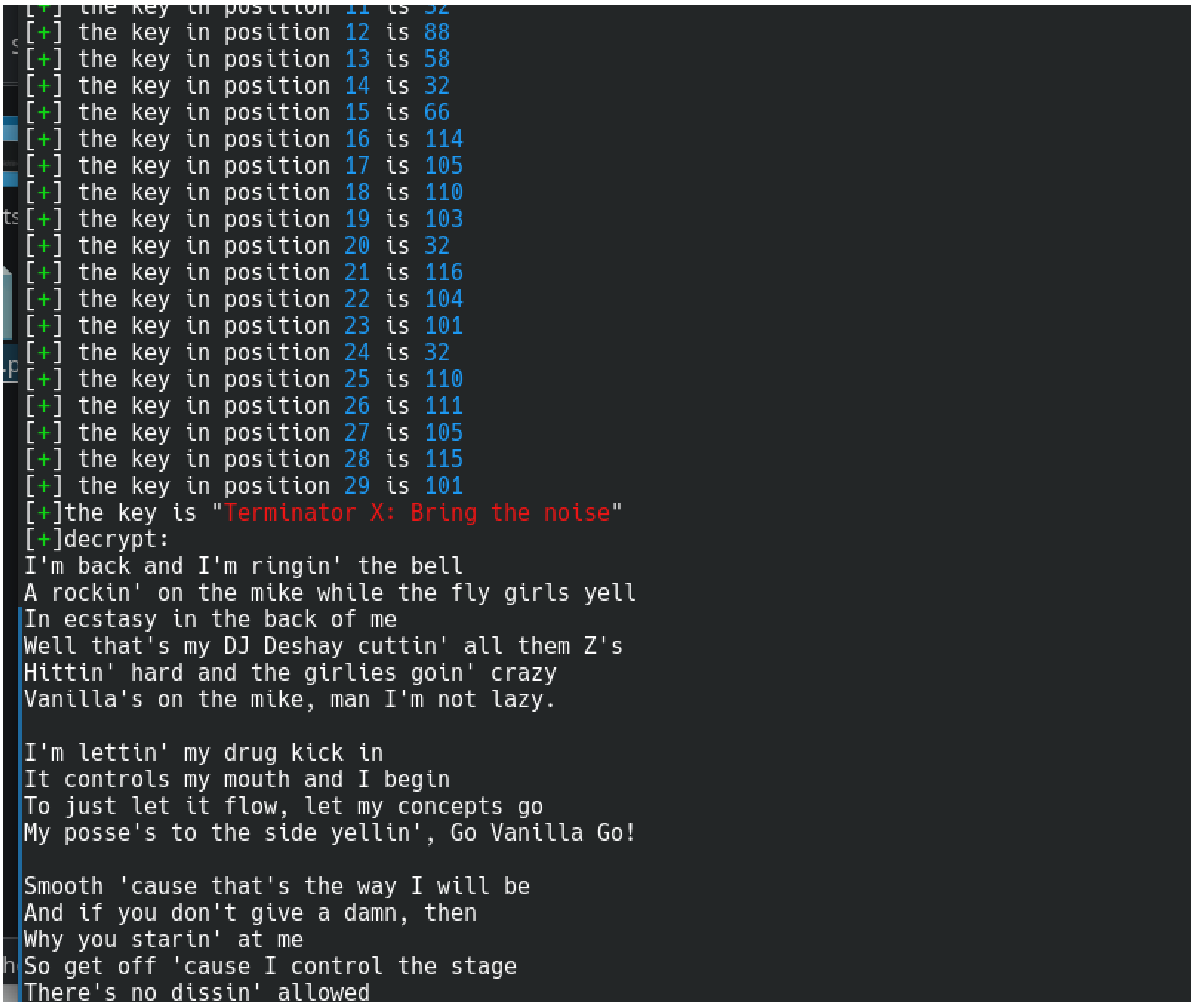

```
[+] the key in position 11 is 32
[+] the key in position 12 is 88
[+] the key in position 13 is 58
[+] the key in position 14 is 32
[+] the key in position 15 is 66
[+] the key in position 16 is 114
[+] the key in position 17 is 105
[+] the key in position 18 is 110
[+] the key in position 19 is 103
[+] the key in position 20 is 32
[+] the key in position 21 is 116
[+] the key in position 22 is 104
[+] the key in position 23 is 101
[+] the key in position 24 is 32
[+] the key in position 25 is 110
[+] the key in position 26 is 111
[+] the key in position 27 is 105
[+] the key in position 28 is 115
[+] the key in position 29 is 101
[+]the key is "Terminator X: Bring the noise"
[+]decrypt:
I'm back and I'm ringin' the bell
A rockin' on the mike while the fly girls yell
In ecstasy in the back of me
Well that's my DJ Deshay cuttin' all them Z's
Hittin' hard and the girlies goin' crazy
Vanilla's on the mike, man I'm not lazy.

I'm lettin' my drug kick in
It controls my mouth and I begin
To just let it flow, let my concepts go
My posse's to the side yellin', Go Vanilla Go!

Smooth 'cause that's the way I will be
And if you don't give a damn, then
Why you starin' at me
So get off 'cause I control the stage
There's no dissin' allowed
```

密钥:

```
Terminator X: Bring the noise
```

```python
from binascii import a2b_base64
from numpy import average
ENDC = '\033[0m'
CRED    = '\33[31m'
CGREEN  = '\33[32m'
CBLUE   = '\33[34m'
INF = "[" + CGREEN + "+" + ENDC + "]"
avail_char = "".join(chr(i) for i in range(32, 126))
CHARACTER_FREQ = {
    'a': 0.0651738, 'b': 0.0124248, 'c': 0.0217339, 'd': 0.0349835, 'e': 0.1041442, 'f': 0.0197881, 'g': 0.0158610,
    'h': 0.0492888, 'i': 0.0558094, 'j': 0.0009033, 'k': 0.0050529, 'l': 0.0331490, 'm': 0.0202124, 'n': 0.0564513,
    'o': 0.0596302, 'p': 0.0137645, 'q': 0.0008606, 'r': 0.0497563, 's': 0.0515760, 't': 0.0729357, 'u': 0.0225134,
    'v': 0.0082903, 'w': 0.0171272, 'x': 0.0013692, 'y': 0.0145984, 'z': 0.0007836, ' ': 0.1918182
}


```

```python
def bytes_to_long(b):
    n = 0
    for i in b:
        n <<= 8
        n += i
    return n


def hamming_dist(a, b):
    ans = 0
    if len(a) > len(b):
        a, b = b, a
    while (len(a) < len(b)):
        a = a + b"\x00"
    l = len(a) * 8
    a, b = bytes_to_long(a), bytes_to_long(b)
    for i in range(l):
        ans += (a ^ b) & 0x1
        a >>= 1
        b >>= 1
    return ans


def load_ciphertext():
    with open("6.txt", "r") as f:
        text = f.read()
    text = text.replace("\n", "")
    return a2b_base64(text.encode())


def decrypt(ct, key):
    plain_text = ""
    for i in range(len(ct)):
        plain_text += chr(ct[i] ^ key[i % len(key)])
    return plain_text



print(INF +" loaded cipher text.")
ct = load_ciphertext()
print(INF + " calculate block hamming dist.")
min_keysize = 0
min_dist = 0xffff
for keysize in range(2, 40):
    dist_list = []
    example_block = ct[0:keysize]
    for i in range(1, len(ct) // keysize):
        target_block = ct[keysize * i : keysize * (i + 1)]
        dist_list.append(hamming_dist(example_block, target_block) / keysize)
    ans = average(dist_list)
    if ans < min_dist:
        min_keysize = keysize
        min_dist = ans
    print(f"  average hamming distance of KEYSIZE={keysize} is " + CBLUE + f"{ans}" + ENDC + ".")
print(INF + f" the min distance is "+ CBLUE + f"{min_dist}" + ENDC + " of KYSIZE="+ CBLUE + f"{min_keysize}" + ENDC)



print(INF + f" using KEYSIZE {min_keysize}, guessing the key.")
key = b""
for pos in range(min_keysize):
    max_ans = 0
    ans_k = 0
    for k in range(255):
        ans = 0
        for j in range(len(ct) // min_keysize):
            if chr(ct[j* min_keysize + pos] ^ k).lower() in CHARACTER_FREQ:
                ans += CHARACTER_FREQ[chr(ct[j* min_keysize + pos] ^ k).lower()]
        if ans > max_ans:
            ans_k = k
            max_ans = ans
    key += chr(ans_k).encode()
    print(INF + " the key in position "+ CBLUE + f"{pos + 1}" + ENDC + " is "+ CBLUE + f"{ans_k}" + ENDC)
print(INF + "the key is \"" + CRED + key.decode() + ENDC + "\"")
print(INF + "decrypt:\n" + decrypt(ct, key))
```