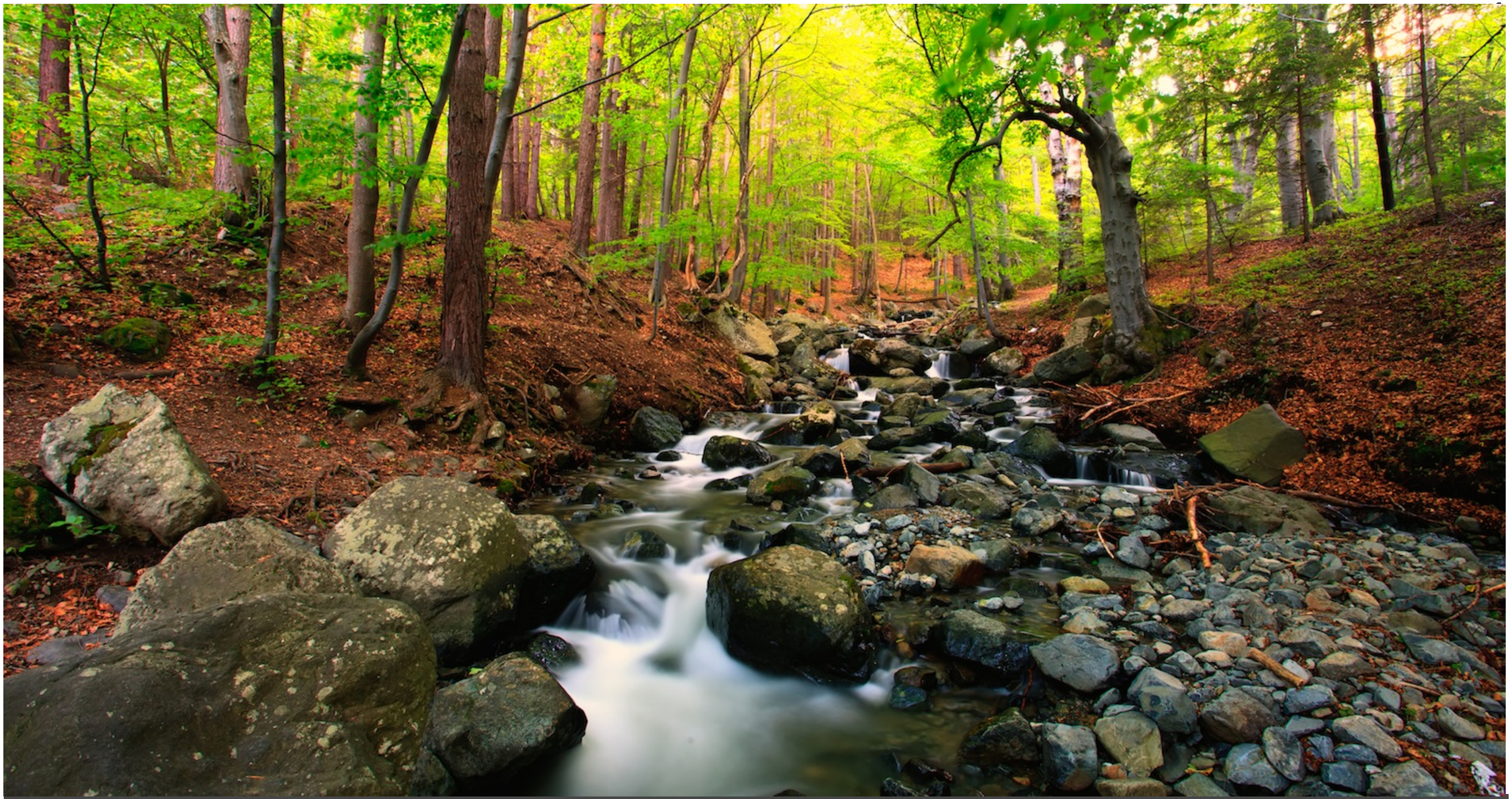
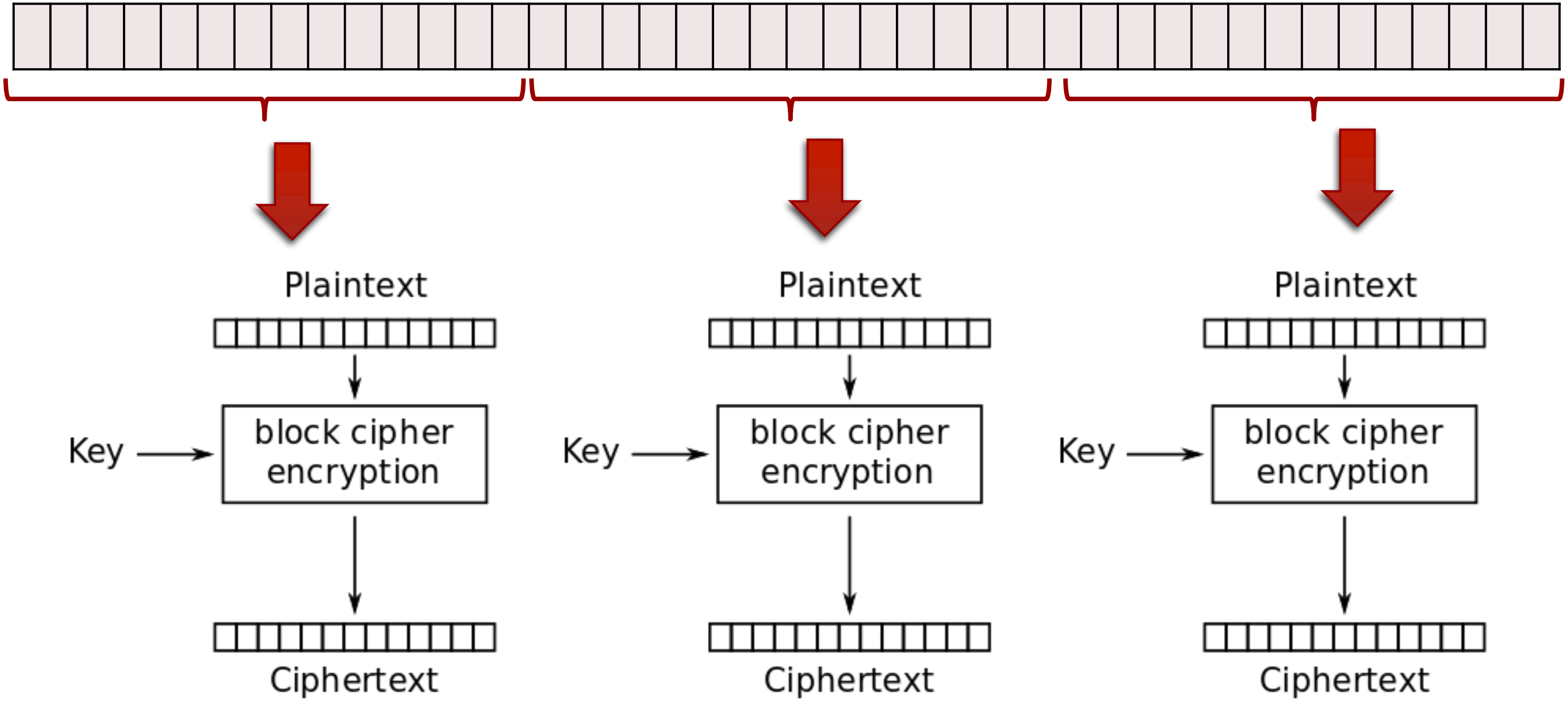


Stream Ciphers

Stream Ciphers

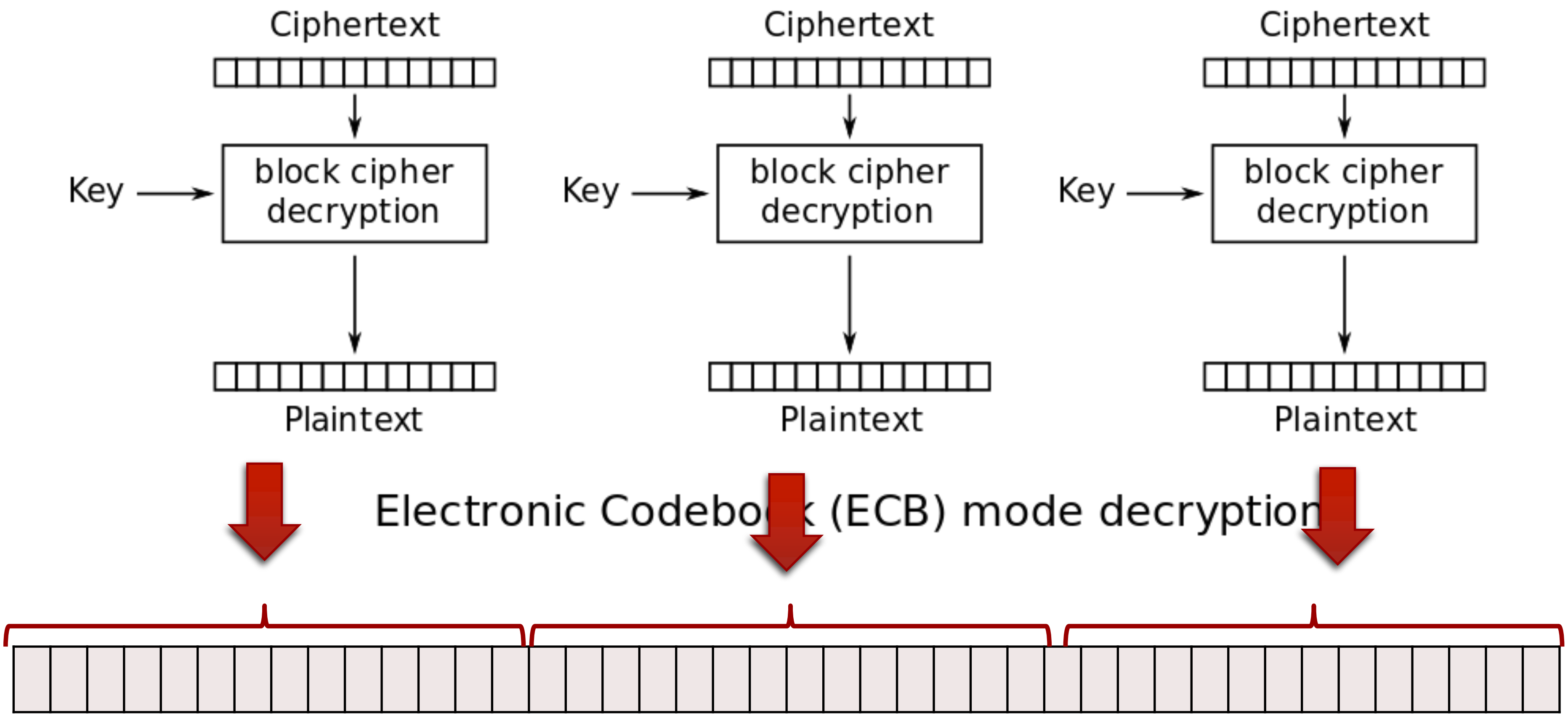
- **Have:** A stream of bits
  - Image, video, webpage, email, …
- **Want:** A cipher that can take an unlimited (or at least very long) stream of plaintext bits and encrypt
- **Idea:** Divide incoming stream into blocks and encrypt each separately via existing block cipher
  - Called **Block Cipher Mode of Operation**
  - First attempt: **Electronic Code Book (ECB)** mode
  - Note: These are **not** AES-specific – Modes of Operation work for any block cipher

Electronic Code Book Mode (ECB)



Electronic Codebook (ECB) mode encryption

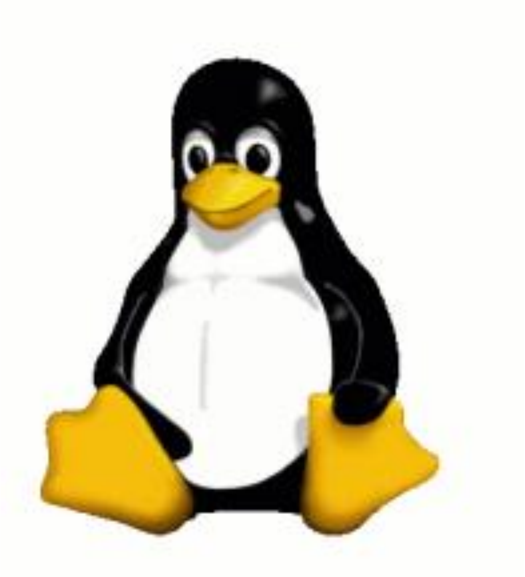
Electronic Code Book (ECB) Mode

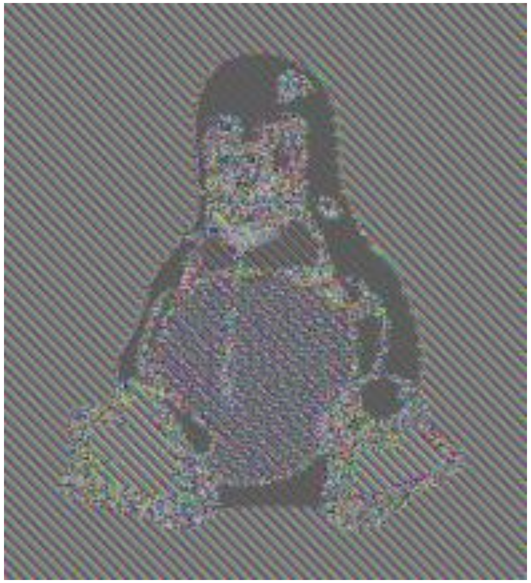


Electronic Code Book Mode (ECB)

- Electronic Code Book Mode (ECB) – **Don't use!**
- **Two big problems**
  - Same input block produces the same cipher block each time
  - Replay attacks

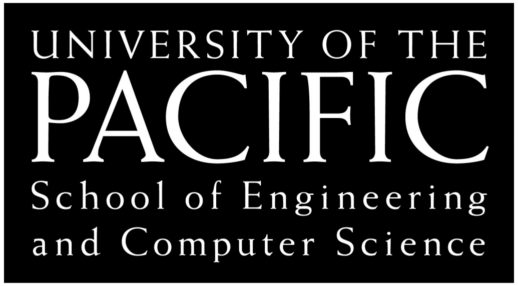
Identical Blocks



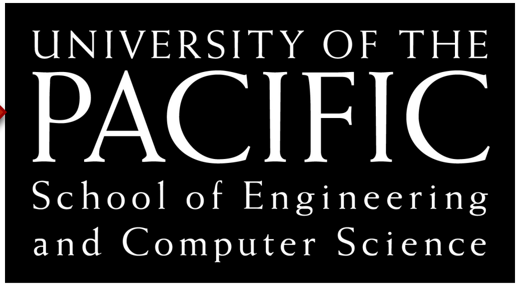


```
#!/usr/bin/python3
# Jeff Shafer, University of the Pacific
# Demo program illustrating information leakage
# of block ciphers (e.g. AES) in ECB mode

# Requires Python3 and PyCrypto
# https://www.pycrypto.org

from Crypto.Cipher import AES
from hashlib import md5

# Example image, 1418x779 pixels, 8 bit color depth
# AES default block size of 128 bits will take this image
# 15 pixels at a time

file = open("pacific.bmp", "rb")
plaintext_original = file.read() + b'000000'  # Pad length to multiple of 16. BMP files don't care.
##print(len(plaintext_original))

# Generate a key for AES encryption/decryption
# AES-128 key length is 16 bytes (128 bits)
key = md5("bogus garbage".encode('ascii')).hexdigest()

# Encrypt with AES in ECB mode
cipher = AES.new(key, AES.MODE_ECB)
ciphertext = cipher.encrypt(plaintext_original)

# "Cheat" for demo purposes - In order to view the ciphertext as a bitmap image,
# we copy the bitmap header bytes (specifying dimensions, color depth, etc...)
# from the unencrypted image and append the ciphertext after that.
# Use the bless hex editor, look at offset 0xA, and that byte will
# tell you where the actual image data starts after the header.
fake_ciphertext = plaintext_original[0:121] + ciphertext[122:]
file2 = open("pacific_encrypted.bmp", "wb")
file2.write(fake_ciphertext)

# Decrypt
plaintext_final = cipher.decrypt(ciphertext)
plaintext_final = plaintext_final[:-6]  # Cut off padding applied earlier

file3 = open("pacific_decrypted.bmp", "wb")
file3.write(plaintext_final)
```

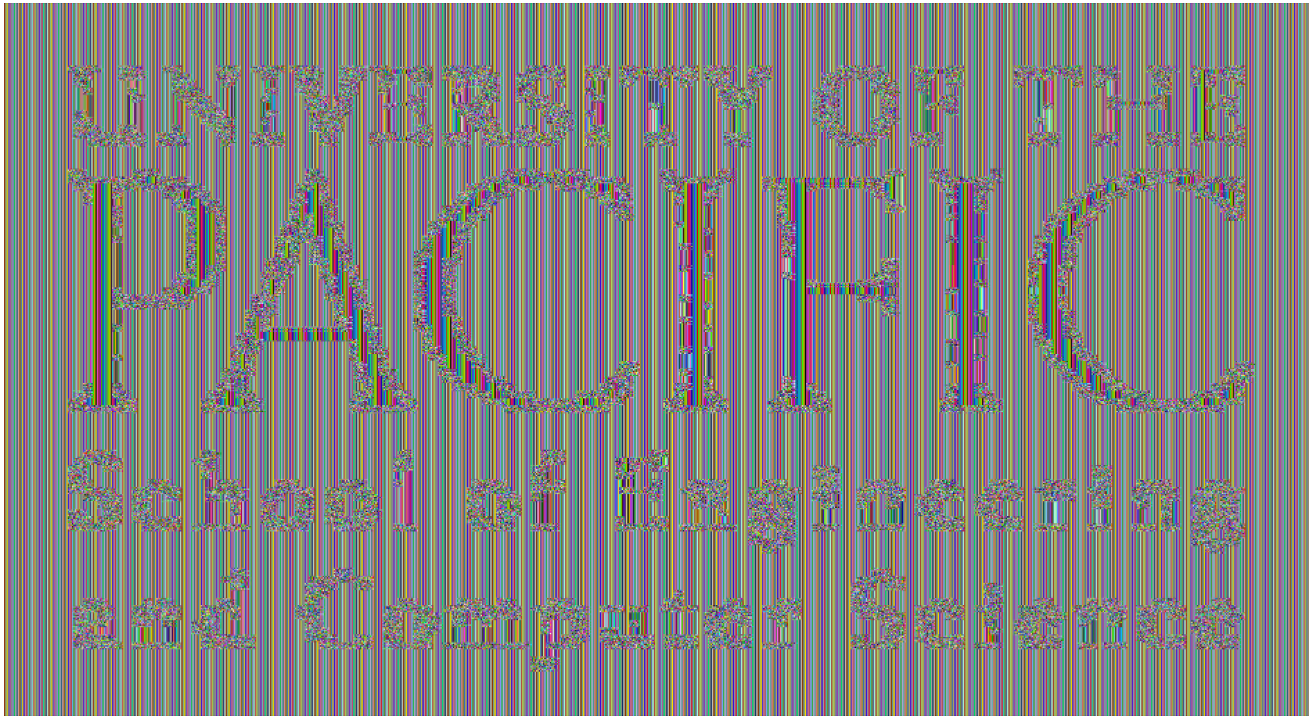
Original



Decrypted

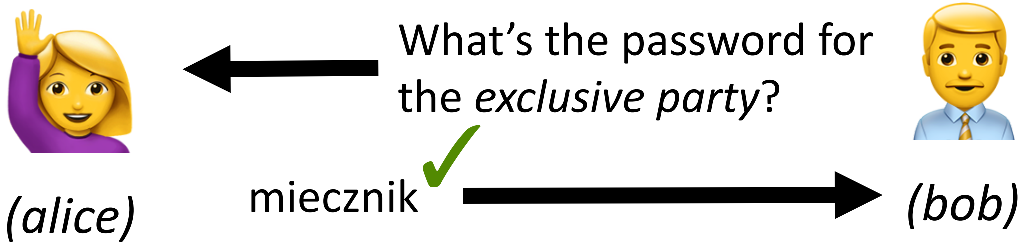


Ciphertext



https://www.zerodayclothing.com/

Replay Attack



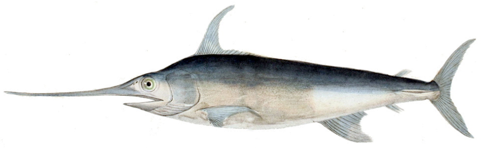
What's the password for the *exclusive party*?

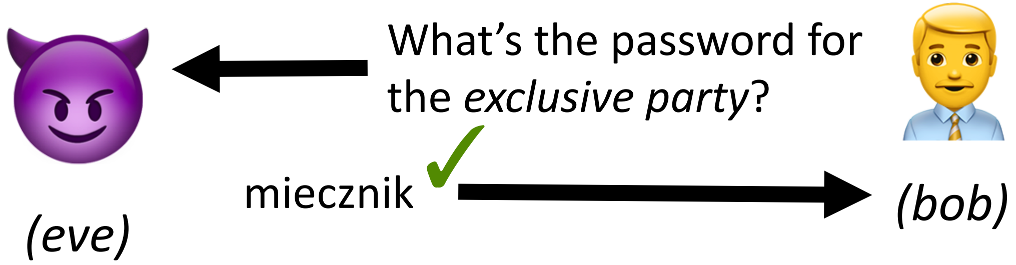
miecznik ✓

(alice)

(bob)

Miecznik = Polish word for swordfish





What's the password for the *exclusive party*?

miecznik ✓

(eve)

(bob)

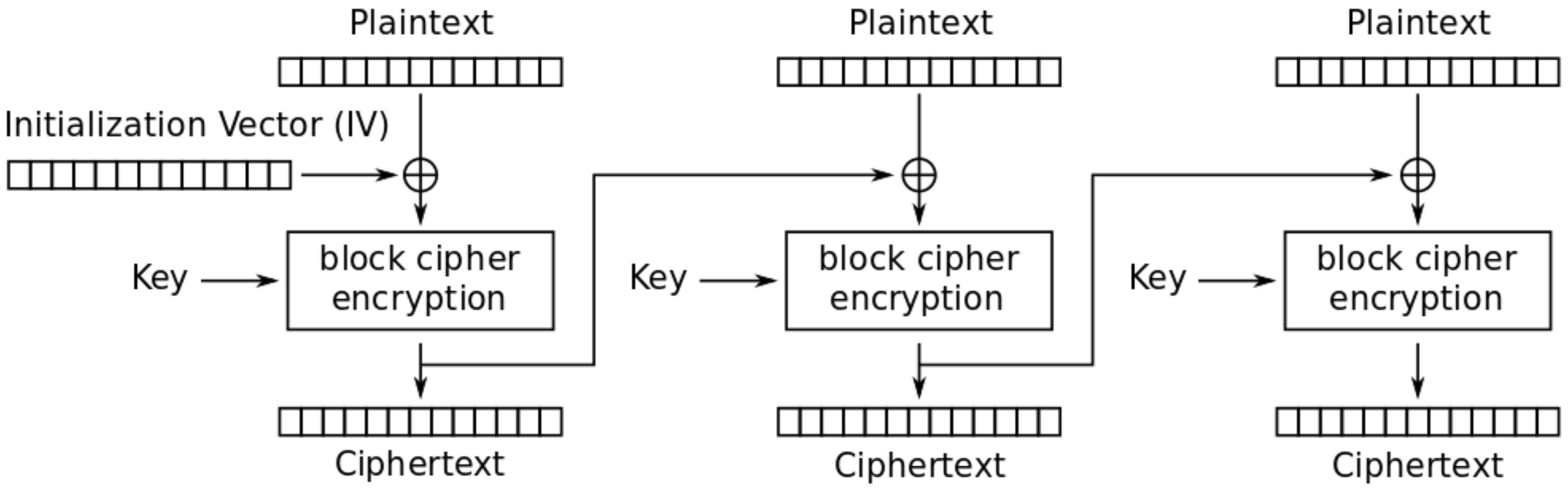
Sneakers (1992)
“My voice is my passport, verify”

Block Cipher Modes of Operation

- Electronic Code Book Mode (ECB) – **Don’t use!**
- Cipher Block Chaining (CBC) – **Good but inefficient**
- Propagating Cipher Block Chaining (PCBC)
- Ciphertext Stealing (CTS)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR) - **Good**
- … and more options that add *authentication* to the *confidentiality* already provided *(will cover later)*
  - CCM, GCM, CWC, EAX, IAPM, OCB….

- **And this is *one* place where people make crypto mistakes**
  - You don’t just pick AES, you pick AES+EBC, AES+CBC, etc…. ☹

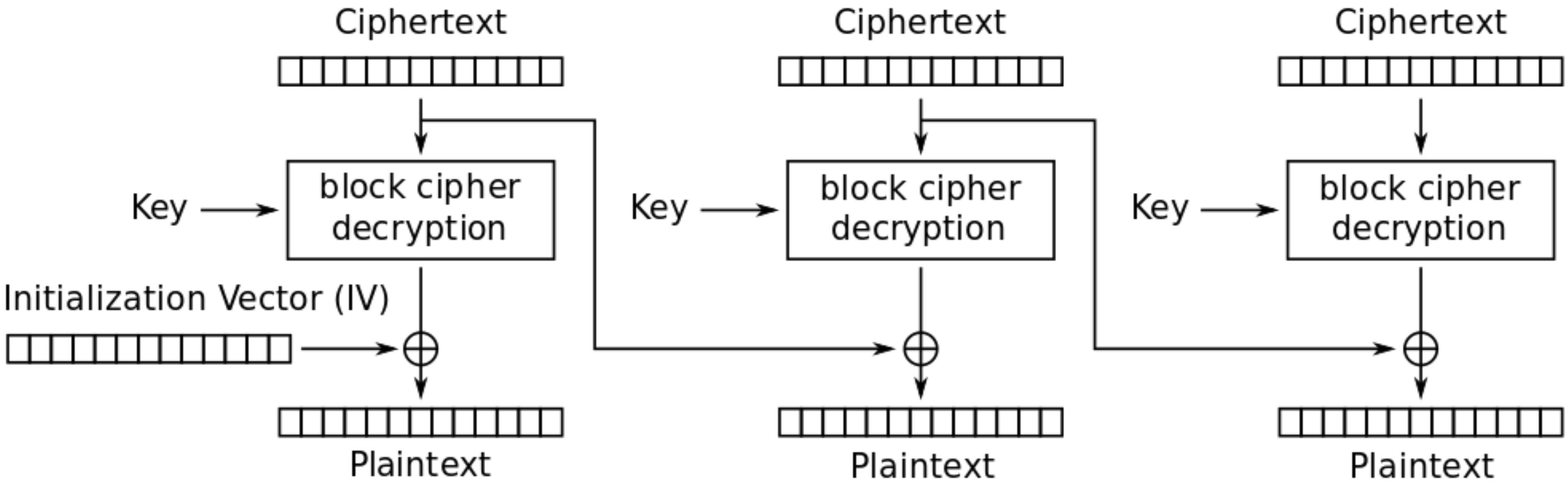
Cipher Block Chaining (CBC) Mode



Cipher Block Chaining (CBC) mode encryption

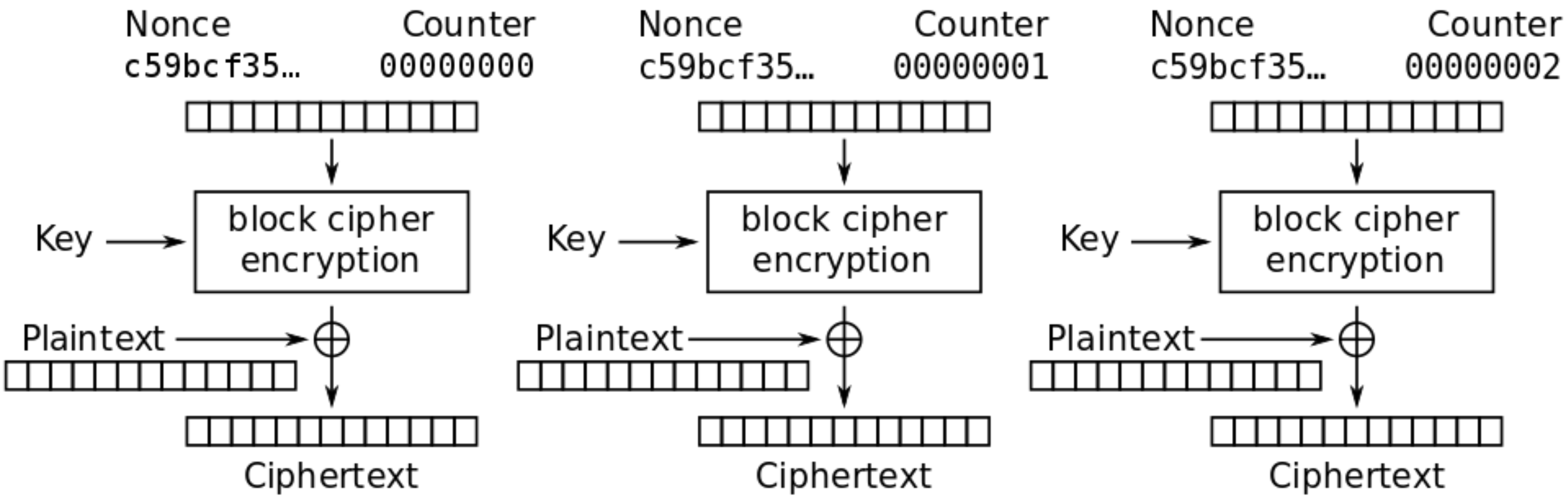
- Plaintext blocks are XORed with previous ciphertext block before being encrypted
- First block is XORed with **Initialization Vector (IV)** – length of 1 block
  - Must be **cryptographically random!** (predictability here was cause BEAST of SSL/TLS attack)
  - Is **<u>not</u>** secret – typically prepended to ciphertext in plain text

Cipher Block Chaining (CBC) Mode



Cipher Block Chaining (CBC) mode decryption

Counter (CTR) Mode

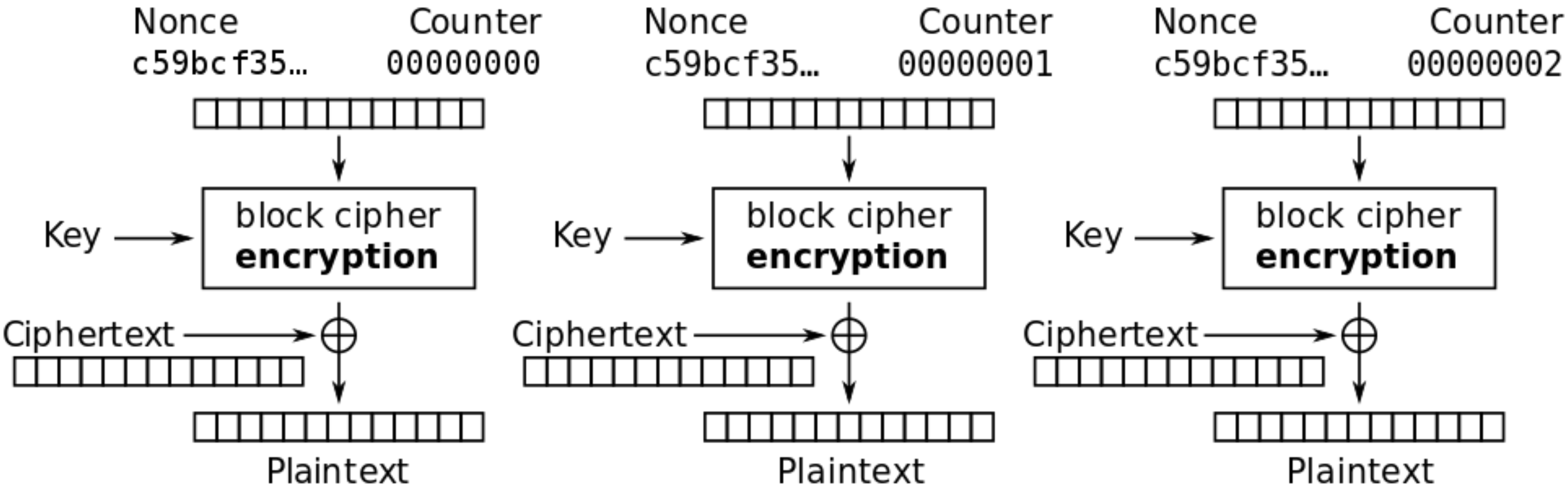


Counter (CTR) mode encryption

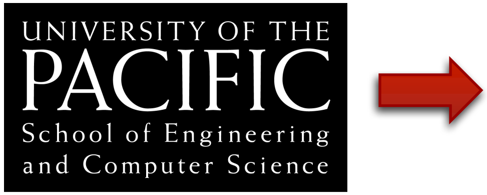
- Developed by **Whitfield Diffie** and **Martin Hellman**, 1979
- Encrypt a {nonce, counter} value, then XOR with plaintext to yield ciphertext
- The encrypted {nonce, counter} are like a **pseudo-OTP!**

- Operation
  - “Nonce” = IV (cryptographically random)
  - “Counter” is any sequence guaranteed not to repeat for long time (like a counter!)
  - Combine Nonce with Counter via concatenation (upper and lower 64 bits)

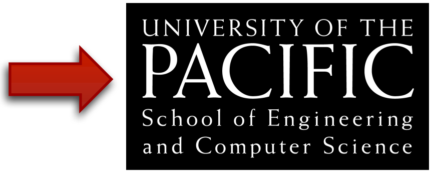
Counter (CTR) Mode



Counter (CTR) mode decryption

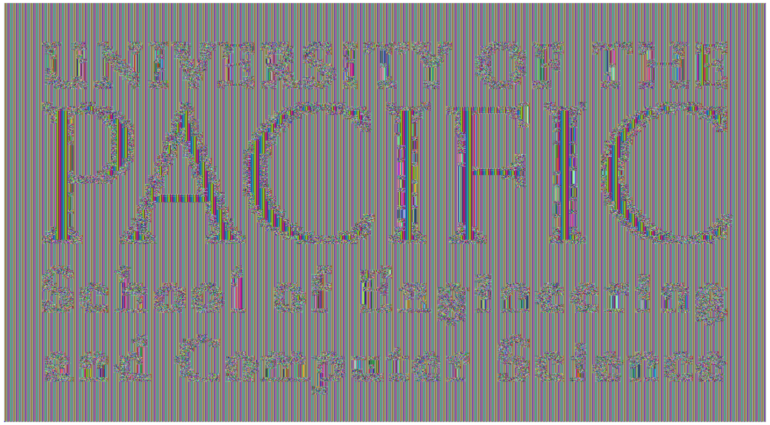


Original

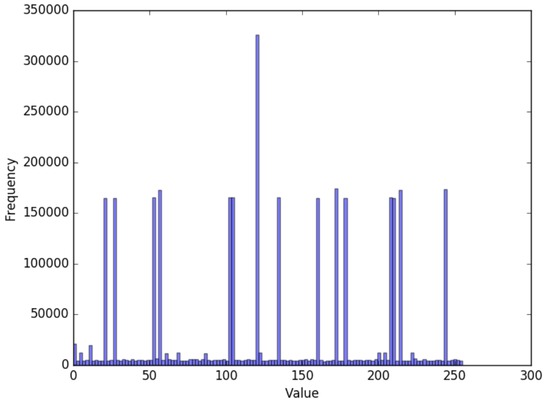


Decrypted

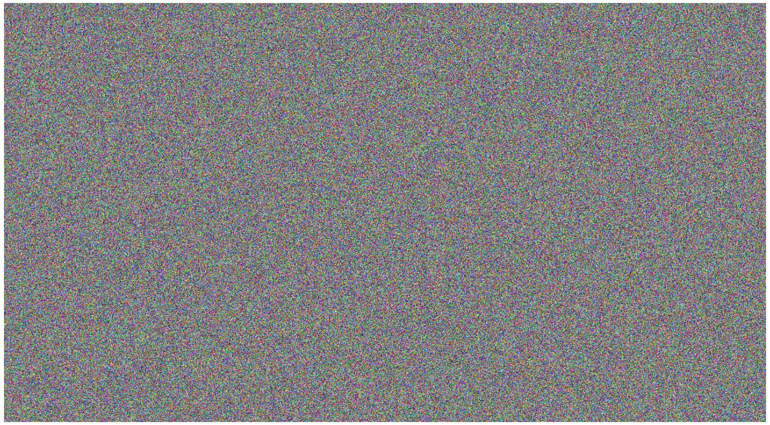
*Ciphertext – AES **ECB** Mode*



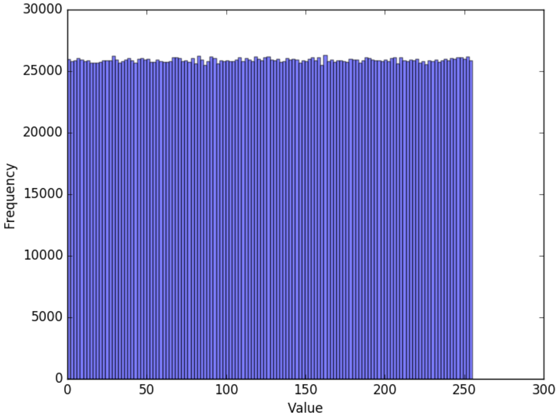
Histogram



*Ciphertext – AES **CTR** Mode*



Histogram

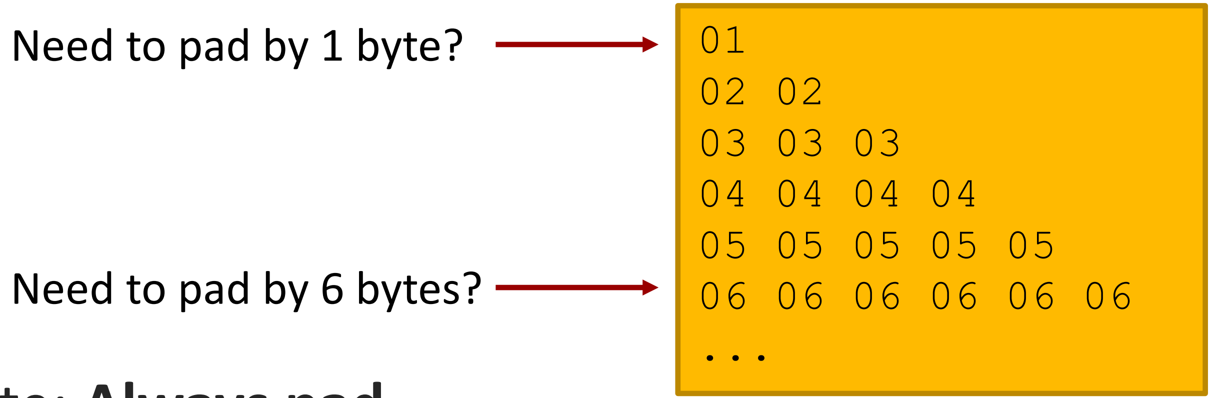


Block Cipher w/Padding

- **Q:** What if the plaintext size isn't a multiple of the block size?
- **A:** Need *padding* at end of plaintext data
- **Q:** How do I distinguish my padding from the original plaintext? (for *arbitrary* plaintext)
- **A:** PKCS#5 / PKCS#7 padding standard

PKCS#7 Padding

- Padding is in whole bytes. Value of each added byte is number of bytes that are added



Need to pad by 1 byte?

Need to pad by 6 bytes?

```
01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
06 06 06 06 06 06
...
```

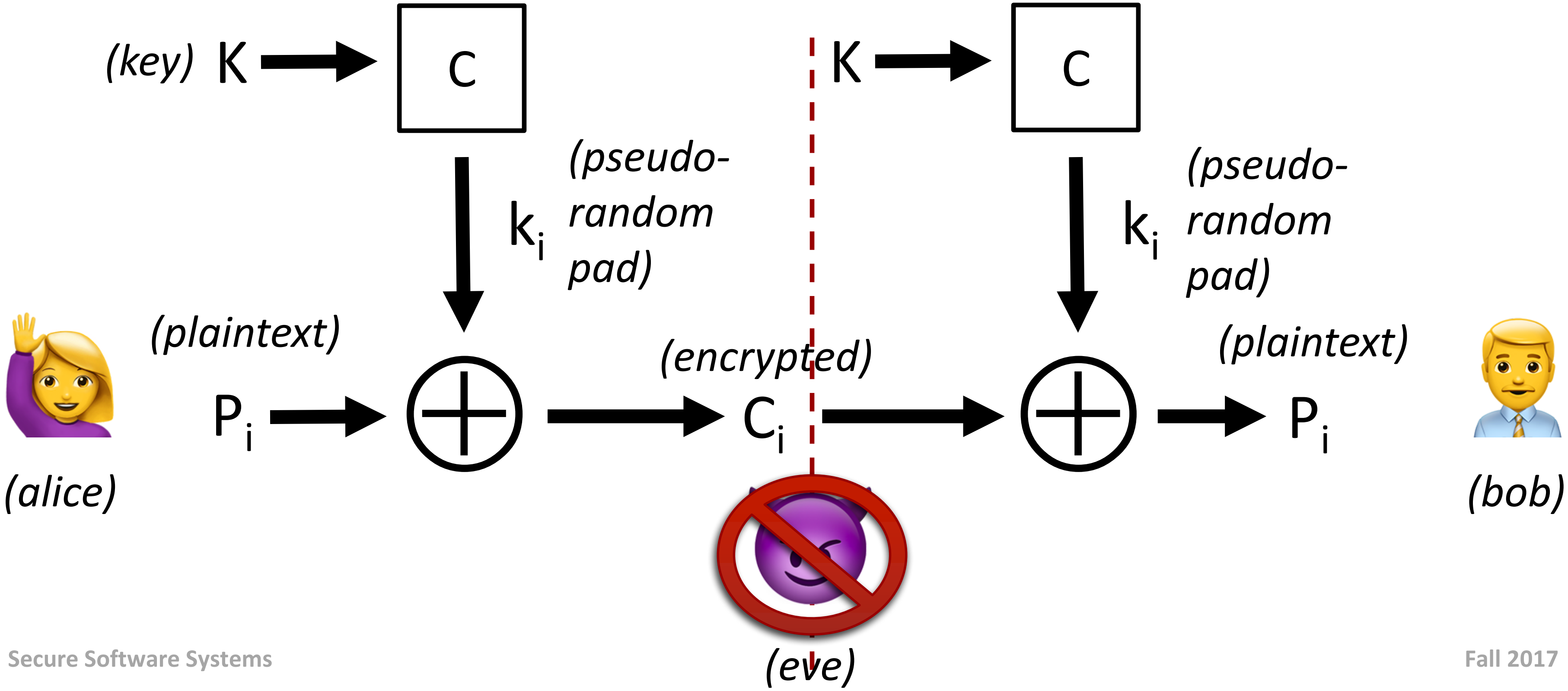
- Note: **Always pad**
  - Even if plaintext is multiple of block size, in which case an entire block is added
  - AES block size: 128 bits (16 bytes), e.g `10` (hex)

(Native) Stream Ciphers

- *What about designing a cipher that doesn’t work with blocks at all?*
  - **Native Stream Cipher**
- Categories
  - Synchronous stream ciphers
  - Self-Synchronizing stream ciphers - *rare*

Synchronous Stream Ciphers

- Like a one-time pad, but with a pseudo-random pad generated by the cipher



Native Stream Cipher Examples

- **Rivest Cipher 4 (RC4)** – **Don’t use!**
  - Designed by Ron Rivest (R in RSA)
  - Simple and fast in software and hardware ☺
  - Used in popular protocols like SSL, TLS, WEP, WPA ☺
  - Insecure ☹
    - Break WPA-TKIP w/RC4 in under an hour
    - Break TLS-protected HTTP cookie in 75 hours
    - Prohibited in TLS in 2016+
      (dropped by Chrome, Firefox, IE/Edge)
    - Shouldn’t be using WEP any more

Research paper

Mathy Vanhoef and Frank Piessens. “All your biases belong to us: breaking RC4 in WPA-TKIP and TLS.” In *Proceedings of the 24th USENIX Conference on Security Symposium* (SEC'15), Jaeyeon Jung (Ed.). USENIX Association, Berkeley, CA, USA, **2015**

Native Stream Cipher Examples

- **Salsa20** and **ChaCha20**
  - Developed by DJB (NaCL author)
  - Secure (that we know…)
  - Useful features?
    - Jump to arbitrary location in bitstream and begin decryption (no need to decrypt from beginning)
    - Sections of ciphertext can be decrypted in parallel
    - Resistant to side channel attacks (all operations are constant time)
    - Competitive performance to AES, even without being hardware accelerated