Лабораторная работа 8

Генералов Даниил, 1032212280

2024 г.

Российский университет дружбы народов, Москва, Россия

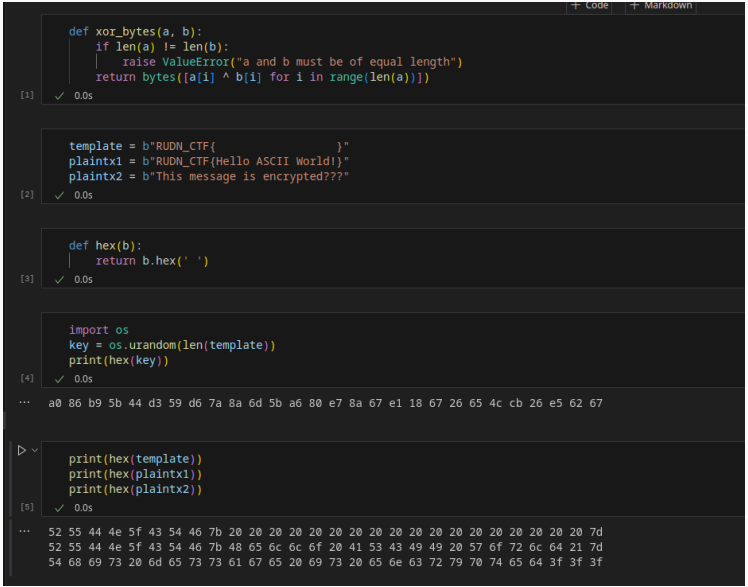
Задание

Задание

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

Выполнение

Python


+ Code
+ Markdown
def xor_bytes(a, b):
if len(a) != len(b):
raise ValueError("a and b must be of equal length")
return bytes([a[i] ^ b[i] for i in range(len(a))])
[1]
0.0s
template = b"RUDN_CTF{ }"
plaintx1 = b"RUDN_CTF{Hello ASCII World!}"
plaintx2 = b"This message is encrypted???"
[2]
0.0s
def hex(b):
return b.hex(' ')
[3]
0.0s
import os
key = os.urandom(len(template))
print(hex(key))
[4]
0.0s
a0 86 b9 5b 44 d3 59 d6 7a 8a 6d 5b a6 80 e7 8a 67 e1 18 67 26 65 4c cb 26 e5 62 67
print(hex(template))
print(hex(plaintx1))
print(hex(plaintx2))
[5]
0.0s
52 55 44 4e 5f 43 54 46 7b 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 7d
52 55 44 4e 5f 43 54 46 7b 48 65 6c 6c 6f 20 41 53 43 49 49 20 57 6f 72 6c 64 21 7d
54 68 69 73 20 6d 65 73 73 61 67 65 20 69 73 20 65 6e 63 72 79 70 74 65 64 3f 3f 3f

Python

```
cipher1 = xor_bytes(plaintx1, key)
cipher2 = xor_bytes(plaintx2, key)
print(hex(cipher1))
print(hex(cipher2))
```

[6] ✓ 0.0s

```
f2 d3 fd 15 1b 90 0d 90 01 c2 08 37 ca ef c7 cb 34 a2 51 2e 06 32 23 b9 4a 81 43 1a
f4 ee d0 28 64 be 3c a5 09 eb 0a 3e 86 e9 94 aa 02 8f 7b 15 5f 15 38 ae 42 da 5d 58
```

```
both_ciphers = xor_bytes(cipher1, cipher2)
print(hex(both_ciphers))
```

[7] ✓ 0.0s

```
06 3d 2d 3d 7f 2e 31 35 08 29 02 09 4c 06 53 61 36 2d 2a 3b 59 27 1b 17 08 5b 1e 42
```

```
both_plus_template = xor_bytes(both_ciphers, template)
print(hex(both_plus_template))
```

[8] ✓ 0.0s

```
54 68 69 73 20 6d 65 73 73 09 22 29 6c 26 73 41 16 0d 0a 1b 79 07 3b 37 28 7b 3e 3f
```

```
def ascii_filter_unprintable(b):
    text = ''
    for c in b:
        if c < 32 or c > 126:
            text += '~'
        else:
            text += chr(c)
    return text

print(ascii_filter_unprintable(both_plus_template))
```

[9] ✓ 0.0s

```
This mess~"]l&sA~~~~y~:7{{>?
```

```
guess2 = b'This message &sA~~~~y~;7({>?'
```
[10] ✓ 0.0s

```
def textdiff(a, b):
    print(ascii_filter_unprintable(a))
    chars = ''
    for q,w in zip(a,b):
        if q != w:
            chars += '|'
        else:
            chars += ' '
    print(chars)
    print(ascii_filter_unprintable(b))

textdiff(template, guess2)
```
[11] ✓ 0.0s

```
RUDN_CTF{                }
||||||||||| ||||||||||||||
This message &sA~~~~y~;7({>?
```

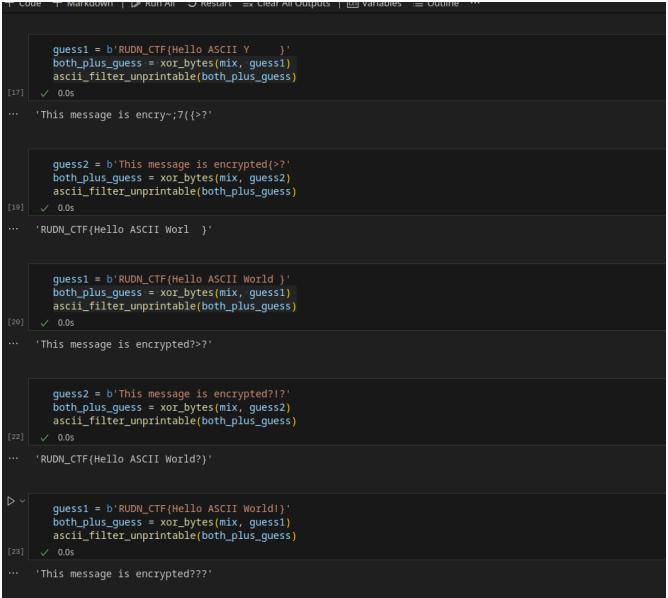
```
mix = xor_bytes(cipher1, cipher2)

both_plus_guess = xor_bytes(mix, guess2)

textdiff(both_plus_guess, both_plus_template)
```
[14] ✓ 0.0s

```
RUDN_CTF{Hell   HSTE Y     }
||||||||||| ||||||||||||||
This mess~")l&sA~~~~y~;7({>?
```

```
guess1 = b'RUDN_CTF{Hello  HSTE Y     }'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
```
[15] ✓ 0.0s

```
'This message isA~~~~y~;7({>?'
```

Python


+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...
guess1 = b'RUDN_CTF{Hello ASCII Y }'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
[17] 0.0s
'This message is encry~;7({>?'
guess2 = b'This message is encrypted{>?'
both_plus_guess = xor_bytes(mix, guess2)
ascii_filter_unprintable(both_plus_guess)
[19] 0.0s
'RUDN_CTF{Hello ASCII Worl }'
guess1 = b'RUDN_CTF{Hello ASCII World)'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
[20] 0.0s
'This message is encrypted?>?'
guess2 = b'This message is encrypted?!?'
both_plus_guess = xor_bytes(mix, guess2)
ascii_filter_unprintable(both_plus_guess)
[22] 0.0s
'RUDN_CTF{Hello ASCII World?}'
guess1 = b'RUDN_CTF{Hello ASCII World!}'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
[23] 0.0s
'This message is encrypted???'

Выводы

Выводы

В этой лабораторной работе мы рассмотрели алгоритм однократного гаммирования и показали его слабость в случае, когда один и тот же ключ используется больше одного раза.