riochr17 / **Jurnal-Sandi**

Branch: master ▾ **Jurnal-Sandi** / task1 / **playfair.py** Find file Copy path

riochr17 final vigenere + playfair 9fddaf6 3 minutes ago

1 contributor

164 lines (129 sloc) | 3.98 KB

```
import string

class PlayFair:
        # linear key
        linear_key = []

        # matrix key
        key = []

        # hash map key
        hash_key = {}

        # bichar delimiter
        bichar_delimiter = ','

        # outline c
        outline_char = 'X'

        # key excluded
        missing_alphabet = 'J'

        # change key excluded
        replacing_alphabet = 'I'

        # constructor
        def __init__(self, str_key):
                self.construct_key(str_key)

        def get_bicrypt(self, bichar, is_encrypt):
                a = self.hash_key[bichar[0]]
                b = self.hash_key[bichar[1]]
                c = ''
                d = ''

                #1 pada baris yang sama
                if(a[0] == b[0]):
                        c = self.key[a[0]][(a[1] + (1 if is_encrypt else -1)) % 5]
                        d = self.key[b[0]][(b[1] + (1 if is_encrypt else -1)) % 5]
                        # ...
                #2 pada kolom yang sama
                else:
                        if(a[1] == b[1]):
                                c = self.key[(a[0] + (1 if is_encrypt else -1)) % 5][a[1]]
                                d = self.key[(b[0] + (1 if is_encrypt else -1)) % 5][b[1]]
                                # ...
                        #3 pada kolom yang sama
                        else:
                                c = self.key[a[0]][b[1]]
                                d = self.key[b[0]][a[1]]
                                # ...

                return c + d

        def encrypt(self, pt):
                ct = ''
                bichars = self.get_bichar_of_string(pt.replace(self.missing_alphabet, self.replacing_alphabet)).split(self.bich
                for bichar in bichars:
                        ct += self.get_bicrypt(bichar, is_encrypt = True)
```

```python

                return ct

        def decrypt_and_predict_text(self, str_data):
                pdt = self.decrypt(str_data)
                last_char = ''
                remove_list_index = []
                for i in range(len(pdt)):
                        if i == 0:
                                continue
                        if i == len(pdt) - 2:
                                break
                        last_char = pdt[i - 1]
                        curr_char = pdt[i]
                        pred_char = pdt[i + 1]
                        if last_char == pred_char and curr_char == self.outline_char:
                                remove_list_index.append(i)

                for i in remove_list_index:
                        pdt = pdt[:i] + pdt[(i + 1):]

                return pdt

        def decrypt(self, pt):
                ct = ''
                bichars = self.get_bichar_of_string(pt).split(self.bichar_delimiter)
                for bichar in bichars:
                        ct += self.get_bicrypt(bichar, is_encrypt = False)

                return ct

        # convert string bichar separated to list
        def bichar_to_list(self, str_data):
                bichars = self.string_to_bichar("", str_data).split(self.bichar_delimiter)
                list_bichar = []
                for bichar in bichars:
                        list_bichar.append([bichar[0], bichar[1]])

                return list_bichar

        def get_bichar_of_string(self, str_data):
                return self.string_to_bichar("", str_data)

        # process string to bichar separated
        def string_to_bichar(self, result_str, old_str):
                old_str = old_str.upper().replace(' ', '')
                if len(old_str) == 0:
                        return (result_str + self.outline_char) if len(result_str.replace(self.bichar_delimiter, '')) % 2 == 1
                else:
                        delimiter = self.bichar_delimiter if len(result_str.replace(self.bichar_delimiter, '')) % 2 == 1 else '
                        if old_str[:1] == result_str[-1:]:
                                c = self.outline_char + delimiter
                                return self.string_to_bichar(result_str + c, old_str)
                        else:
                                c = old_str[0] + delimiter
                                return self.string_to_bichar(result_str + c, old_str[1:])

        # contruct key in linear and matrix
        def construct_key(self, str_key):
                self.construct_linear_key(str_key)
                self.construct_matrix_key()
                self.contruct_hash_key()

        # contruct key return linear one dimensional list
        def construct_linear_key(self, str_key):
                max_key_len = 25
                for c in str_key.upper().replace(' ', '').replace(self.missing_alphabet, self.replacing_alphabet):
                        if not c in self.linear_key:
                                self.linear_key.append(c)
                        if len(self.linear_key) == 25:
```

```python
                break

        char_list = list(string.ascii_uppercase.replace(self.missing_alphabet, ''))
        for c in char_list:
            if not c in self.linear_key:
                self.linear_key.append(c)
            if len(self.linear_key) == 25:
                break

    # contruct key return linear matrix dimensional list
    def construct_matrix_key(self):
        line = 0
        matrix_width = 5
        iterator = matrix_width
        tmp_row = []
        for c in self.linear_key:
            if iterator == 0:
                self.key.append(tmp_row)
                tmp_row = []
                iterator = matrix_width
            tmp_row.append(c)
            iterator -= 1

        self.key.append(tmp_row)

    def contruct_hash_key(self):
        for i in range(len(self.key)):
            for j in range(len(self.key[i])):
                self.hash_key[self.key[i][j]] = [i, j]
```