

Question:

Python (CTR mode) vs. Online Calculator (CBC Mode) ::

- Please compare the results of using the following two approaches of AES (CTR Mode & CBC Mode) to encrypt and decrypt the message "Coronavirus" by using the key '01234567890123456789012345678901'.

- [Python](#) - CTR Mode
- [AES Online Calculator](#) - CBC Mode

Ans:

Using online tool

The Key Size in Bits affect the Secret Key's length, if choice the '128', the Secret length is 16, if choice the '256', the Secret length is 32, based on our Secret key we should choice the '256'

### AES Online Encryption

Enter text to be Encrypted

Coronavirus

OR

选择文件 未选择任何文件

Select Mode

CBC

Key Size in Bits

128

Enter IV (Optional)

Enter initialization vector

Enter Secret Key

0123456789012345

Output Text Format: Base64 OHex

Encrypt

AES Encrypted Output:

/g0vVQ/9cvSmnDtff/ep3A==

### AES Online Decryption

Enter text to be Decrypted

/g0vVQ/9cvSmnDtff/ep3A==

Input Text Format: Base64 OHex

Select Mode

CBC

Enter IV Used During Encryption(Optional)

Enter initialization vector

Key Size in Bits

128


Enter Secret Key

0123456789012345

Decrypt

AES Decrypted Output (Base64):

Q29yb25hdmlkdXM=

 **Dedicated Hosting**  
From \$129/Mo

Decode to Plain Text

Coronavirus

## AES Online Encryption

Enter text to be Encrypted

Coronavirus

OR

选择文件 未选择任何文件

Select Mode

CBC

Key Size in Bits

256

Enter IV (Optional)

Enter initialization vector

Enter Secret Key

01234567890123456789012345678901

Output Text Format: ☒ Base64 ☐ Hex

Encrypt

AES Encrypted Output:

iU+yUXJ9oUfNotSgiUpXFA==

## AES Online Decryption

Enter text to be Decrypted

iU+yUXJ9oUfNotSgiUpXFA==

Input Text Format: ☒ Base64 ☐ Hex

Select Mode

CBC

Enter IV Used During Encryption(Optional)

Enter initialization vector

Key Size in Bits

256

Enter Secret Key

01234567890123456789012345678901

Decrypt

AES Decrypted Output (Base64):

Q29yb25hdmlydXM=



Dedicated Hosting  
From \$129/Mo



Decode to Plain Text

Coronavirus

**Python code:** (using ideone)

```
import binascii
from Crypto.Cipher import AES
from Crypto.Util import Counter
from Crypto import Random
```

```
# AES supports multiple key sizes: 16 (AES128), 24 (AES192),
# or 32 (AES256).
```

```
key_bytes = 32
```

```
# You need to define 32-byte key. Like
```

```
# Key = '12345678901234567890123456789012'
```

# Please refer this page on how to create key.

# Otherwise you will get this error.

# Takes as input a 32-byte key and an arbitrary-length

# plaintext and returns a pair (iv, ciphertext). "iv" stands for

# initialization vector.

```
def encrypt(key, plaintext):
```

```
    assert len(key) == key_bytes
```

```
    # Choose a random, 16-byte IV.
```

```
    iv = Random.new().read(AES.block_size)
```

```
    # Convert the IV to a Python integer.
```

```
    iv_int = int(binascii.hexlify(iv), 16)
```

```
    # Create a new Counter object with IV = iv_int.
```

```
    ctr = Counter.new(AES.block_size * 8, initial_value=iv_int)
```

```
    # Create AES-CTR cipher.
```

```
    aes = AES.new(key, AES.MODE_CTR, counter=ctr)
```

```
    # Encrypt and return IV and ciphertext.
```

```
    ciphertext = aes.encrypt(plaintext)
```

```
    return (iv, ciphertext)
```

# Takes as input a 32-byte key, a 16-byte IV, and a ciphertext,

# and outputs the corresponding plaintext.

```
def decrypt(key, iv, ciphertext):
```

```
    assert len(key) == key_bytes
```

```
    # Initialize counter for decryption. iv should be
```

```
    # the same as the output of encrypt().
```

```
    iv_int = int(iv.encode('hex'), 16)
```

```
    ctr = Counter.new(AES.block_size * 8, initial_value=iv_int)
```

```
    # Create AES-CTR cipher.
```

```
    aes = AES.new(key, AES.MODE_CTR, counter=ctr)
```

```
    # Decrypt and return the plaintext.
```

```
    plaintext = aes.decrypt(ciphertext)
```

```
    return plaintext
```

```
key = '01234567890123456789012345678901'
```

```
(iv, ciphertext) = encrypt(key, 'Coronavirus')
print ("cipheretxt is: ", encrypt(iv, ciphertext))
print ("plaintext is: ", decrypt(key, iv, ciphertext))
```

```
39. def decrypt(key, iv, ciphertext):
40.     assert len(key) == key_bytes
41.
42.     # Initialize counter for decryption. iv should be
43.     # the same as the output of encrypt().
44.     iv_int = int(iv.encode('hex'), 16)
45.     ctr = Counter.new(AES.block_size * 8, initial_value=iv_int)
46.
47.     # Create AES-CTR cipher.
48.     aes = AES.new(key, AES.MODE_CTR, counter=ctr)
49.
50.     # Decrypt and return the plaintext.
51.     plaintext = aes.decrypt(ciphertext)
52.     return plaintext
53.
54. key = '01234567890123456789012345678901'
55. (iv, ciphertext) = encrypt(key, 'Coronavirus')
56. print ("cipheretxt is: ", encrypt(iv, ciphertext))
57. print ("plaintext is: ", decrypt(key, iv, ciphertext))
```


Success #stdin #stdout 0.02s 11544KB

 comments (0)

 stdin

 copy

Standard input is empty

 stdout

 copy

```
('cipheretxt is: ', ('\x15\x83\xb2\x19\xeeL\x1e\xcb\x9c\x9co\xe7\x13w(M', 'cn\x98K\xc
c$\xc81\x11\xef\xf3'))
('plaintext is: ', 'Coronavirus')
```

Compare those two's result, their models are different, thus the ciphertext is different. CTR can ignore the iv, and it is more secure (need to decoding after decryption).