

1.

• 1-7)

1. multiplication table for Z_4 :

\times	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

2. multiplication table for Z_5 :

\times	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

addition tables for Z_5 :

\times	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

3. multiplication table for Z_6 :

\times	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

addition tables for Z_6 :

\times	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

4. Elements without a multiplicative inverse in Z_4 : 0, 2

Elements without a multiplicative inverse in Z_6 : 0, 2, 3, 4

5 is a prime number, so a multiplicative inverse for all nonzero elements of Z_5 exists. And all nonzero elements smaller than 5 are relatively prime to 5.

• 1-8)

Because 5 is relatively prime to 11, 12 and 13; there is a multiplicative inverse of it in each Z_{11} , Z_{12} and Z_{13} .

• trial-and-error search in Z_{11} :

- $5 \cdot 0 = 0$
- $5 \cdot 1 = 5$
- $5 \cdot 2 = 10$
- $5 \cdot 3 = 15 \bmod 11 = 4$
- $5 \cdot 4 = 20 \bmod 11 = 9$
- $5 \cdot 5 = 25 \bmod 11 = 3$
- $5 \cdot 6 = 30 \bmod 11 = 8$
- $5 \cdot 7 = 35 \bmod 11 = 2$
- $5 \cdot 8 = 40 \bmod 11 = 7$
- $5 \cdot 9 = 45 \bmod 11 = 1$
- $5 \cdot 10 = 50 \bmod 11 = 6$

So, the multiplicative inverse of 5 in Z_{11} is 9.

- trial-and-error search in Z_{12} :

- | | |
|----------------------------------|----------------------------------|
| - $5 \cdot 0 = 0$ | - $5 \cdot 1 = 5$ |
| - $5 \cdot 2 = 10$ | - $5 \cdot 3 = 15 \bmod 12 = 3$ |
| - $5 \cdot 4 = 20 \bmod 12 = 8$ | - $5 \cdot 5 = 25 \bmod 12 = 1$ |
| - $5 \cdot 6 = 30 \bmod 12 = 6$ | - $5 \cdot 7 = 35 \bmod 12 = 11$ |
| - $5 \cdot 8 = 40 \bmod 12 = 4$ | - $5 \cdot 9 = 45 \bmod 12 = 9$ |
| - $5 \cdot 10 = 50 \bmod 12 = 2$ | - $5 \cdot 11 = 55 \bmod 12 = 7$ |

So, the multiplicative inverse of 5 in Z_{12} is 5.

- trial-and-error search in Z_{13} :

- | | |
|-----------------------------------|----------------------------------|
| - $5 \cdot 0 = 0$ | - $5 \cdot 1 = 5$ |
| - $5 \cdot 2 = 10$ | - $5 \cdot 3 = 15 \bmod 13 = 2$ |
| - $5 \cdot 4 = 20 \bmod 13 = 7$ | - $5 \cdot 5 = 25 \bmod 13 = 12$ |
| - $5 \cdot 6 = 30 \bmod 13 = 4$ | - $5 \cdot 7 = 35 \bmod 13 = 9$ |
| - $5 \cdot 8 = 40 \bmod 13 = 1$ | - $5 \cdot 9 = 45 \bmod 13 = 6$ |
| - $5 \cdot 10 = 50 \bmod 13 = 11$ | - $5 \cdot 11 = 55 \bmod 13 = 3$ |
| - $5 \cdot 12 = 60 \bmod 13 = 8$ | |

So, the multiplicative inverse of 5 in Z_{13} is 8.

- 1-9)

- $3^2 \bmod 13 = 9 \rightarrow \mathbf{x = 9}$
- $7^2 \bmod 13 = 49 \bmod 13 = 10 \rightarrow \mathbf{x = 10}$
- $3^{10} \bmod 13 = 3^4 \cdot 3^3 \cdot 3^3 \bmod 13 = 81 \cdot 27 \cdot 27 \bmod 13 = 3 \cdot 1 \cdot 1 \bmod 13 = 3$
 $\rightarrow \mathbf{x = 3}$
- $7^{100} \bmod 13 = (7^2)^{50} \bmod 13 = 49^{50} \bmod 13 = 10^{50} \bmod 13 = (-3)^{50} \bmod 13$
 $= (3^{10})^5 \bmod 13 = 3^5 \bmod 13 = 3^4 \cdot 3 \bmod 13 = 81 \cdot 3 \bmod 13$
 $= 3 \cdot 3 \bmod 13 = 9 \rightarrow \mathbf{x = 9}$
- $11 \bmod 13 = 11 + 13m = 7^x \rightarrow 7^x \bmod 13$
 trial-and-error: $x = 5 \rightarrow 7^5 = 7^2 \cdot 7^2 \cdot 7 \bmod 13$
 $= 10 \cdot 10 \cdot 7 \bmod 13$
 $= 70 \cdot 10 \bmod 13$
 $= 50 \bmod 13 = 11$
 $(x = 12 \rightarrow 7^{12} = (7^3)^4 \bmod 13 = 5^4 \bmod 13 = 1)$
 $\rightarrow \mathbf{x = 5 + 12k}$

• 1-10)

- $m = 4 \rightarrow n = 1, 3 \rightarrow \varphi(4) = 4 \cdot (1 - \frac{1}{2}) = 2$
- $m = 5 \rightarrow n = 1, 2, 3, 4 \rightarrow \varphi(5) = 5 \cdot (1 - \frac{1}{5}) = 4$
- $m = 9 \rightarrow n = 1, 2, 4, 5, 7, 8 \rightarrow \varphi(9) = 9 \cdot (1 - \frac{1}{3}) = 6$
- $m = 26 \rightarrow n = 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25$
 $\rightarrow \varphi(26) = 26 \cdot (1 - \frac{1}{2}) \cdot (1 - \frac{1}{13}) = 12$

- 1-13) In affine cipher: $y = a x + b$, And here we have (x_1, y_1) and (x_2, y_2)

$$\text{So: } \begin{cases} y_1 = a x_1 + b \\ y_2 = a x_2 + b \end{cases} \longrightarrow \begin{cases} a = (x_1 - x_2)^{-1} (y_1 - y_2) \bmod m \\ b = y_1 - a x_1 \bmod m \end{cases}$$

[note that: $(x_1 - x_2)^{-1}$ must exist mod $m \rightarrow$ i.e. $\gcd((x_1 - x_2), m) = 1$]

2.

(2-1)

- LFSR هایی که چندجمله ای فیدبک آن ها از نوع Irreducible polynomial است، یعنی نمی توان آن ها را به چندجمله ای با درجه کوچک تر تجزیه کرد؛ دوره تناوب خروجی شان دقیقاً برابر exponent شان می شود ($P = N$). اگر tap آخر وجود داشته باشد، تضمین می کند که خروجی حتماً پریودیک است، و چندجمله ای تبدیل به primitive می شود؛ اگر وجود نداشته باشد، نهایتاً پریودیک است.
- LFSR هایی که چندجمله ای فیدبک آن ها از نوع primitive polynomial است، خروجی شان دارای حداکثر پریود ممکن است. یعنی اگر طول LFSR را با m نمایش دهیم، پریود برابر است با:

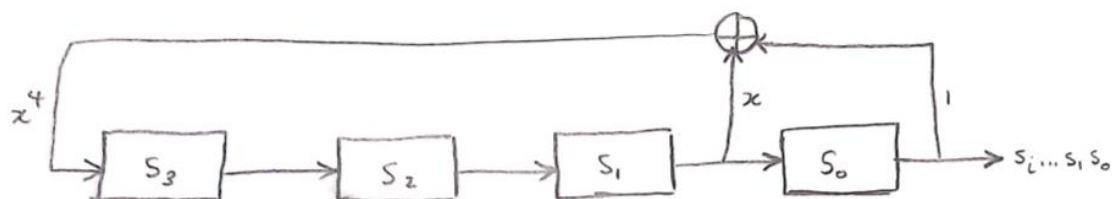
$$P = 2^m - 1$$

این ها حالت خاصی از Irreducible polynomial ها هستند. و دارای ویژگی های آن ها نیز هستند. همچنین، خروجی ای که تولید می کنند، دارای خواص آماری خوبی است و دارای توزیع یکنواخت است. در این ها tap آخر باید حتماً وجود داشته باشد. و خروجی حتماً پریودیک است. ورودی را باید از حالت غیر صفر شروع کنیم.

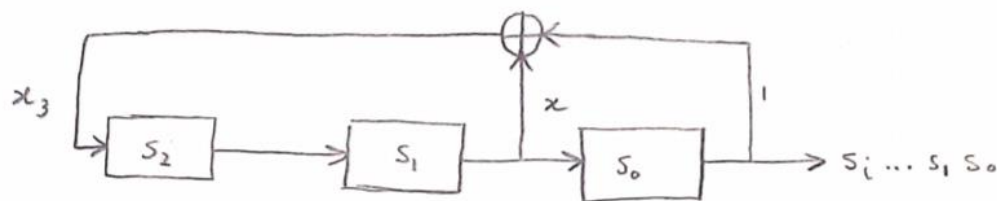
- چون tap آخر وجود دارد، اگر از حالت ورودی غیر صفر شروع کنیم، هیچ گاه به صفر نمی رسیم.
- برای LFSR هایی که چندجمله ای فیدبک آن ها از نوع Reducible polynomial است، یعنی می توان آن ها را به چندجمله ای با درجه کوچک تر تجزیه کرد؛ دیگر نمی توان گفت که خروجی شان دقیقاً برابر exponent شان می شود. بلکه فقط می توان گفت که:

$$p \mid N : (N = p \cdot q)$$

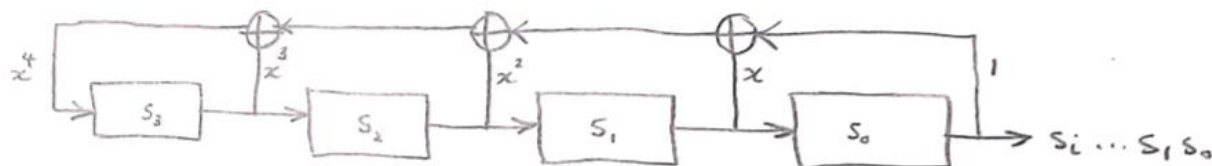
- $x^4 + x + 1$:



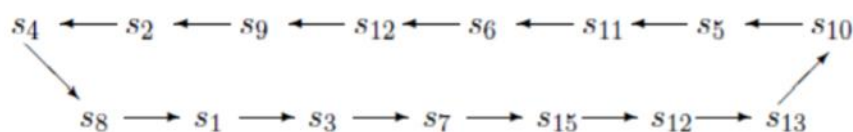
- $x^3 + x + 1$:



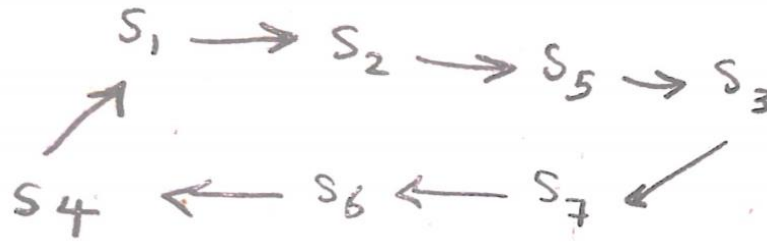
- $x^4 + x^3 + x^2 + x + 1$:



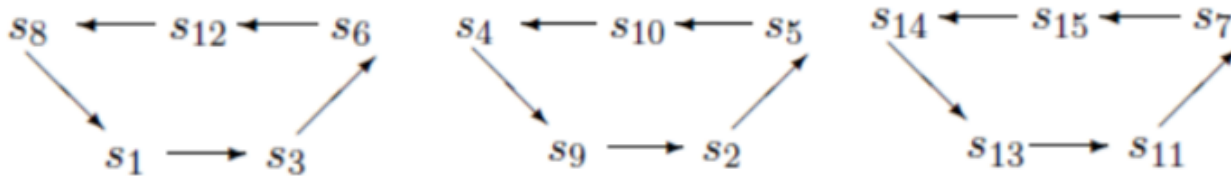
- چندجمله ای اول Irreducible و primitive است؛ زیرا اولاً به چندجمله ای های کوچک تر قابل تجزیه نیست، و دوماً دارای یک لوپ با طول حداکثر است. یعنی پرپود آن به این شکل است:
 $P = 2^L - 1 = 2^4 - 1 = 15$



- چندجمله ای دوم Irreducible و primitive است؛ زیرا اولاً به چندجمله ای های کوچک تر قابل تجزیه نیست، و دوماً دارای یک لوپ با طول حداکثر است. یعنی پرپود آن به این شکل است:
 $P = 2^L - 1 = 2^3 - 1 = 7$



- چندجمله ای سوم Irreducible است؛ زیرا به چندجمله ای های کوچک تر قابل تجزیه نیست، و دارای چند لوپ است:



(2-4)

- چند جمله ای اول: $2^L - 1 = 2^4 - 1 = 15$
- چند جمله ای دوم: $2^L - 1 = 2^3 - 1 = 7$
- چند جمله ای دوم: $N = P = 5$

3.

(a) Attacker به 512 جفت بیت متوالی x_i, y_i مربوط به plainText نیاز دارد، تا حمله ی موفقیت آمیزی داشته باشد.

(b)

1. Attacker اول باید آن 512 جفت بیتی که در قسمت قبل گفته شد، را تهیه کند.
 2. سپس باید $s_i = x_i + y_i \mod 2$; $i = 0, 1, \dots, 2m-1$ را محاسبه کند.
 3. Attacker باید به منظور به دست آوردن چندجمله ای فیدبک، 256 عبارت خطی با استفاده از رابطه ی بین بیت های آن و key stream خروجی آن که با رابطه ی زیر محاسبه می شود، تولید کند:
- $$s_{i+m} \equiv \sum_{j=0}^{m-1} p_j \cdot s_{i+j} \mod 2 \quad ; \quad s_i, p_j \in \{0,1\} ; i = 0, 1, 2, \dots, 255 \quad ; \quad m = 256$$
4. پس از تولید این عبارت خطی، آن را حل می کنیم. و آن 256 ضریب فیدبک را فاش می کنیم.

(c) کلید این سیستم با 256 ضریب فیدبک نشان داده می شود.
با توجه به اینکه محتوای اولیه ی این LFSR به صورت یک طرفه به بیرون آن شیفِت داده می شود و با 256 بیت اول plaintext XOR می شوند؛ به آسانی قابل محاسبه هستند.

4.

به توجه به اینکه می دانیم سه حرف اول، چه حروفی هستند، بررسی می کنیم:

W: 22 = 10110₂ → J: 9 = 01001₂
P: 15 = 01111₂ → 5: 31 = 11111₂
I: 8 = 01000₂ → A: 0 = 00000₂

با توجه به اینکه داریم:

$\text{plaintext} \oplus \text{key} = \text{ciphertext} \rightarrow \text{plaintext} \oplus \text{ciphertext} = \text{key}$

با استفاده XOR برای این 3 جفت کاراکتر اصلی و کاراکتر رمز شده ی متناظر، کلید را پیدا می کنیم:

10110 01111 01000 : plaintext
01001 11111 00000 : ciphertext

11111 10000 01000 : keystream

1.

```
def bit_encoding_map():  
    return zip(string.ascii_lowercase + string.digits, range(0, 32))  
  
def bitencode(text):  
    def encode_char(c):  
        map = dict(bit_encoding_map())  
        return bin(map[c.lower()]).lstrip("0b").zfill(5)  
    return "".join([encode_char(c) for c in text])  
  
def bitdecode(text):  
    def decode_block(bits):  
        map = dict([(b, a) for a, b in bit_encoding_map()])  
        return map[int(bits, 2)].upper()  
    return "".join([decode_block(bits) for bits in re.findall("{1,5}", text)])  
  
def lfsr(init_vector):  
    registers = list(bin(init_vector).lstrip("0b").zfill(6)[-6:])  
    while True:  
        registers.insert(0, "1" if registers[-1] != registers[-2] else "0")  
        yield registers.pop()  
  
def xor_bitstream(a, b):  
    return "".join(["1" if a != b else "0" for a, b in zip(a, b)])
```

```

def apply_keystream(lfsr_init_vector, text):
    return bitdecode(xor_bitstream(lfsr(lfsr_init_vector), bitencode(text)))

def print_ciphertext_and_chosen_plaintext(ciphertext, chosen_plaintext):
    print "Ciphertext:"
    print "-----\n"
    print ciphertext
    print "-----\n"
    print "Chosen Plaintext:"
    print "-----\n"
    print chosen_plaintext

def print_keystream(keystream):
    print "-----\n"
    print "Revealed keystream:"
    print "-----\n"
    print keystream

def print_decrypted_message(plaintext):
    print "-----\n"
    print "Revealed plaintext:"
    print "-----\n"
    print plaintext, "\n"

if __name__ == "__main__":
    ciphertext = "j5a0edj2b"
    chosen_plaintext = "WPI"
    lfsr_init_vector = 63
    keystream = xor_bitstream(bitencode(ciphertext), bitencode(chosen_plaintext))
    plaintext = apply_keystream(lfsr_init_vector, ciphertext)
    print_ciphertext_and_chosen_plaintext(ciphertext, chosen_plaintext)

```



```
print_keystream(keystream)
print_decrypted_message(plaintext)
```

2. با توجه به اینکه طول LFSR برابر 6 است، 6 بیت ابتدایی کلید، همان initialization vector است:
111111

3. با توجه به این عبارات:

$$\begin{aligned} S_5p_5 + S_4p_4 + S_3p_3 + S_2p_2 + S_1p_1 + S_0p_0 &= S_6 \\ S_6p_5 + S_5p_4 + S_4p_3 + S_3p_2 + S_2p_1 + S_1p_0 &= S_7 \\ S_7p_5 + S_6p_4 + S_5p_3 + S_4p_2 + S_3p_1 + S_2p_0 &= S_8 \\ S_8p_5 + S_7p_4 + S_6p_3 + S_5p_2 + S_4p_1 + S_3p_0 &= S_9 \\ S_9p_5 + S_8p_4 + S_7p_3 + S_6p_2 + S_5p_1 + S_4p_0 &= S_{10} \\ S_{10}p_5 + S_9p_4 + S_8p_3 + S_7p_2 + S_6p_1 + S_5p_0 &= S_{11} \end{aligned}$$

، ضرایب به این ترتیب محاسبه می شوند:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{aligned} &= p_0 \\ &= p_1 \\ &= p_2 \\ &= p_3 \\ &= p_4 \\ &= p_5 \end{aligned}$$

4. 'wombat' یک حیوان است که در چند ایالت استرالیا زندگی می کند.

5. Known-plaintext attack

5. با توجه به اینکه یک plaintext و ciphertext معادل آن را داریم، می توانیم keystream نظیر را محاسبه کنیم. به این منظور، باید اعداد اسکی نظیر حروف شرکت کننده در plaintext را به جای آن ها قرار دهیم. طبق رابطه $\text{plaintext} \oplus \text{key} = \text{ciphertext}$ که بر الگوریتم حاکم است، و طبق خواص XOR داریم:

$$\text{plaintext} \oplus \text{ciphertext} = \text{key}$$

Plaintext	B	A	R	A	C	K	O	B	A	M	A
Plaintext ASCII	01000010	01000001	01010010	01000001	01000011	01001011	01001111	01000010	01000001	01001101	01000001
Ciphertext	01000011	00011011	00010010	00110000	11111000	10100111	10001110	11101001	00010100	00011101	01100100
Keystream + Nonce	00000001	01011010	01000000	01110001	10111011	11101100	11000001	10101011	01010101	01010000	00100101

با توجه به اینکه مقدار nonce ها را می دانیم، کلید استفاده شده در عبارت دوم را به دست می آورم:

Keystream + 1	00000001	01011010	01000000	01110001	10111011	11101100	11000001	10101011	01010101	01010000	00100101
Keystream	00000000	01011001	00111111	01110000	10111010	11101011	11000000	10101010	01010100	01001111	00100100
Keystream + 2	00000010	01011011	01000001	01110010	10111100	11101101	11000010	10101100	01010110	01010001	00100110

برای رمزنگاری متن دوم، دوباره از رابطه ی زیر استفاده می کنم:

$$\text{plaintext} \oplus \text{key} = \text{ciphertext} \quad \rightarrow \quad \text{ciphertext} \oplus \text{key} = \text{plaintext}$$

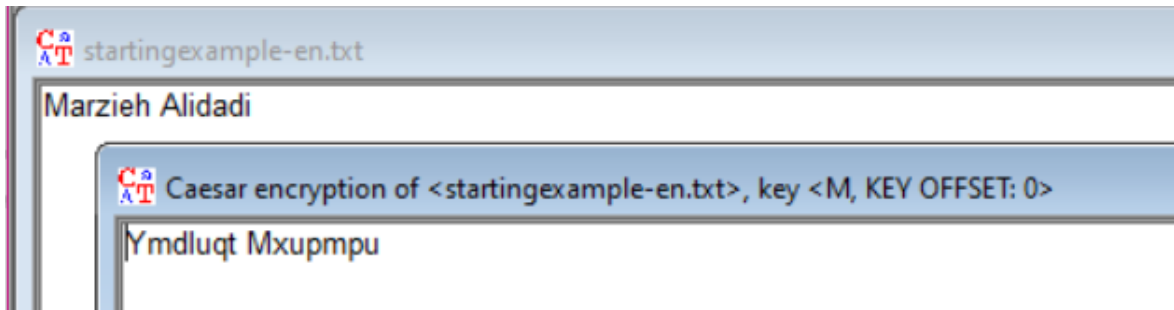
Ciphertext	01000110	00010100	00001111	00110011	11110000	10101001	10010110	11111110	00000011	00011100	01110110
Keystream + 2	00000010	01011011	01000001	01110010	10111100	11101101	11000010	10101100	01010110	01010001	00100110
Plaintext ASCII	01000100	01001111	01001110	01000001	01001100	01000100	01010100	01010010	01010101	01001101	01010000
Plaintext	D	O	N	A	L	D	T	R	U	M	P

حروف نظیر این اعداد اسکی به دست آمده، همان Plaintext مد نظر است و بدین شکل است:

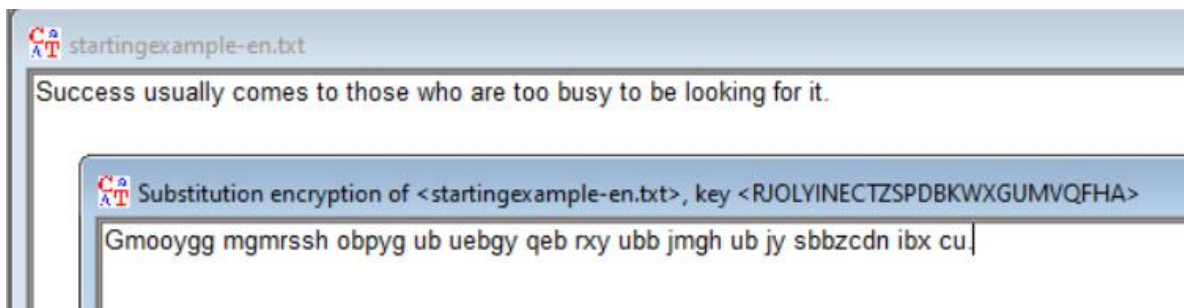
DONALDTRUMP

6.

1. The encrypted text: **Ymdlugt Mxupmpu** → the alphabet shifted by **12** letters.

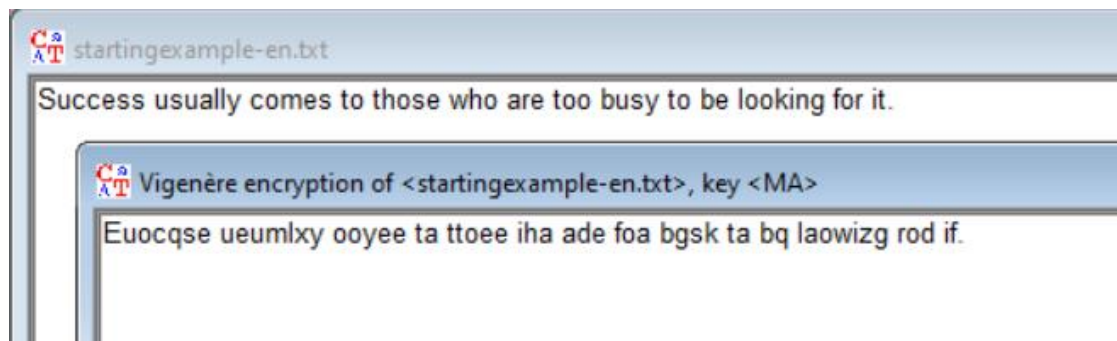


2. $\text{Offset} = 9631983 \% 26 = 23$ → The encrypted text:
Gmooygg mgmrssh obpyg ub uebgy qeb rxy ubb jmgh ub jy sbbzcdn ibx cu.



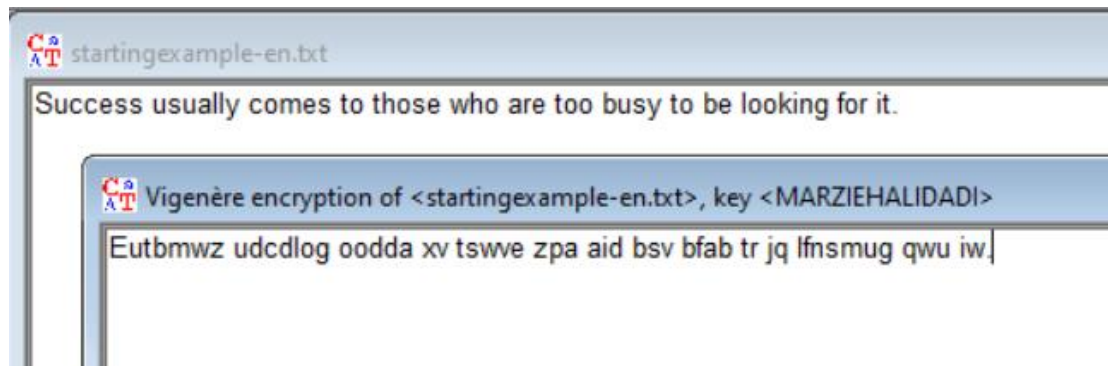
3.

- a) The encrypted text:
Euocqse ueumlxy ooyee ta ttoee iha ade foa bgsk ta bq laowizg rod if.

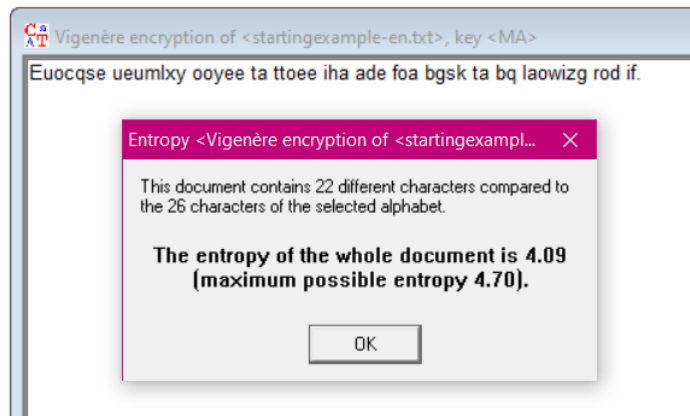


b) The encrypted text:

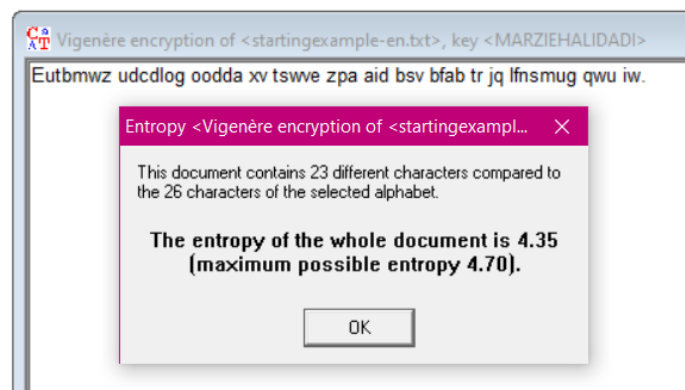
Eutbmwz udcldlog oodda xv tswve zpa aid bsv bfab tr jq lfnsmug qwu iw.



c) First part with key = MA



Second part with key = MARZIEHALIDADI



Entropy refers to the randomness collected by a system for use in algorithms that require random data. A lack of good entropy can leave a cryptosystem vulnerable and unable to encrypt data securely.

So, when the entropy of a cryptosystem is better, it is more secure.

In one method, the longer key results a more secure encryption. As we see, when I used “MA” as key, the entropy is less than when I used “MARZIEHALIDADI” as the key of encryption. So it’s necessary to use a bigger key.

4.

Automatic Vigenère Analysis

Derived key length:

Continue

Cancel

Automatic Analysis

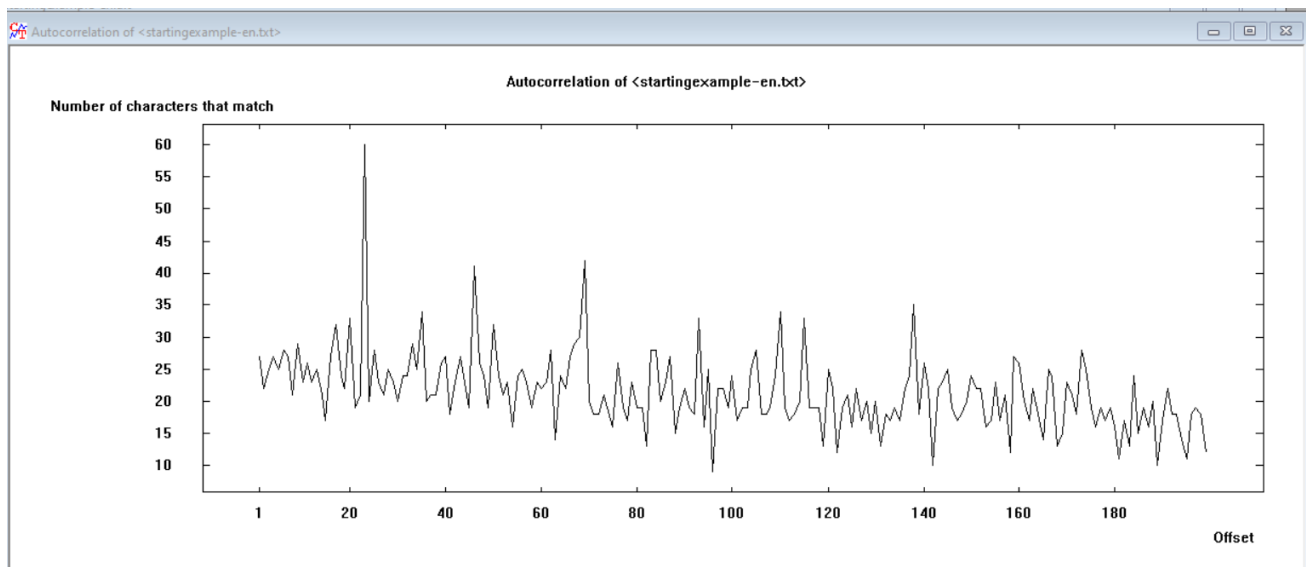
Derived key:

Decrypt

Cancel

Automatic Vigenère Analysis of <startingexample-en.txt>, key: <ISFAHANUNIVOFTECHNOLOGY>

Container-based Virtualization provides a different level of abstraction in terms of virtualization and isolation when compared with hypervisors. In particular, it can be considered as a lightweight alternative to hypervisor-based virtualization. Hypervisors abstract hardware, which results in overhead in terms of virtualizing hardware and virtual device drivers. A full operating system (e.g., Linux) is typically run on top of this virtualized hardware in each virtual machine instance. In contrast, containers implement isolation of processes at the operating system level, thus avoiding such overhead. These containers run on top of the same shared operating system kernel of the underlying host machine, and one or more processes can be run within each container.



- Diagram is showing the **frequency** graph.
- **Plaintext:** Container-based Virtualization provides a different level of abstraction in terms of virtualization and isolation when compared with hypervisors. In particular, it can be considered as a lightweight alternative to hypervisor-based virtualization. Hypervisors abstract hardware, which results in overhead in

a.



c.

- Part a (the long key):

XOR Analysis

Derived key length:

Expected most common character (hex):

Continue
Cancel

XOR Analysis

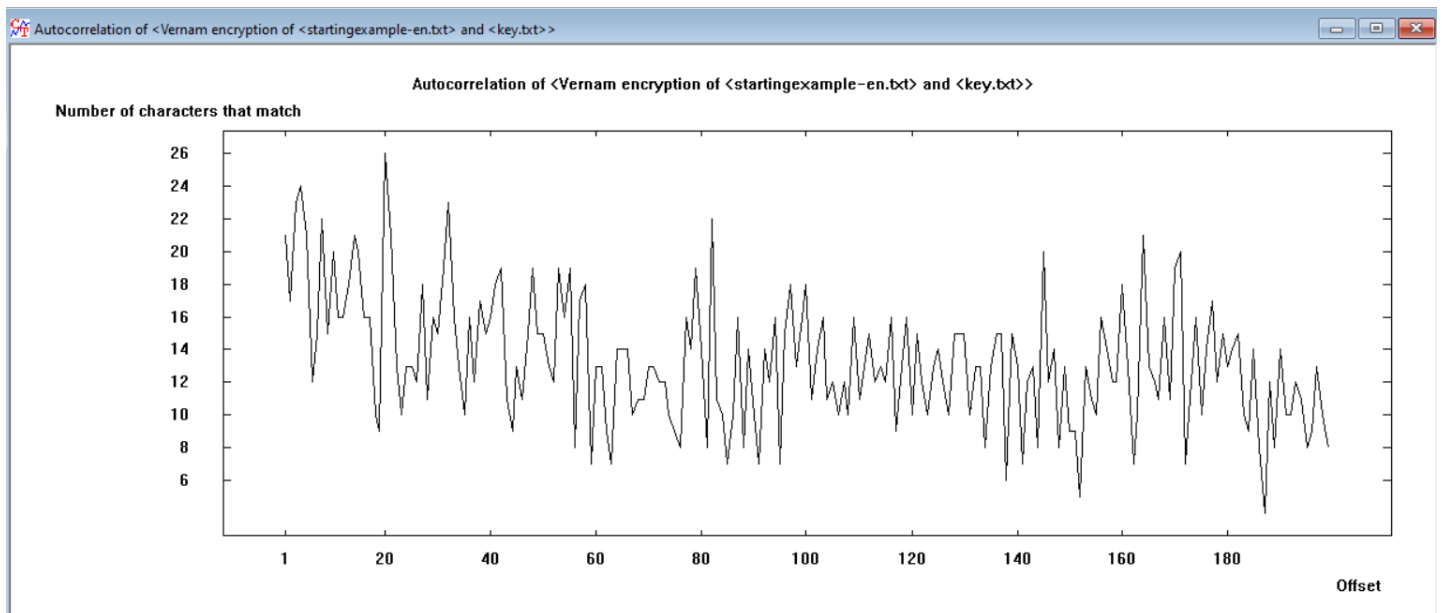
Derived key:

00

Decrypt
Cancel

Automatic XOR Analysis of <Vernam encryption of <startingexample-en.txt> and <key.txt>, key: <00>

00000000	00 07 0F 12 09 00 00 08 08 01 14 14 09 12 1E 00 12 07 1C 07 02 08 10 00 08 16 07 05 18 08 07
00000001	14 1C 00 00 1D 0A 04 17 00 00 1E 1D 14 00 00 07 00 0F 0C 0A 0B 0E 01 0B 00 1E 0A 04 0A 00 0A
00000002	16 16 00 0C 1B 1E 0F 2D 7C 16 09 11 1D 04 49 11 11 14 58 18 45 0B 13 01 1B 08 07 53 0B 4E 13
00000003	03 12 03 18 0E 03 06 4A 18 49 07 08 50 0C 48 11 1B 05 55 1E 45 05 04 0A 14 02 16 03 10 0F 53
00000004	39 4E 3A 03 07 1F 03 13 57 0D 43 12 1D 12 01 47 0A 46 09 11 1E 5E 55 5F 78 69 0D 56 12 15 00	9N:.....W.C..G.F..^U..x..V..
00000005	11 16 49 04 09 14 47 1F 05 4C 1F 7E 7D 41 56 47 13 0B 12 58 45 65 4E 2F 10 03 1F 00 0D 49 42	...I...G...L...JAVG...XEEN?...IB
00000006	4D 4E 02 42 14 49 08 14 03 4B 43 09 7A 48 20 04 53 1B 49 0D 1A 19 16 1E 0B 1E 49 01 45 02 16	MN.B.I...KC..sh..S.I...I.E...
00000007	06 0B 1C 17 10 51 05 48 05 50 0A 45 06 03 4F 0E 62 78 44 1D 1E 52 16 45 06 1E 57 11 1A 09 00Q.H.F.E...O.bsD..R.E..W...
00000008	1A 4E 1B 09 44 62 6F 16 58 19 02 03 14 0F 0C 08 58 12 4F 19 06 56 05 45 00 07 1B 08 07 51 05	..N...Dbo.X.....X.O..V.E...Q...
00000009	4F 4E 06 4F 1E 14 0A 44 0D 48 1D 18 0C 55 07 45 13 03 1C 46 43 7A 48 30 18 51 13 45 10 08 07	ON.O...D.H...U.E...FC=HO.Q.E...
00000010	1D 4D 01 68 69 01 0E 12 14 1E 58 1E 00 46 10 0F 14 12 0B 10 52 17 0B 4D 16 16 43 6B 69 4A 52	..M.hi.....X..F.....R..M..CkIUR
00000011	19 18 59 00 55 07 1B 18 1F 03 13 47 0A 52 1F 1E 18 02 1F 51 02 51 17 1C 17 04 0F 19 1F 4E 52	..Y.U.....G.R.....Q.Q.....NR
00000012	62 78 0B 6C 73 5D 4F 4E 04 45 05 04 0A 14 02 16 03 10 0F 4F 15 62 7F 1B 16 1D 44 10 1C 44 4C	ex..s}ON.E...G.R.....O.b...D..DL
00000013	0E 0B 00 0E 0E 4D 11 42 04 13 54 16 04 09 1D 6A 7E 4B 68 6F 0D 6C 74 09 4E 55 12 42 10 4C 05M.B..T.....j-Kho..ls..NU.B.L.
00000014	4D 1F 41 00 07 4D 10 00 15 4F 1C 18 07 09 49 01 52 03 05 07 14 40 44 4E 48 03 56 0C 52 14 16	M.A..M...O...I.R.....@DNH.V.R...
00000015	0D 4A 0E 4B 0D 4A 06 68 7E 14 14 09 1B 50 01 02 57 48 00 1F 5E 0D 48 04 04 44 4F 17 1A 42 00	..J.K.J.h.....P..WH...E..DO..C...
00000016	7A 6C 02 55 01 16 12 05 06 17 12 1A 1A 04 07 17 10 4A 59 5B 49 0E 0D 54 0E 00 07 4F 00 07 1D	sl.U.....P.E.....E.F....B.NK..eoX.
00000017	01 0C 02 0F 0F 1F 50 1E 45 0E 04 16 1A 00 09 45 0C 46 0B 0B 09 12 42 0B 4E 58 1B 65 6F 58 0F	..V...W..C.....@V().9WR.S...l@Z
00000018	0C 56 01 05 08 57 07 00 43 02 06 06 1F 0F 04 56 28 5D 00 39 57 52 0E 53 01 09 04 5D 6F 7C 5A	h1.e/A.Z.N.H.FE.H.H.I...B.C...
00000019	68 31 0D 65 2F 41 12 5A 04 4E 02 48 09 46 45 11 48 12 48 0A 49 17 0A 04 42 1B 43 06 18 0E 0A	...P.H..K.E..N..ie...C..T...G...
00000020	11 1B 18 50 17 48 18 1F 4B 07 45 14 10 4E 19 69 65 05 02 01 0A 43 1C 0D 54 0D 11 0C 47 01 1E	ER4!vs='R.....KF.C...E.E.FA.kl.
00000021	45 52 34 21 76 73 2D 27 52 0A 17 1B 0F 0A 4B 46 05 43 2D 7E 0C 45 06 06 1C 02 1A 1C 0C 1C 17Z...DW...C.....HE@.S.I.G...
00000022	43 10 54 00 52 05 4F 0C 17 02 0E 1A 0F 0B 5A 01 4E 4E 11 45 02 1B 16 45 0D 46 41 1F 6B 6C 0D	C.T.R.O.....T.NN.E...E.E.FA.kl.
00000023	08 05 04 1E 04 0C 5A 12 08 44 57 1F 00 43 16 05 07 19 15 48 45 3C 40 00 25 1A 49 11 47 16 09C.H.A)f.m7.i.....X..A.A!L
00000024	41 15 1E 16 0B 45 4D 01 49 00 1B 01 45 6A 60 14 4F 07 07 0C 48 10 4F 54 0C 45 18 0E 1A 06 16	2.....S..S.....T..DB..P...
00000025	1D 43 0D 48 10 41 29 66 05 6D 37 7F 69 02 02 0D 01 17 1A 1E 1E 1D 58 1D 11 41 01 41 21 16 4C	..U.....n*.O.....Q.R...VHXOC.(!
00000026	32 03 05 1E 12 17 04 53 1C 16 53 1B 0E 0A 10 01 54 16 14 44 42 0B 02 50 15 02 0E 04 17 16 08	k.....H...SD..T.....@
00000027	02 55 18 0C 00 09 6E 2A 02 4F 09 0A 00 1A 0C 11 0C 1C 51 11 52 1C 07 15 56 48 58 4F 42 04 7B	
00000028	6B 1F 19 1B 1A 0C 48 1D 11 53 44 18 1F 54 0C 19 18 07 0B 1D 0A 07 09 40	



- Part b (the short key):

XOR Analysis
✕

Derived key length:

Expected most common character (hex):

Continue
Cancel

XOR Analysis
✕

Derived key:

6D 41 06 5A 49 16 0D 00 24 4C 0C 44 04 10 1D

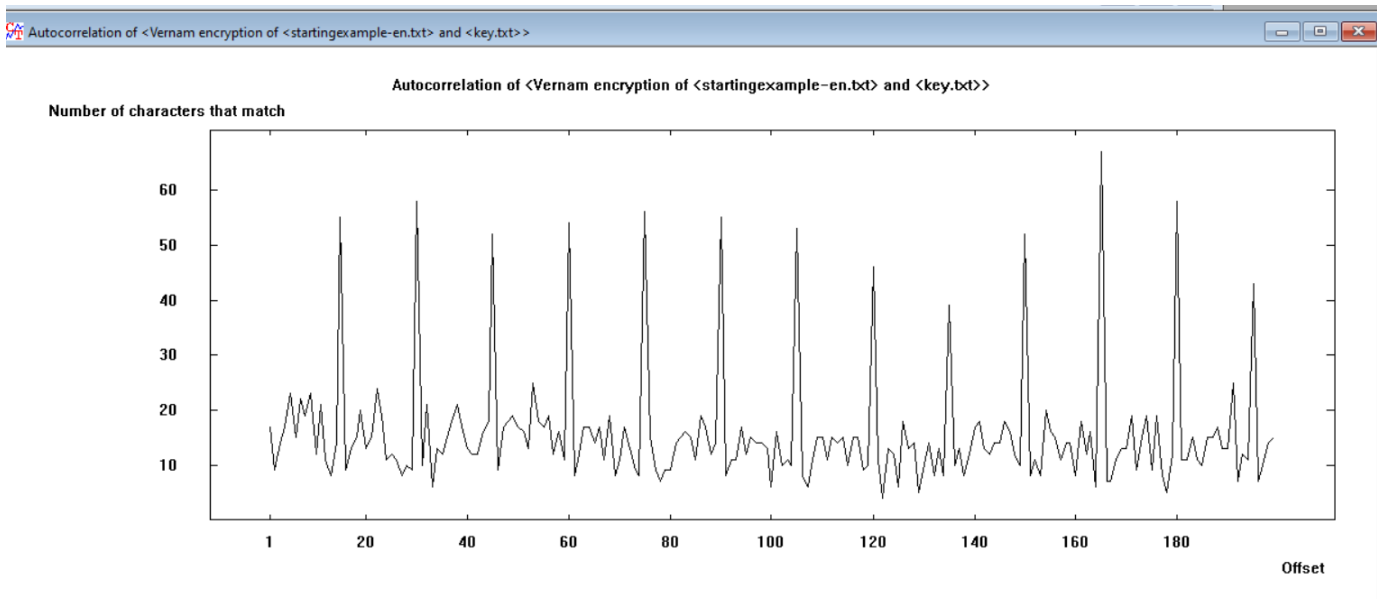
⬆

⬇

Decrypt
Cancel

Automatic XOR Analysis of <Vernam encryption of <startingexample-en.txt> and <key.txt>, key: <6D 41 06 5A 49 16 0D 00 24 4C 0C 44 04 10 1D>
⏏ ⏏ ✕

<pre> 00000000 74 4F 10 41 59 5F 45 69 0B 54 00 52 0B 11 00 00 53 11 52 56 1A 06 45 45 50 17 4F 13 1D 10 45 0000001F 52 07 00 08 3A 36 70 16 09 45 54 17 0D 54 54 4F 54 44 45 1F 0C 56 00 52 45 4D 0A 06 11 00 41 0000003E 1A 44 00 1E 0A 52 00 00 13 41 09 01 11 41 44 10 45 44 53 16 45 17 56 0C 42 00 07 54 49 4E 00 0000005D 45 47 01 04 54 00 44 45 57 0C 00 1C 00 54 1C 45 49 01 45 52 00 53 0C 44 00 1A 00 49 41 18 00 0000007C 69 1D 11 45 17 4E 00 54 45 15 17 43 45 07 53 00 1C 02 46 00 52 49 00 16 01 17 48 00 15 53 00 0000009B 07 17 49 15 4C 00 0D 15 15 15 2D 7E 08 56 1C 0C 43 00 0C 45 69 0B 00 11 52 4E 11 54 0C 53 000000BA 04 4E 01 00 13 49 01 11 1B 09 0E 54 74 45 1A 16 00 16 49 11 55 04 00 1D 4F 4E 54 47 45 1D 00 000000D9 52 04 54 00 53 45 00 1C 45 00 1A 45 45 17 45 46 0A 52 45 4D 0A 06 11 00 50 1B 57 45 01 03 55 000000F8 09 00 04 4E 01 79 7E 45 53 04 45 4E 00 0C 56 00 00 0D 4F 08 11 54 44 45 02 49 43 16 16 00 11 00000117 4F 45 43 0A 02 11 52 00 00 45 45 00 00 0B 45 00 44 16 5A 54 74 45 1D 52 00 17 00 56 0C 42 00000136 00 00 17 11 17 45 49 02 45 53 52 01 49 02 46 00 52 00 1A 00 00 4E 15 4D 45 00 49 2D 6F 46 17 00000155 4F 08 54 17 55 53 00 4F 4D 16 17 00 15 52 00 4D 0C 07 11 00 45 05 55 49 02 08 45 0B 54 45 08 00000174 26 24 21 09 00 00 4F 00 01 00 53 0C 44 00 4E 11 1D 15 4C 00 06 4F 55 07 00 52 45 41 0B 44 45 00000193 00 1B 00 48 1B 4D 45 53 02 41 11 45 12 41 1C 79 7E 08 68 32 77 09 5F 45 42 10 54 45 41 09 18 000001B2 54 48 41 02 45 00 12 45 43 0A 4D 08 4F 0B 54 13 52 4F 01 4E 44 49 45 54 0D 45 45 54 17 15 10 000001D1 45 0D 1B 46 46 53 07 45 11 57 00 45 0B 54 18 4F 57 59 43 4F 00 11 00 04 4E 01 00 17 1D 17 48 000001F0 2D 7E 46 55 1D 06 54 0C 4F 0B 41 09 1D 00 49 45 07 0C 00 04 0C 54 0D 00 04 00 15 1B 00 45 4E 0000020F 00 49 41 1F 09 59 45 4E 00 47 04 00 1D 56 45 54 45 46 15 00 43 11 00 0A 4E 45 00 1C 45 00 10 0000022E 45 56 1A 06 45 45 53 00 42 10 06 1D 54 59 5A 00 61 00 45 41 45 52 00 52 10 18 00 0C 2D 7E 56 0000024D 68 34 22 00 12 41 16 00 0A 1A 11 00 4F 12 00 54 1B 00 00 03 49 17 53 11 54 07 43 45 1A 41 52 0000026C 1A 0A 53 45 54 0D 41 11 54 03 45 52 11 00 41 17 0A 50 11 45 01 00 12 1D 00 48 49 1A 00 54 1B 0000028B 00 00 2B 66 32 00 15 1B 06 41 44 1D 47 4D 5F 45 54 0A 2D 6F 44 00 19 1B 4E 53 00 52 41 07 00 000002AA 00 0C 54 16 00 15 1B 00 45 4E 00 49 41 1F 45 49 0B 00 11 45 17 19 07 00 4F 12 00 45 15 03 49 000002C9 06 49 00 4E 06 0D 54 41 4E 10 00 53 16 06 55 17 49 11 59 4B 54 2D 4E 00 00 48 49 00 45 43 0D 000002E8 41 15 54 00 06 58 00 57 11 00 41 01 00 2D 6F 47 0A 49 0B 12 54 54 4F 54 44 45 00 06 52 0C 42 00000307 00 00 11 1C 11 00 6E 32 76 00 12 17 43 0D 49 11 45 06 00 01 52 45 54 54 45 12 11 00 31 45 09 00000326 45 03 1B 1A 49 43 15 00 44 16 16 49 02 4E 00 44 45 15 1A 44 00 1D 4D 50 1F 00 4D 00 4E 11 45 00000345 01 54 1D 4E 00 15 2D 2A 10 0A 4D 08 45 17 43 0C 15 18 00 54 06 49 41 1F 49 00 11 4F 45 45 13 00000364 15 18 55 41 00 45 00 1A 11 53 45 50 0A 54 00 1A 00 49 41 18 49 54 0A 4B </pre>	<pre> oOAY_Ei.T.R...S.RV...EEP.O...E R...:6p...ET...TOTDE...V.REM...A ...D...R...A...AD.EDS.E.V.C...TIN. EG...T.DEW...T.EI.ER.S.D...IA.. ...E.N.TE...CE.S...F.RI...H...S. ...I.L...Y...V...C...Ei...RN.T.S ...N...I...T...H...I.U...ONTGE... R.T.S...E...EE.EF.REM...P.WE...U ...N.y-EX...EN...V...O...TDE.IC... OEC...R...HE...E.D.StH.S...V.C ...Ei.ESS.I.F.R...N.ME.I-oF. O.T.US.OM...R.M...E.UI...E.TE. 421...O...S.D.N...L...OU...REA.DE ...H.MES.A.E.A.y...h3w...ES.TEA... THA.E...EC.M.O.T.RO.NDIET.EET... E...FFS.E.W.E.T.ONVCO...N...H ...FU...T.O.A...IE...T...EN ...IA...YEN.G...VETEF...C...NE...E EV...EES.C...TYZ...A.EAER.S...~V h42...A...O...T...I.S.T.CE.AR ...SET.A.T.ER...A...P.E...HI...T. ...+ES...AD.GM...ET...oD...NS.RA... ...T...EN...IA.EI...E...O...E...I ...I.N...TAN...S...U.I.YKT=N...HI.EC. A.T...X.W...A...oG.I...TOTDE...R.Bn2w...C.I.E...RETH...1E. E...IC...D...I.N.DE...D.MP...M.N.E .T.N...*...M.E.C...T.TIA.I...OEE. ...UA.E...SEP.T...IA.IT.K </pre>
---	---



همانطور که مشخص است، در حالت اول که طول کلید بزرگتر مساوی طول plaintext بود، رمز شکسته نشد و طول کلید قابل حدس نبود و متن اصلی به دست نیامد. این نشان دهنده ی امنیت بالا در این حالت است.

در حالت دوم که طول کلید خیلی کوتاه بود نسبت به طول متن، رمز خیلی راحت تر شکسته شد و بخش قابل توجهی از متن اصلی به دست آمد. این نشان دهنده ی امنیت پایین در این حالت است.