Some Encryption Attacks

CMSC 28400, Autumn 2021, Lecture 10

David Cash

University of Chicago

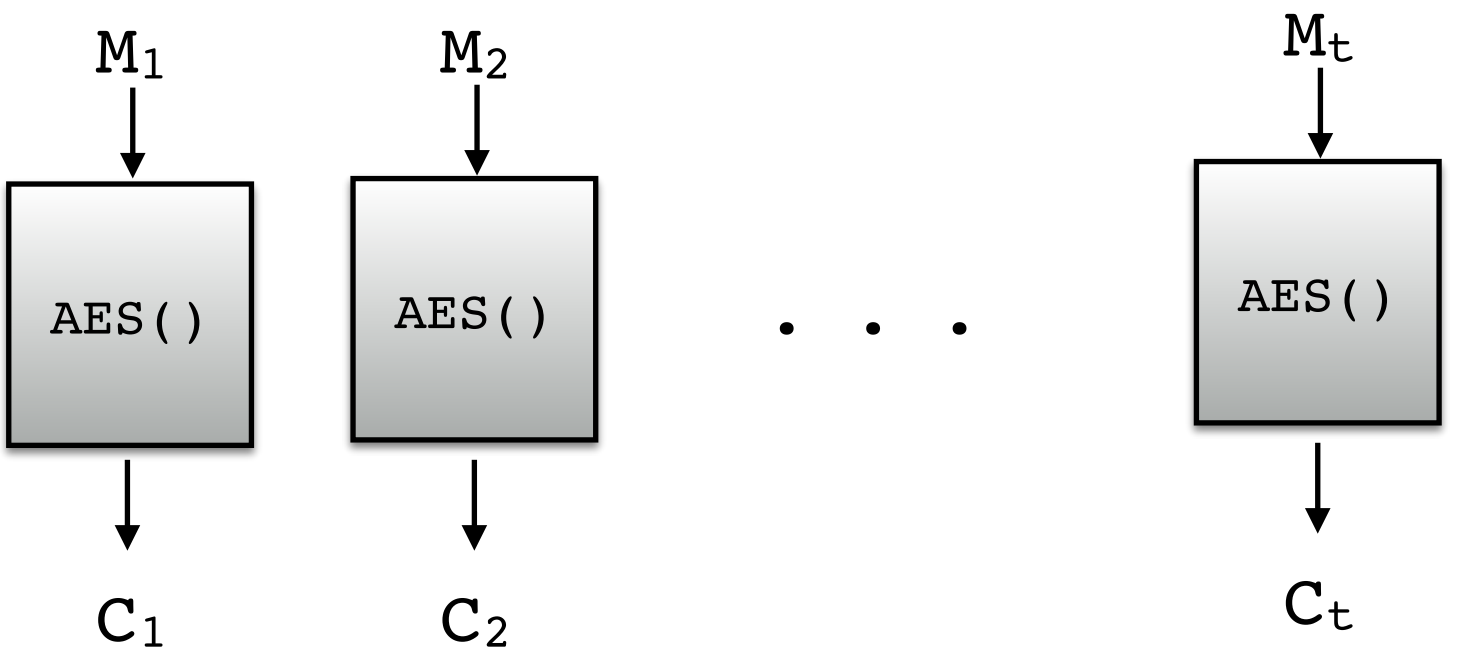
Two Attacks

1. ECB: The Penguin, and Worse
2. Compression + CTR = Insecure

ECB

AES-ECB(K,M)

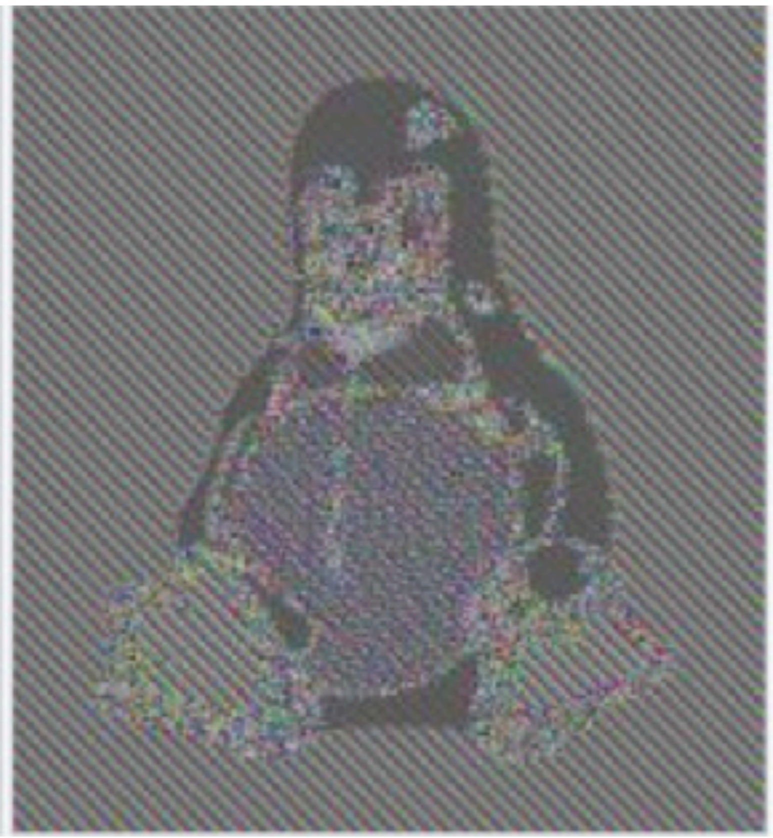
- Parse M into blocks $M_1, M_2, \ldots, M_t$
  // all blocks except M_t are 16 bytes
- Pad M_t up to 16 bytes
- For $i=1\ldots t$:
  - $C_i \leftarrow \mathrm{AES}(K, M_i)$
- Return $C_1, \ldots, C_t$



The ECB Penguin

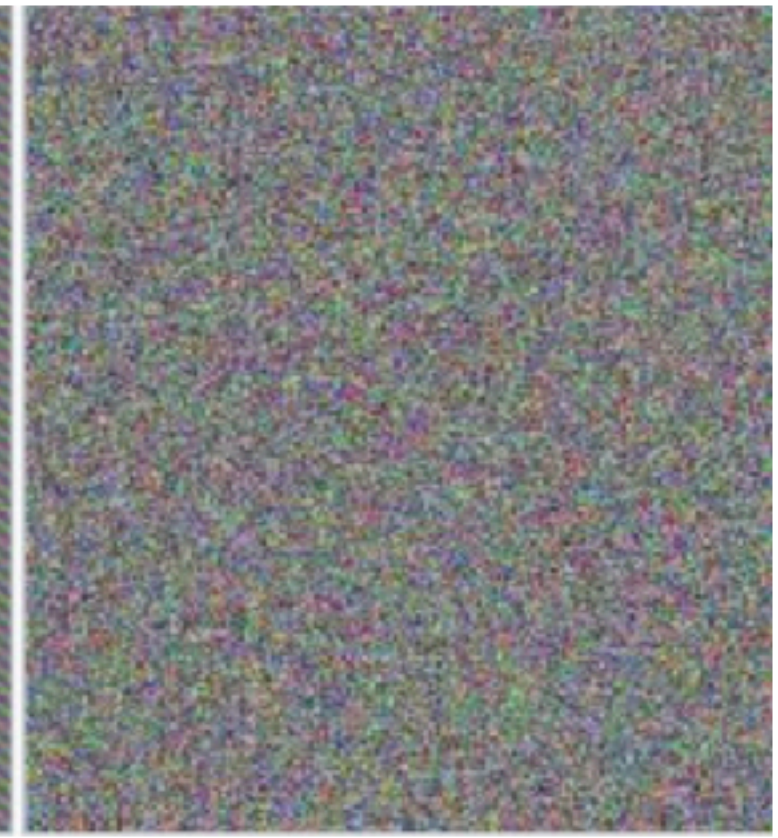
Plaintext

ECB Ciphertext



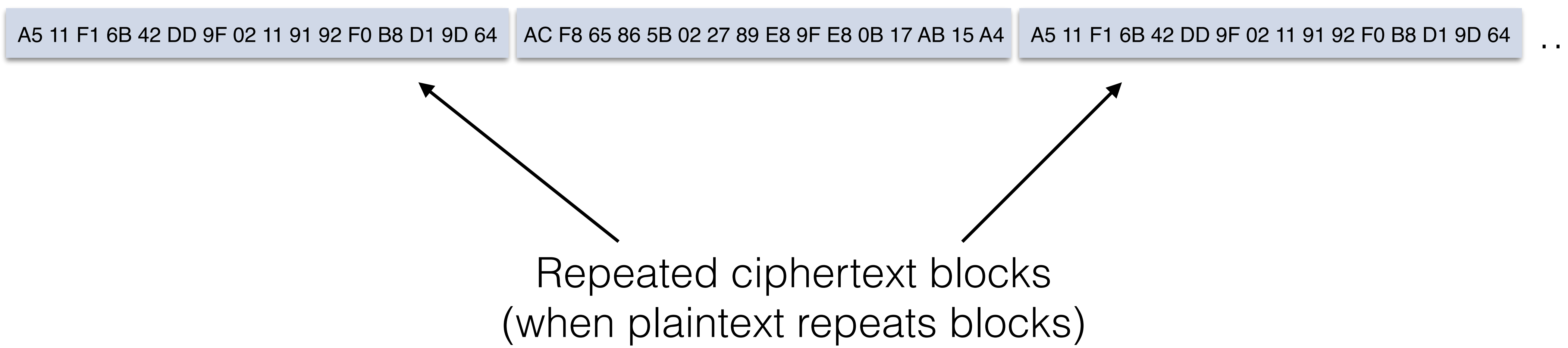
No

CTR Ciphertext



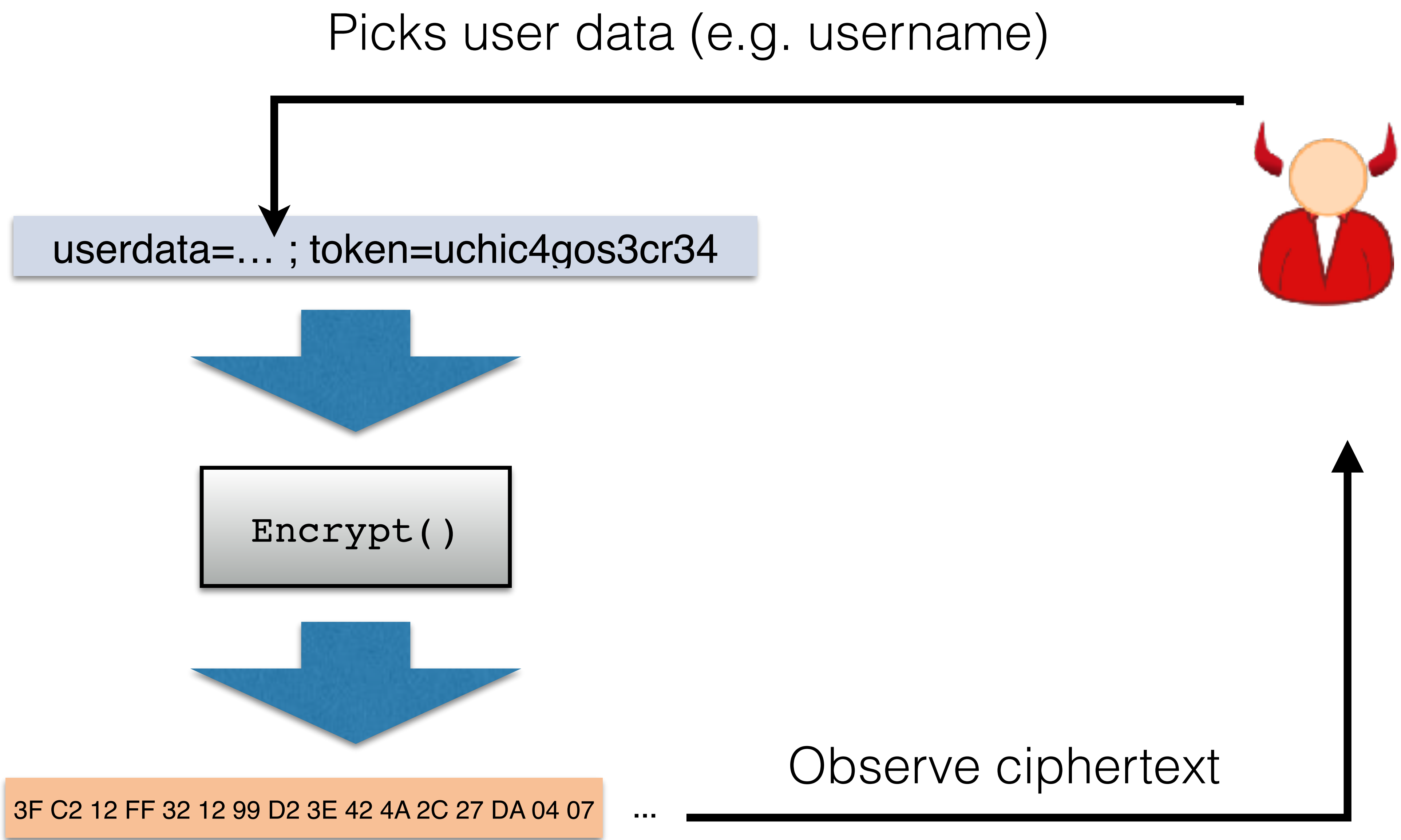
Looks random

Detecting ECB in Practice



- Good modes like CTR and CBC basically never repeat a block

CPAs and Partial Plaintext Recovery



- Goal: Recover token
- Capability: Submit user data, observe ciphertext (many times)

PKCS7 Padding

- Need to pad a byte string up to a multiple of 16 bytes
- First look at how many bytes are missing. Here, need 10 bytes
- Fill missing k bytes with value k (k = 10 = 0x0A in example)

```
48 65 6C 6C 6F 20 43 53 31 37 37 20 73 74 75 64 | 65 6E 74 73 21 00 | 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
```

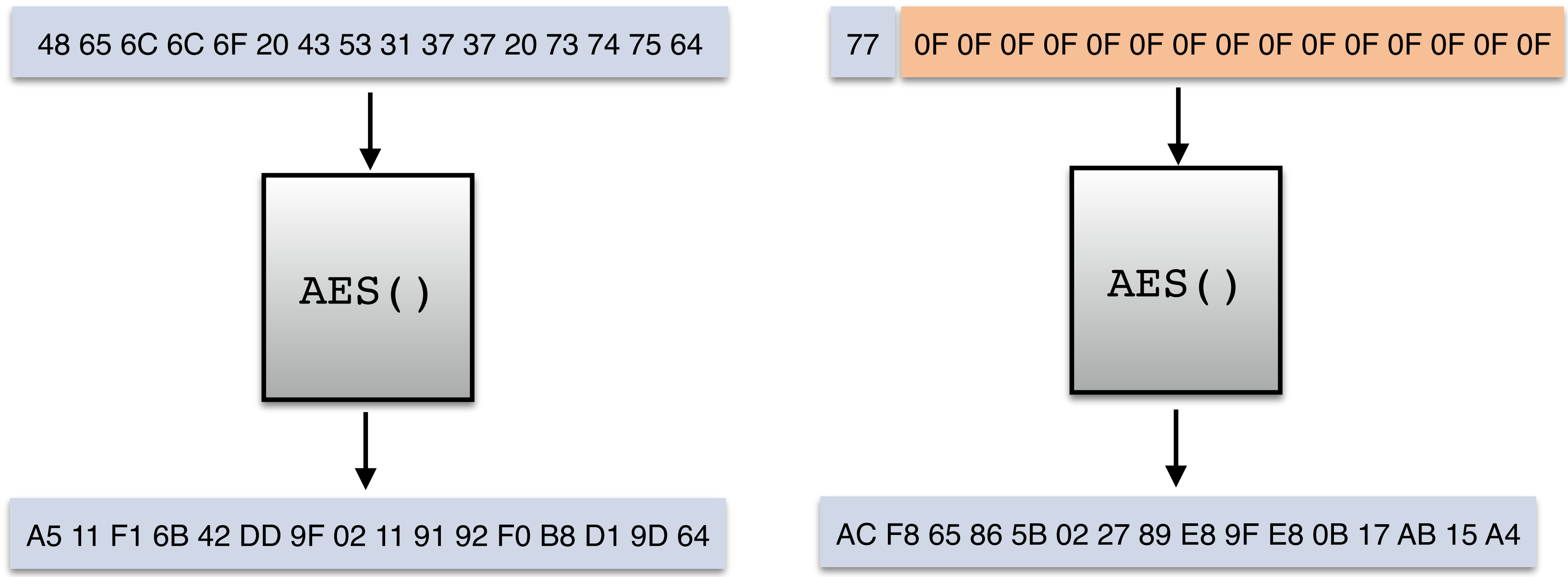
- If data is already a multiple of 16 bytes long, add an entire block of 0x10 bytes

```
48 65 6C 6C 6F 20 43 53 31 37 37 20 73 74 02 02 | 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

Can't leave data unchanged;
Bytes might be interpreted as padding.

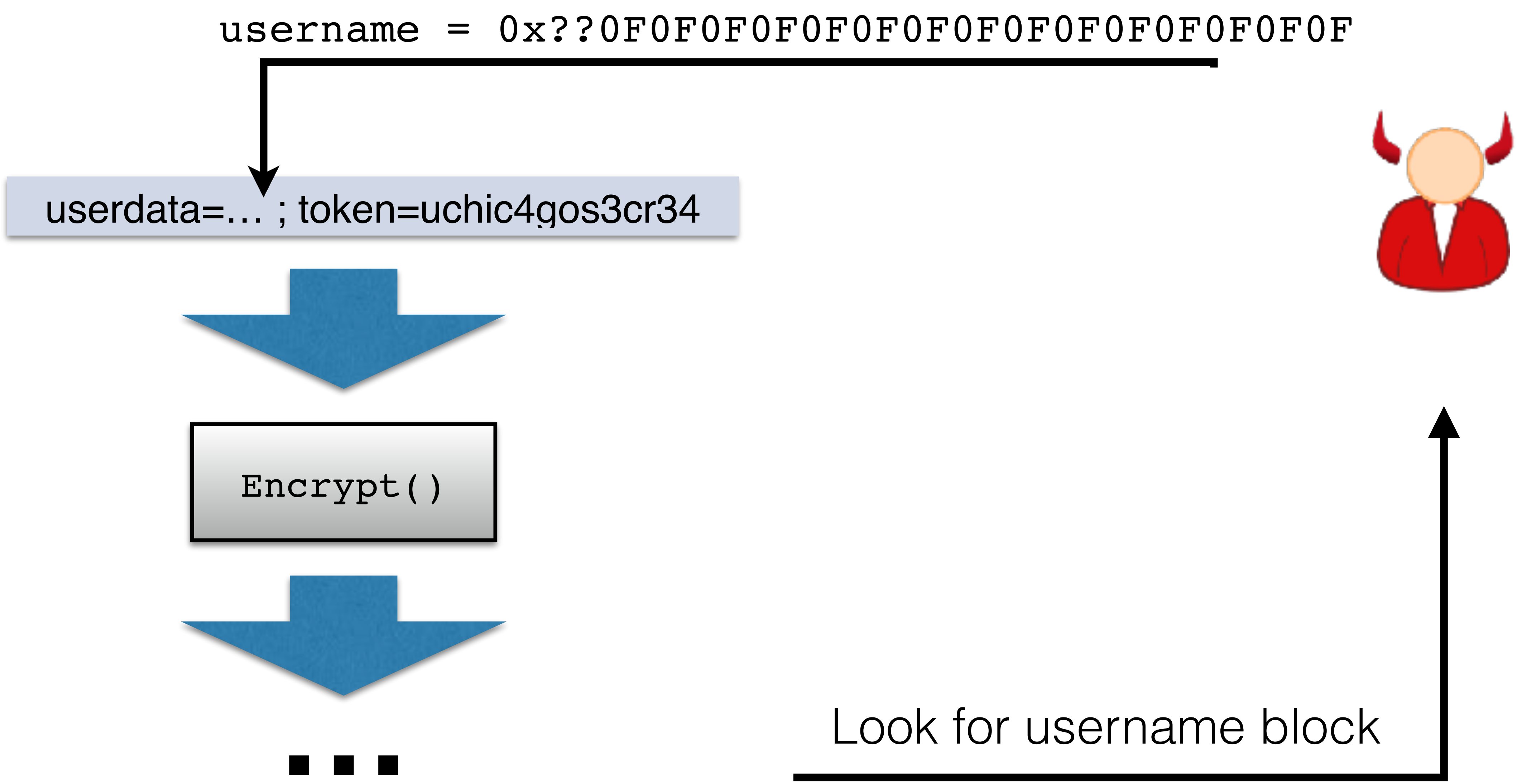
- Un-padding is easy
- Sometimes un-padding will throw error (ex: message ends in … `06 03 03`)

ECB and Partial Plaintext Recovery: One Byte



- Plan: Learn AES(K, **??**0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F) for all 256 values of **??**.
- Easily read off value of last byte

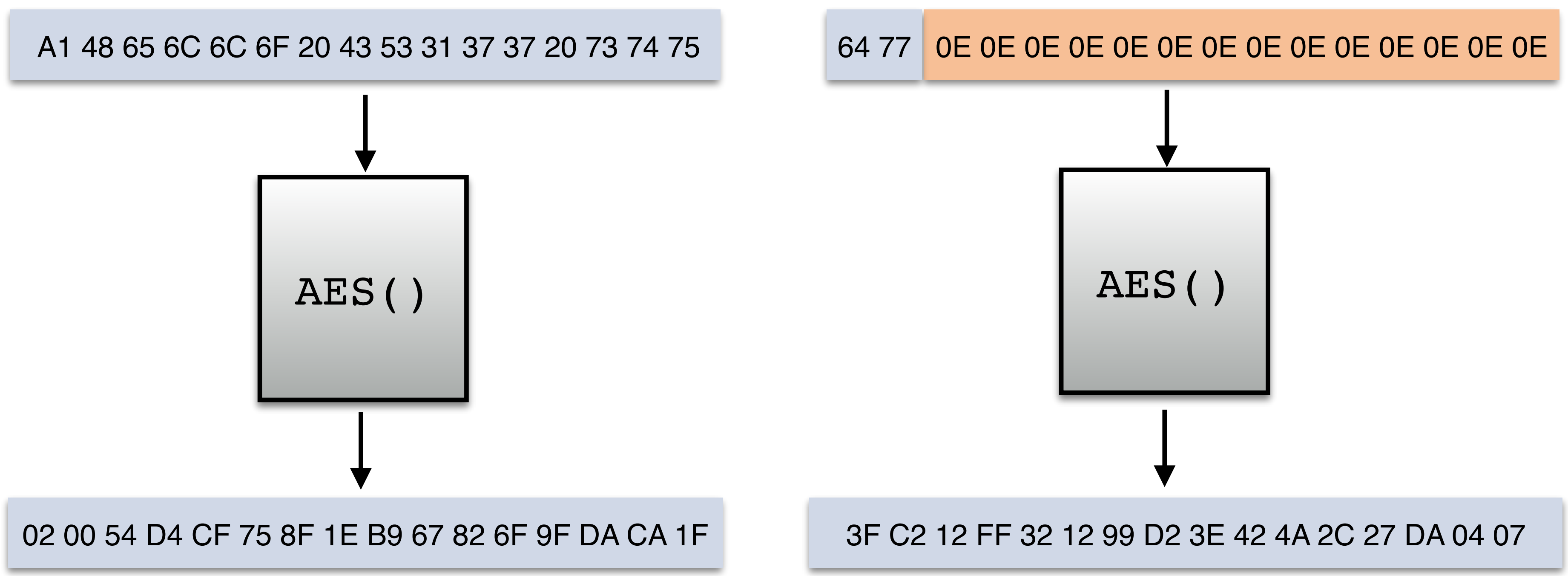
Learning the Padded Blocks



- 256 Queries
- Easy if username is block-aligned.
- Harder but still possible if not aligned, or if alignment is random
    - Need to pre-pend username with bytes to shift it around

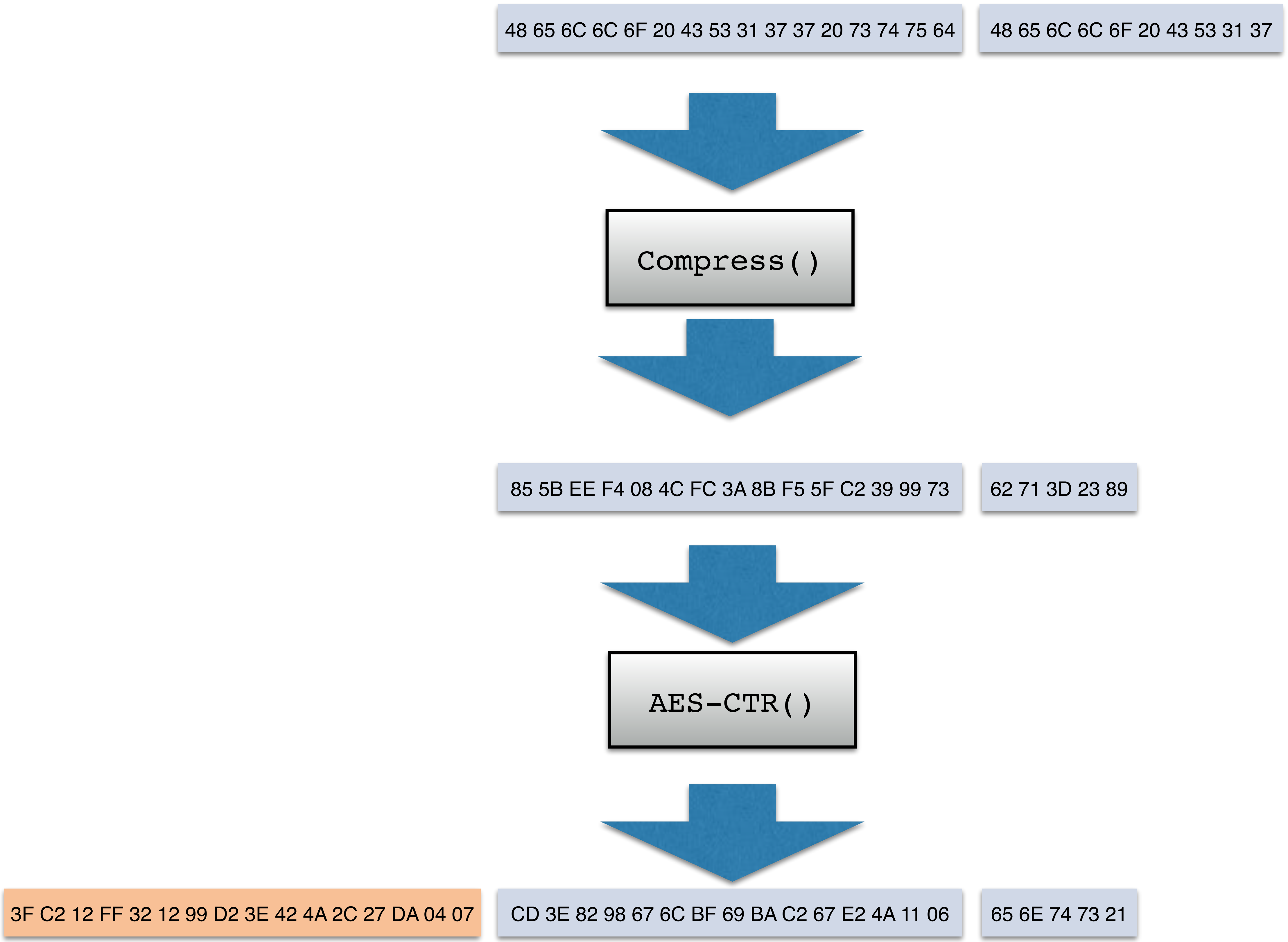
ECB and Partial Plaintext Recovery: Second Byte

(username one char longer)



- Plan: Learn AES(K, ??770E0E0E0E0E0E0E0E0E0E0E0E0E0E) for all 256 values of ??.
- Easily read off value of next byte
- Continue for each byte, then move onto next block

Compress-then-CTR-Encrypt



token=uchic4gos3cr34;userdata=…

Secret info

Username, etc
(Adversary controlled)

Real-world attacks against:
- TLS (2012)
- HTTPS (2013)

`Compress()`

85 5B EE F4 08 4C FC 3A 8B F5 5F C2 39 99 73 | 62 71 3D 23 89

Adversary can see ciphertext length.

More overlap with secret means better compression, i.e shorter ciphertext.

`AES-CTR()`

Project 2: Seeing several ciphertexts enables full plaintext recovery.

3F C2 12 FF 32 12 99 D2 3E 42 4A 2C 27 DA 04 07 | CD 3E 82 98 67 6C BF 69 BA C2 67 E2 4A 11 06 | 65 6E 74 73 21

The End