1_and_2_word_ and_character_list.pdf. Retrieved August, 2018.

Table 2: Starting 17x17 matrix

key	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F	丨	一	冖	黹	冫	黽	至	匚	匸	他	口	六	囗	胖	鬼	不
E	亅	鬲	贯	韭	几	鼎	龜	黃	十	鳥	尢	走	黑	土	辣	和
D	亠	食	累	辵	凵	书	豸	衣	卜	山	尸	你	士	身	新	脚
C	人	隶	鼻	首	刀	采	麻	車	卩	工	屮	齒	夊	長	咸	腿
B	入	面	瘦	酉	力	她	自	肉	厂	巾	广	魚	夕	阜	也	绿
A	八	骨	非	韋	勹	頁	音	門	厶	干	幺	辰	大	青	飛	红
9	冂	金	隹	鬥	匕	疋	瓦	麥	又	黍	馬	是	女	雪	龠	叉
8	子	尺	髟	里	宀	龍	癶	齊	寸	高	豕	会	小	网	茶	甜
7	曰	鹵	香	吗	斤	跟	老	足	支	行	角	革	彳	廴	谷	海
6	月	目	禾	辛	方	疒	邑	五	攴	课	白	鼠	心	廾	鬯	家
5	木	赤	虍	穴	无	艮	矢	貝	文	四	玉	弋	戈	言	弓	二
4	欠	虫	生	耳	日	而	豆	舛	斗	聿	玄	示	戶	耒	彡	三
3	父	羽	点	内	水	艸	見	用	毛	色	矛	竹	手	舌	止	九
2	爻	瓜	石	羊	火	糸	牙	臣	氏	立	血	臼	毋	雨	歹	两
1	爿	鹿	田	舟	这	犬	米	牛	气	缶	片	在	比	殳	風	天
0	福	喜	热	风	穿	看	有	吃	笔	谁	那	我	只	件	个	元

Table 3: First shift of key value

key	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
F	丨	一	冖	黹	冫	黽	至	匸	匚	他	口	六	囗	胖	鬼	不
E	亅	鬲	贯	韭	几	鼎	龜	黃	十	鳥	尢	走	黑	土	辣	和
D	亠	食	累	辵	凵	书	豸	衣	卜	山	尸	你	士	攵	身	新	脚
C	人	隶	鼻	首	刀	采	麻	車	卩	工	屮	齒	夊	長	咸	腿
B	入	面	瘦	酉	力	她	自	肉	厂	巾	广	魚	夕	阜	也	绿
A	八	骨	非	韋	勹	頁	音	門	厶	干	幺	辰	大	青	飛	红
9	冂	金	隹	鬥	匕	疋	瓦	麥	又	黍	馬	是	女	雪	龠	叉
8	子	尺	彡	里	宀	龍	入	齊	寸	高	豕	会	小	网	茉	甜
7	曰	鹵	香	吗	斤	跟	老	足	支	行	角	革	彳	廴	谷	海
6	月	目	禾	辛	方	广	邑	五	攴	课	白	鼠	心	井	鬯	家
5	木	赤	虍	穴	无	艮	矢	貝	文	四	玉	弋	戈	言	弓	二
4	欠	虫	生	耳	日	而	豆	舛	斗	聿	玄	示	户	耒	彡	三
3	父	羽	点	内	水	艸	見	用	毛	色	矛	竹	手	舌	止	九

Table 4: Second shift of key value

key	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1
F	丨	一	冖	黹	冫	黽	至	匸	匚	他	口	六	囗	胖	鬼	不
E	亅	鬲	贯	韭	几	鼎	龜	黃	十	鳥	尢	走	黑	土	辣	和
D	亠	食	累	辵	凵	书	豸	衣	卜	山	尸	你	士	身	新	脚
C	人	隶	鼻	首	刀	采	麻	車	卩	工	屮	齒	夊	長	咸	腿
B	入	面	瘦	酉	力	她	自	肉	厂	巾	广	魚	夕	阜	也	绿

Table 5: Third shift of key value

key	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2
F	丨	一	冖	黹	冫	黽	至	匸	匚	他	口	六	囗	胖	鬼	不
E	亅	鬲	贯	韭	几	鼎	龜	黃	十	鳥	尢	走	黑	土	辣	和
D	亠	食	累	辵	凵	书	豸	衣	卜	山	尸	你	士	身	新	脚
C	人	隶	鼻	首	刀	采	麻	車	卩	工	屮	齒	夊	長	咸	腿
B	入	面	瘦	酉	力	她	自	肉	厂	巾	广	魚	夕	阜	也	绿

Figure 1: The interface of the enhanced algorithm with a plaintext to be encrypted

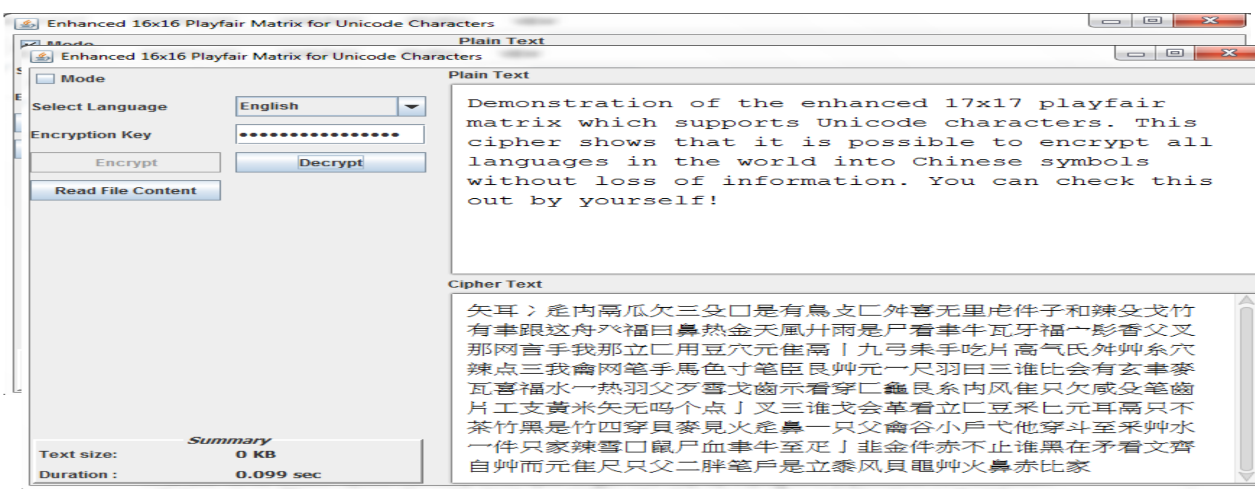

Figure 2: Displays the generated Chinese ciphertext from the plaintext.

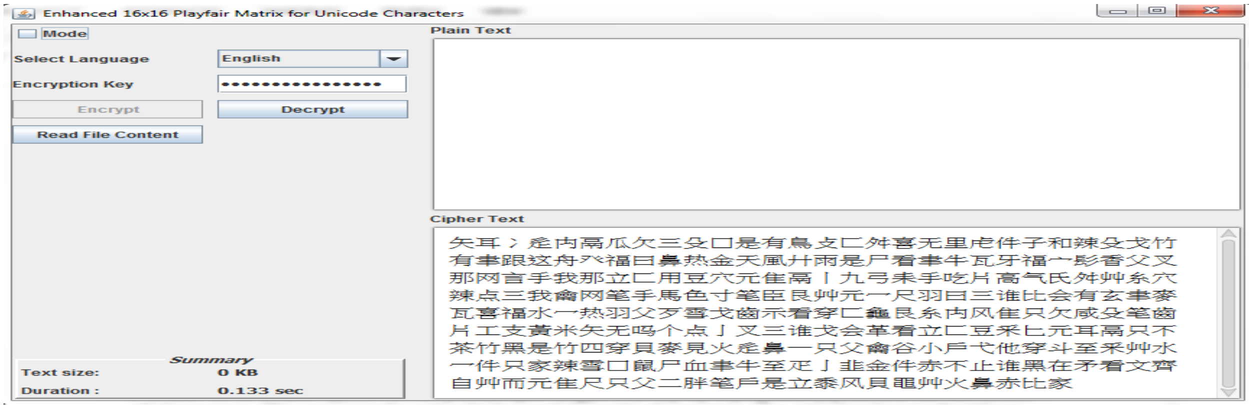

Figure 3: Blank text area which will display the result of the decryption of the ciphertext.

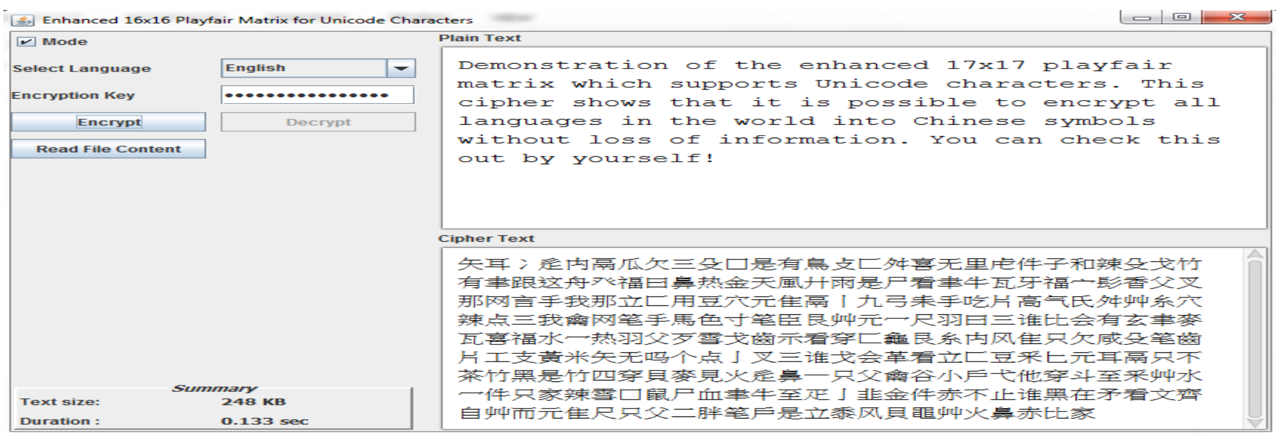

Figure 4: Decryption of the Chinese ciphertext back to the original plaintext that was used for the encryption

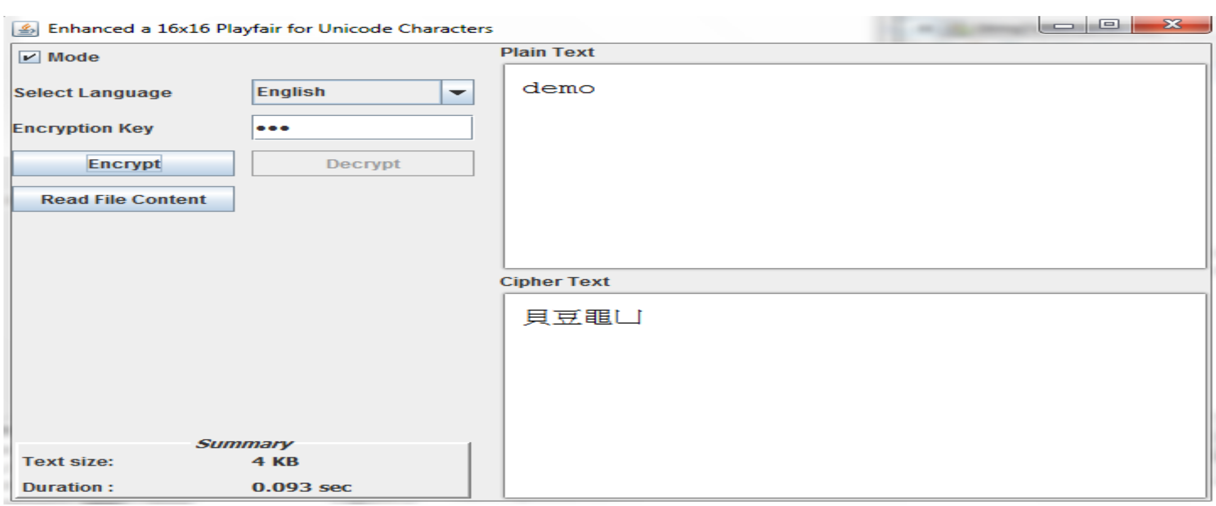

Figure 5: shows an encryption demo

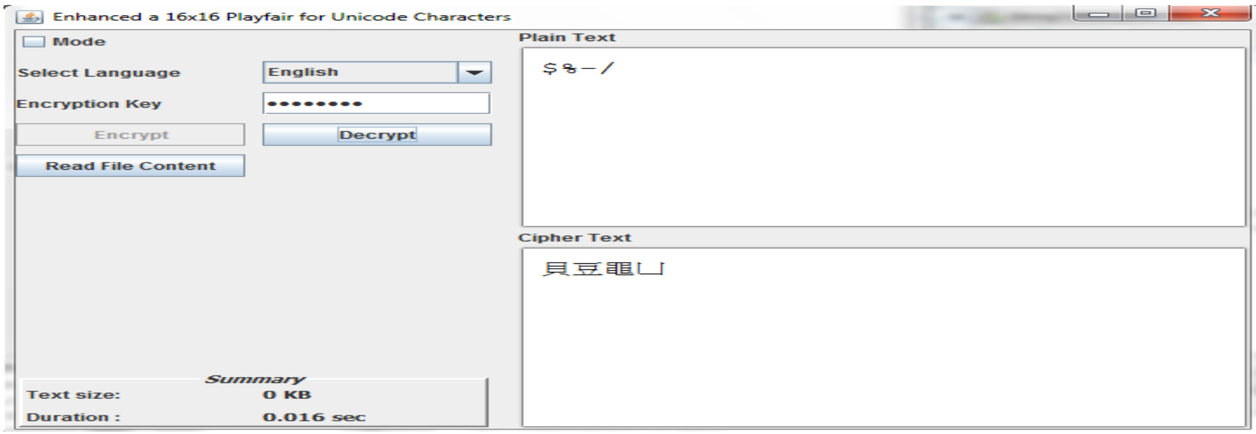

Figure 6: shows a decryption of demo with a wrong keyword

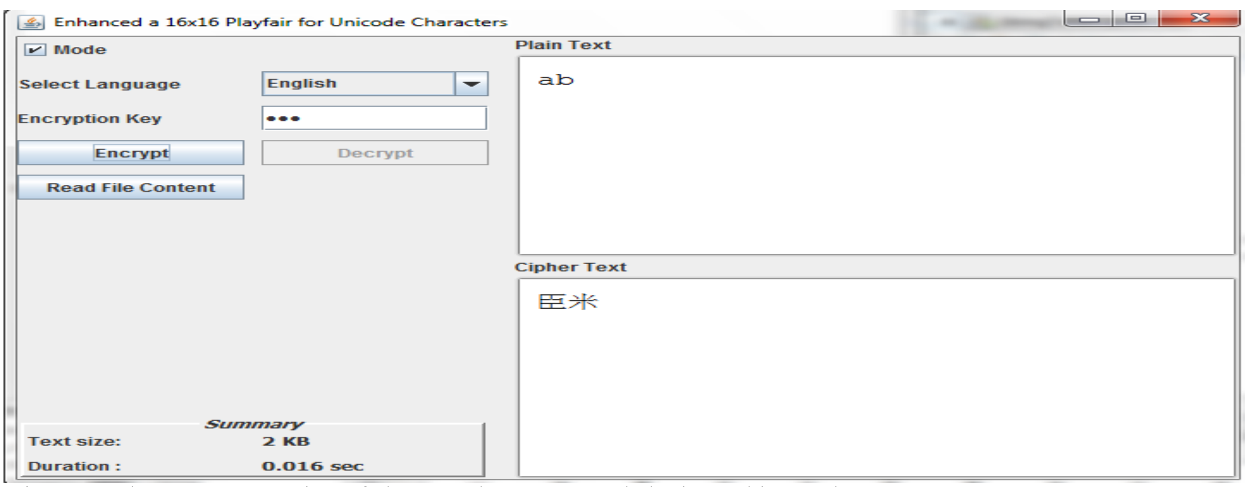

Figure 7: shows an encryption of character letters “a” and “b” into Chinese characters

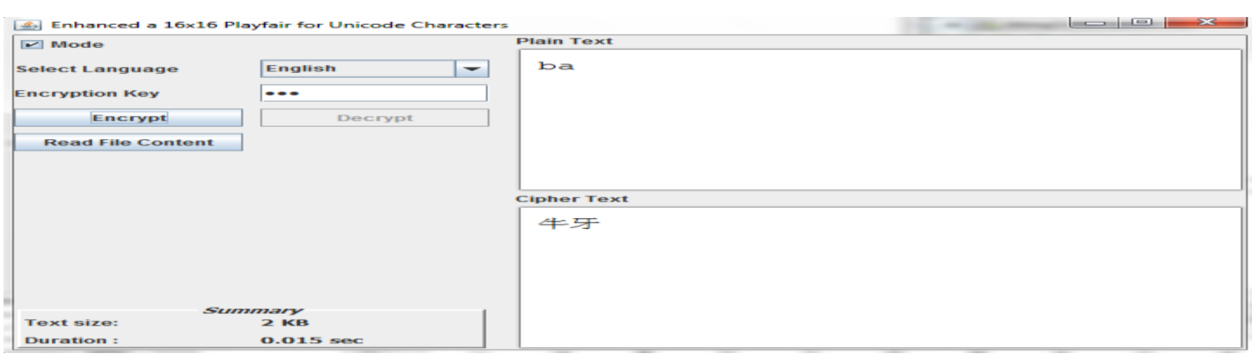

Figure 8: Shows decryption of 臣米 Chinese characters