CSCI 3330 Cybersecurity

Word of the day

WHAT'S A SPLINE? This phrase expands to: "You have just used a term that I've heard for a year and a half, and I feel I should know, but don't. My curiosity has finally overcome my guilt." The PARC lexicon adds "Moral: don't hesitate to ask questions, even if they seem obvious."

Block ciphers problem

- So I have 64-bit block
- What if my message doesn’t exactly fit?

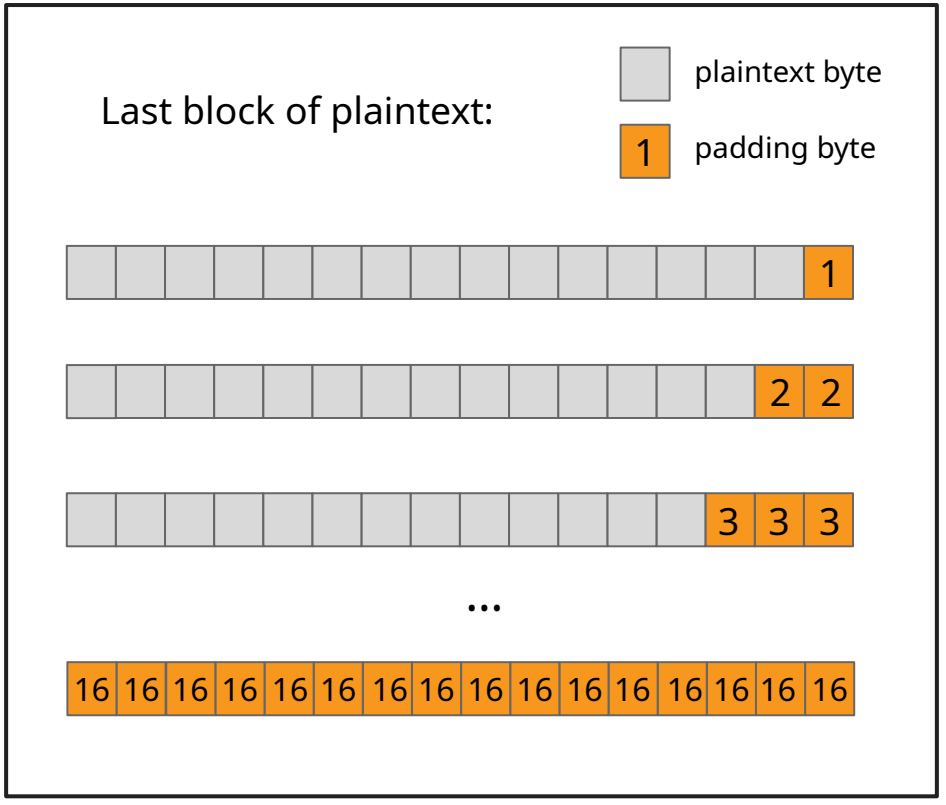
Padding

- Needs to be unambiguous
- What can we do?

PKCS#7 padding

- Pad the remaining *n* bytes with the byte *n*
- What if there are no remaining bytes?

PKCS#7 padding



Symmetric encryption

- Same key used to encrypt and decrypt

Common symmetric block ciphers

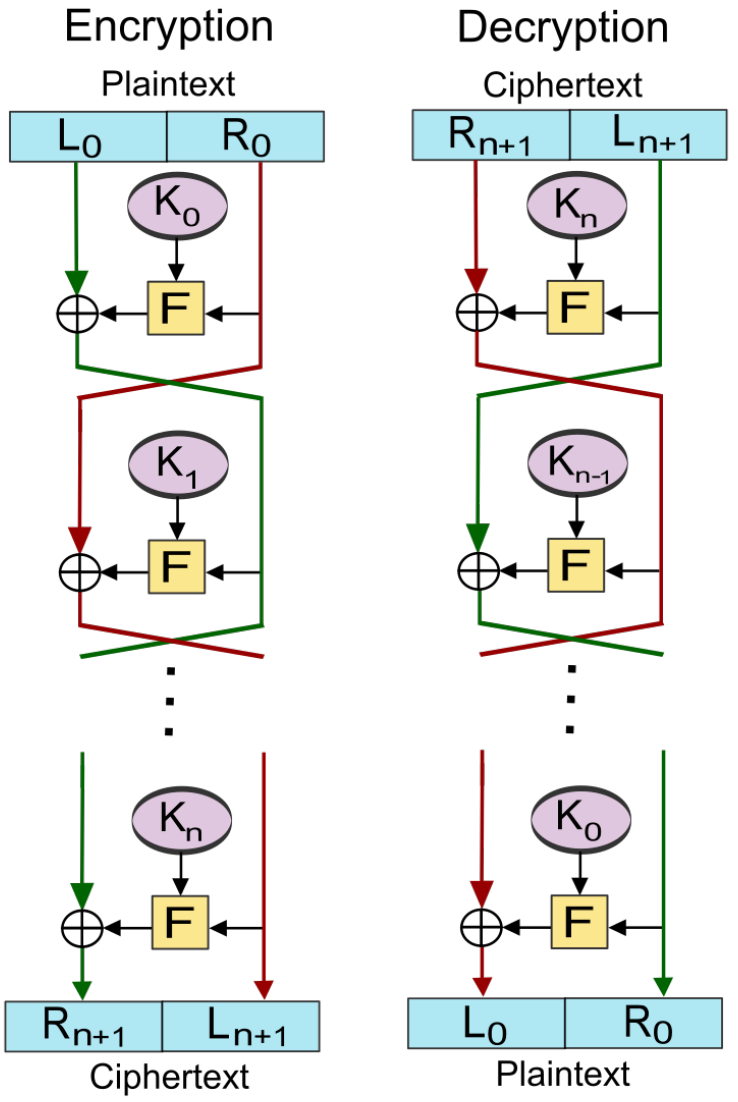
- DES
  - Data Encryption Standard
- AES
  - Advanced Encryption Standard

DES

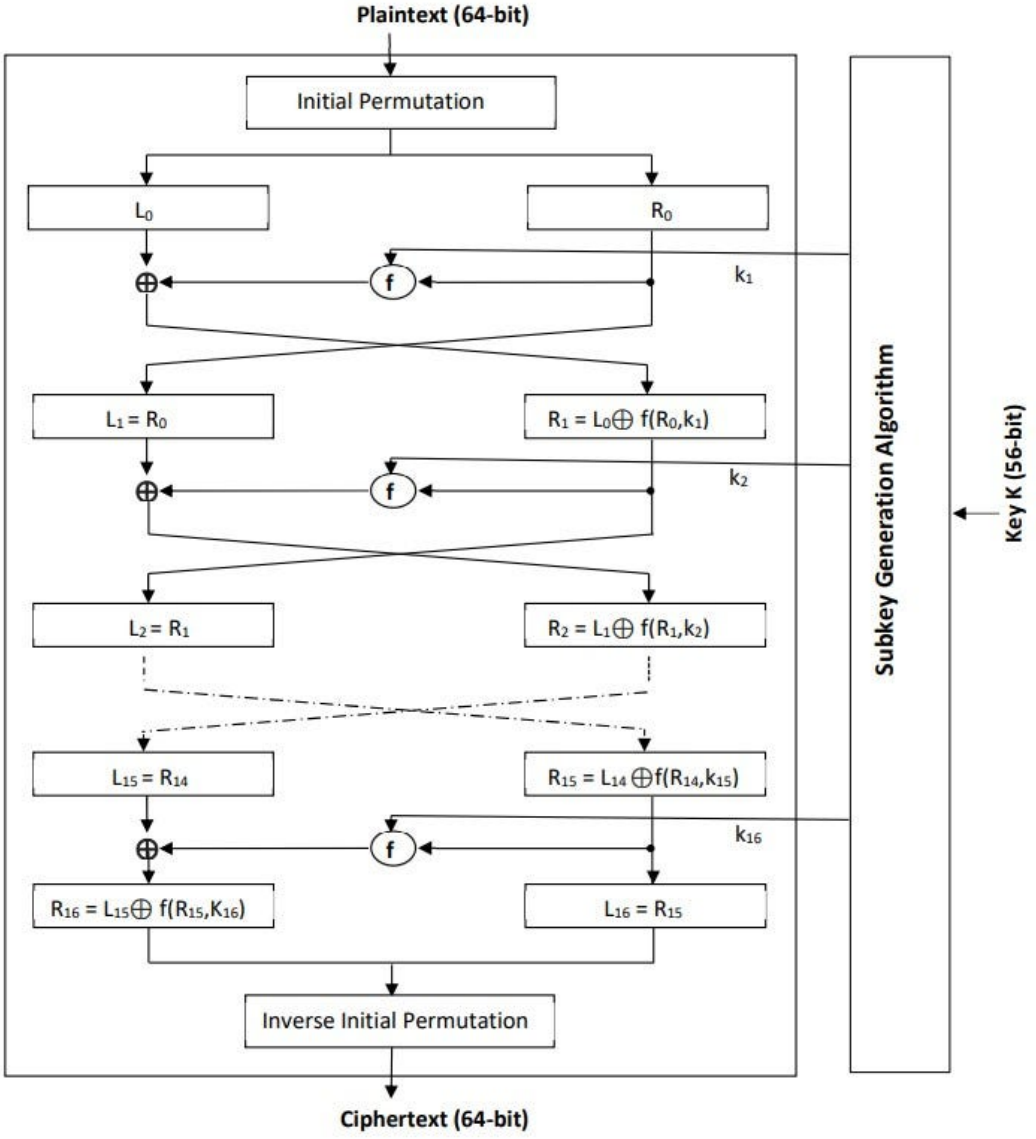
- Developed in early 70s by IBM
- 64-bit block cipher with a 56-bit key
- Feistel structure

Feistel cipher

- Rounds
- Key schedule
- Decryption is encryption with key schedule in reverse
- F: Feistel function

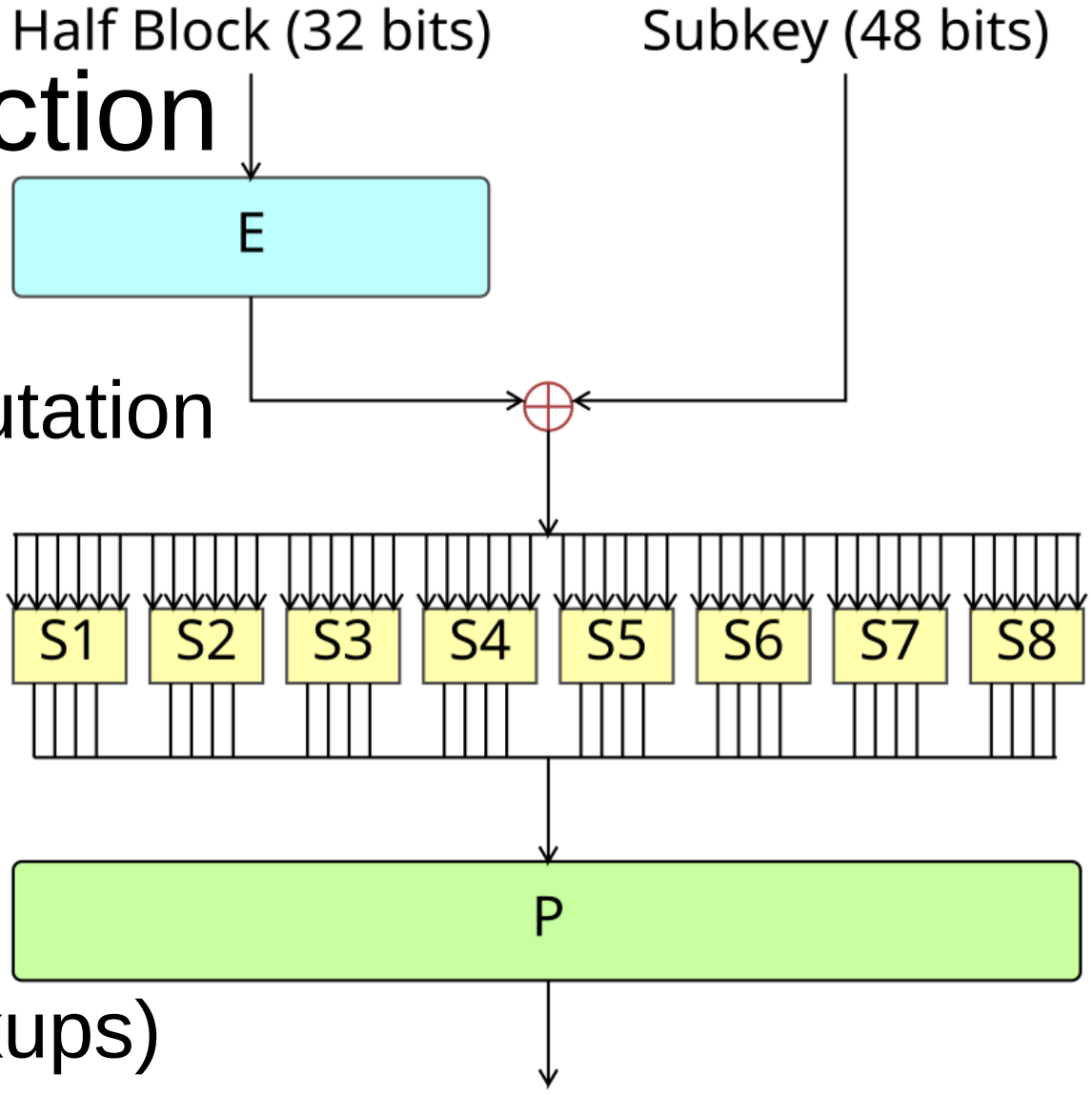


16 rounds


Plaintext (64-bit)
Initial Permutation
L_0
R_0
k_1
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0,k_1)$
k_2
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1,k_2)$
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14},k_{15})$
k_{16}
$R_{16} = L_{15} \oplus f(R_{15},K_{16})$
$L_{16} = R_{15}$
Inverse Initial Permutation
Ciphertext (64-bit)
Subkey Generation Algorithm
Key K (56-bit)

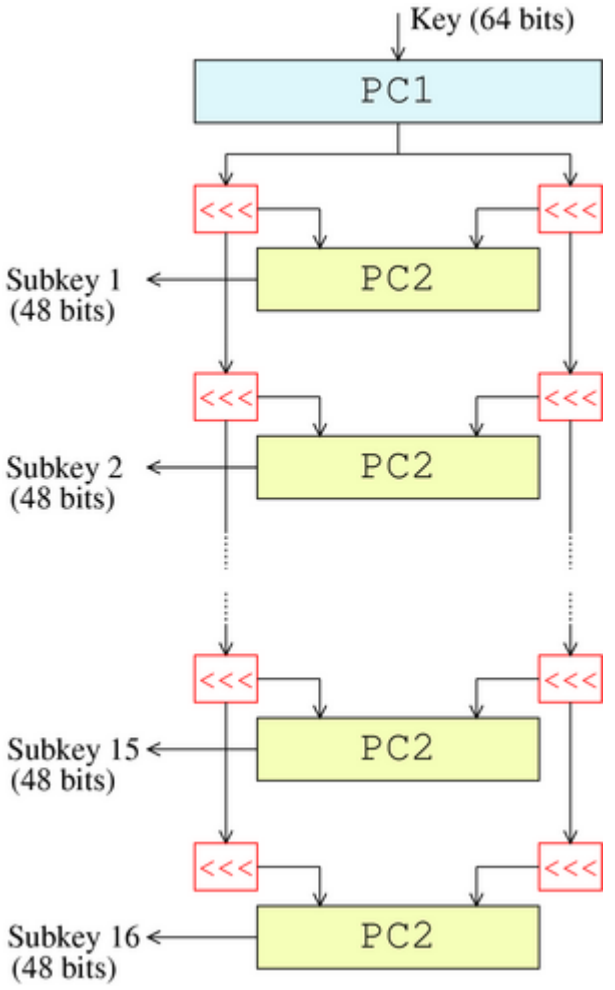
DES Feistel function

- E: expansion permutation (randomly choose bits with some chosen twice in a random order)
- P: permutation
- S: S-box (table lookups)



DES key schedule

- PC: permuted choice
  - (choosing subset of bits in a random order)



DES

- Is DES secure?
- There are generally no proofs of a cryptosytem being secure outside of trivial examples like a one-time pad

DES

- NSA nudged IBM to change S-boxes
- Did they introduce a backdoor?

DES

- NSA nudged IBM to change S-boxes
- Did they introduce a backdoor?
- Added resistance to differential cryptanalysis
- A kind of attack not discovered until 1990!

DES

- No longer considered secure
- Multiple attacks against it
- Can be brute forced using specialized hardware

3DES (Triple DES)

- Key = 3 DES keys k_1 k_2 k_3
- 56 * 3 == 168 bit key

  DES_encrypt(DES_decrypt(DES_encrypt(*plaintext*, k_1), k_2) k_3)

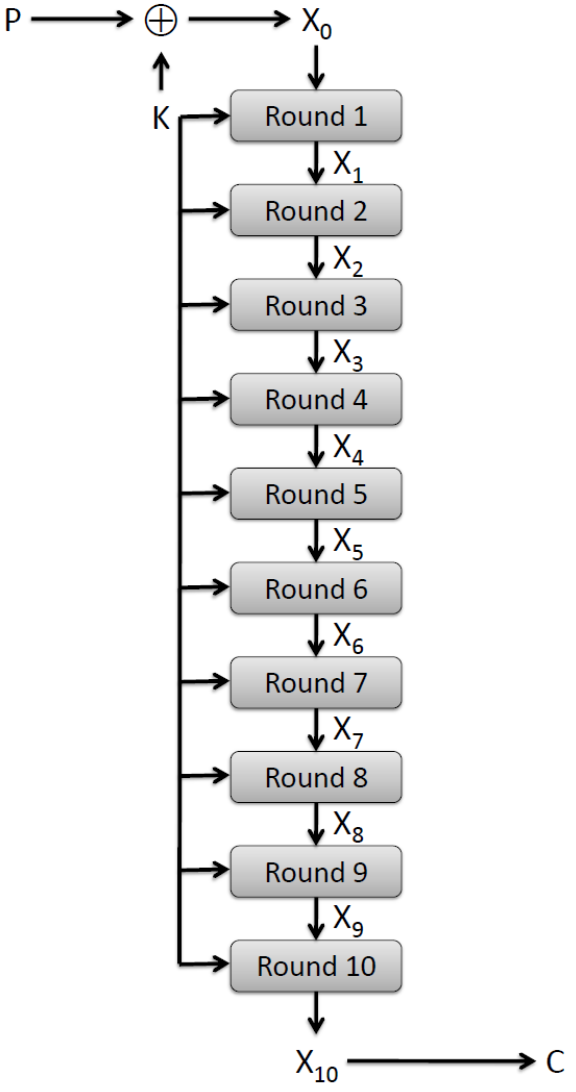
- Can be implemented using DES hardware with three DES operations
- 3DES hardware can implement DES
  - When k_1 == k_2 == k_3, implements DES with key k_1

AES (Advanced Encryption Standard)

- DES weak
- U.S. government wanted to standardize the next encryption standard
- Contest
- Based on Rijndael (1998)

AES tl;dr

- 128-bit block cipher
- Keys are either
  128, 192, or 256 bits long
- # of rounds depends on key size



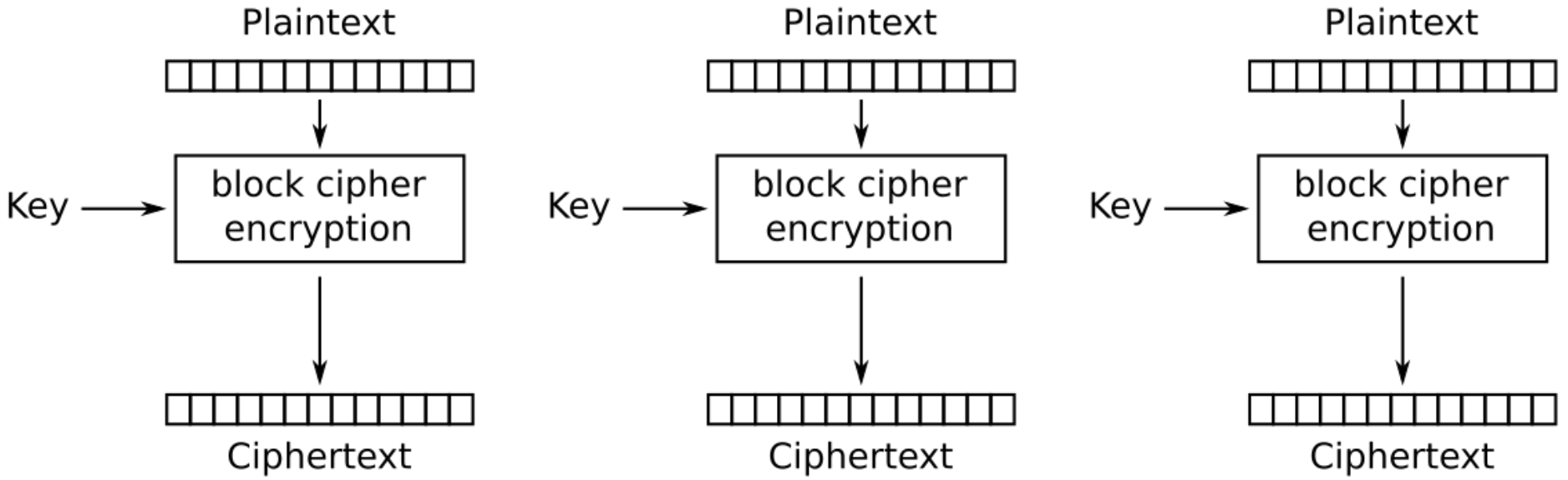
	DES	**AES**
Key length	56 bits	128, 192, 256 bits
Block size	64 bits	128 bits
Rounds	16	10, 12, 14
Construction	Substitution, permutation	Substitution, permutation, mixing, addition
Developed	1977	1998
Status	Broken!	OK (for now)

“OK” means that you cannot do much better than guessing all possible keys

Block cipher modes

- Block ciphers encrypt/decrypt blocks
- Problem: my data is more than a block!
- Block cipher *modes*

ECB mode



Electronic Codebook (ECB) mode encryption

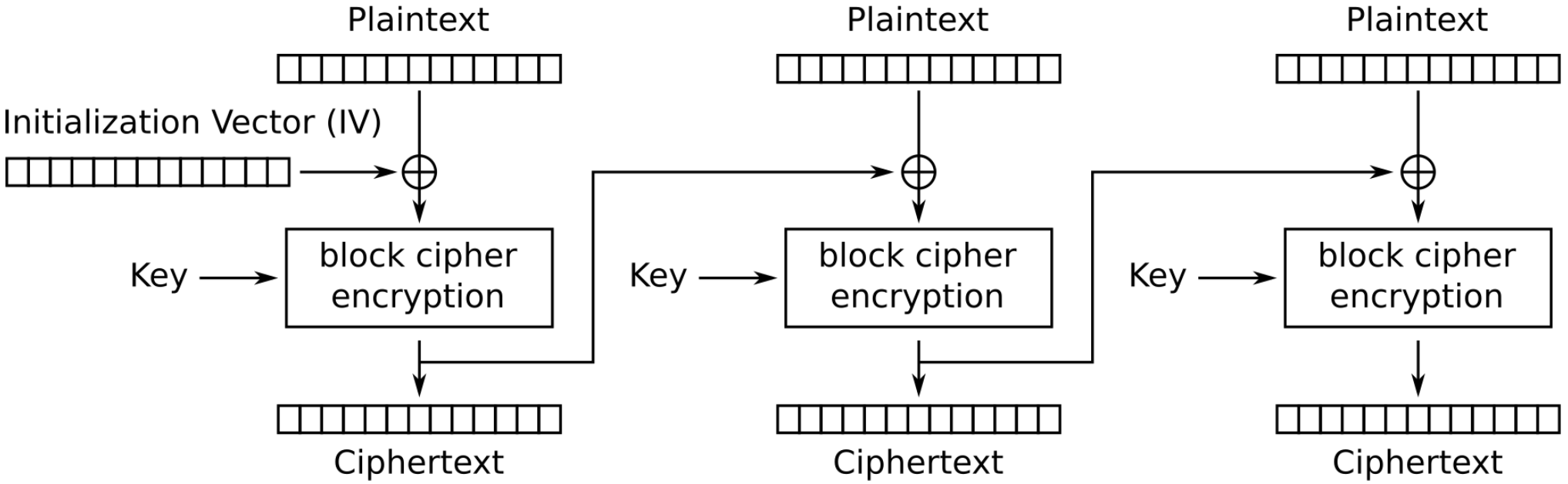
ECB mode

Original image

Using ECB allows patterns to be easily discerned

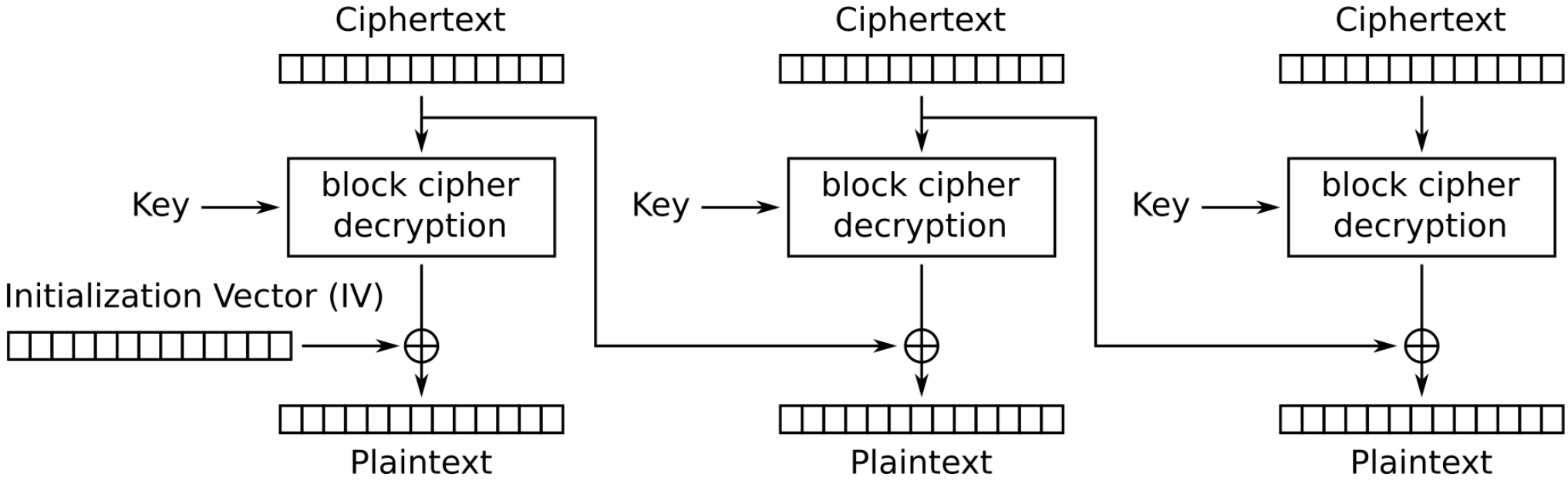
Modes other than ECB result in pseudo-randomness

CBC mode



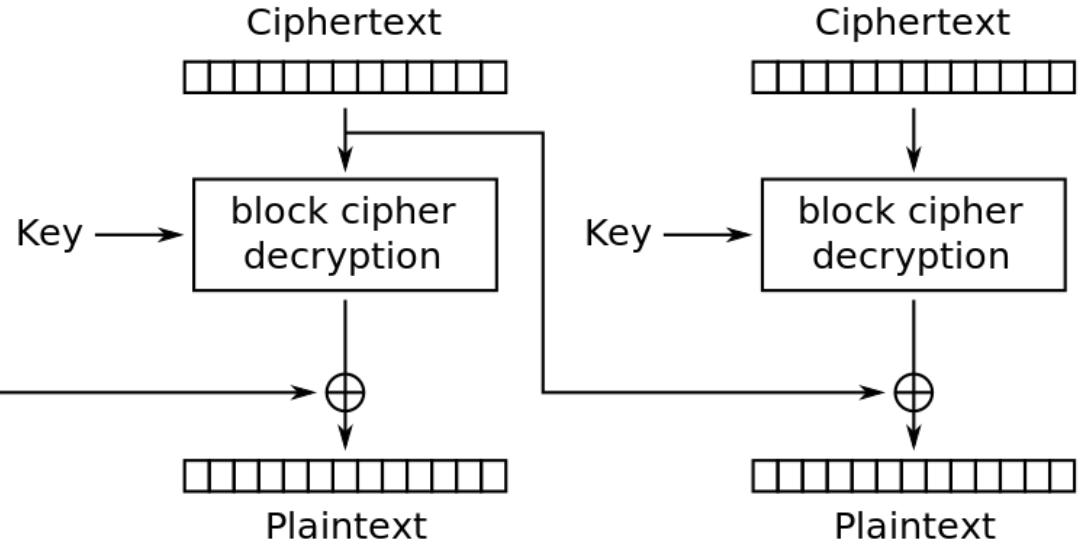
Cipher Block Chaining (CBC) mode encryption

CBC padding oracle



Cipher Block Chaining (CBC) mode decryption

CBC padding oracle



$$\text{ciphertext}_{n-1} \oplus \text{Decrypt}(\text{ciphertext}_n) = \text{plaintext}_n$$

Chaining (CBC) mode decryption

CBC padding oracle

Find byte *b* such that

$$b \oplus \text{Decrypt}(\text{ciphertext}_n)[15] == 0x01 \Rightarrow \text{Decrypt}(\text{ciphertext}_n)[15] == b \oplus 0x01$$

To recover $\text{plaintext}_n[15]$

$$\text{plaintext}_n[15] = \text{ciphertext}_{n-1}[15] \oplus b \oplus 0x01$$

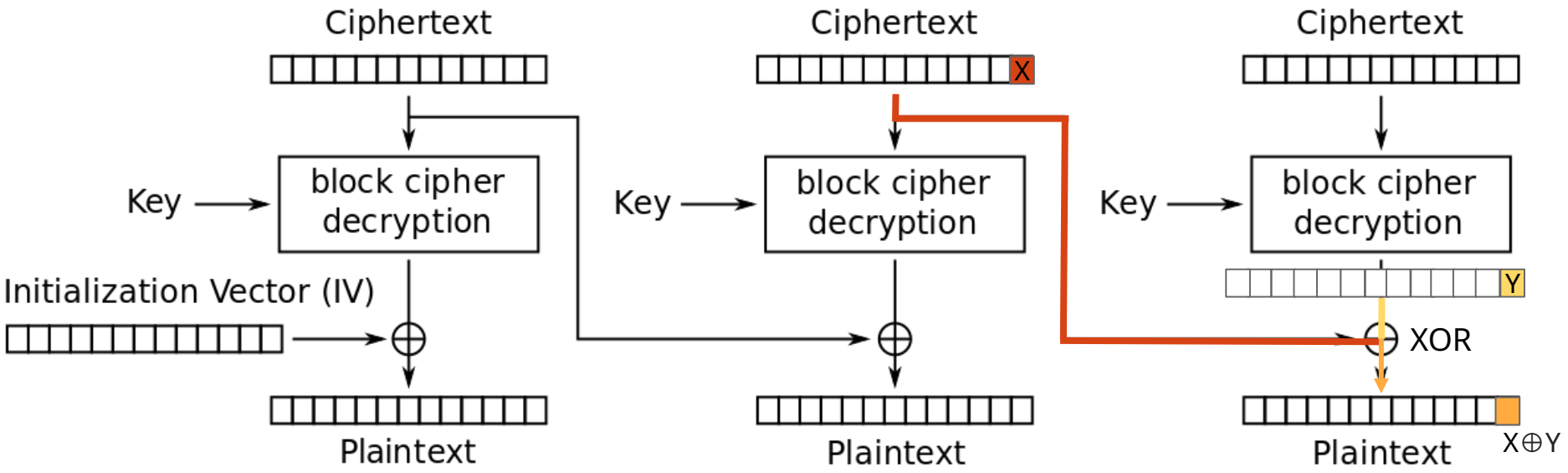
Set

$$\text{ciphertext}_{n-1}[15] = b \oplus 0x01 \oplus 0x02$$

Find *b* such that

$$b \oplus \text{Decrypt}(\text{ciphertext}_n)[14] == 0x02$$

…



Cipher Block Chaining (CBC) mode decryption

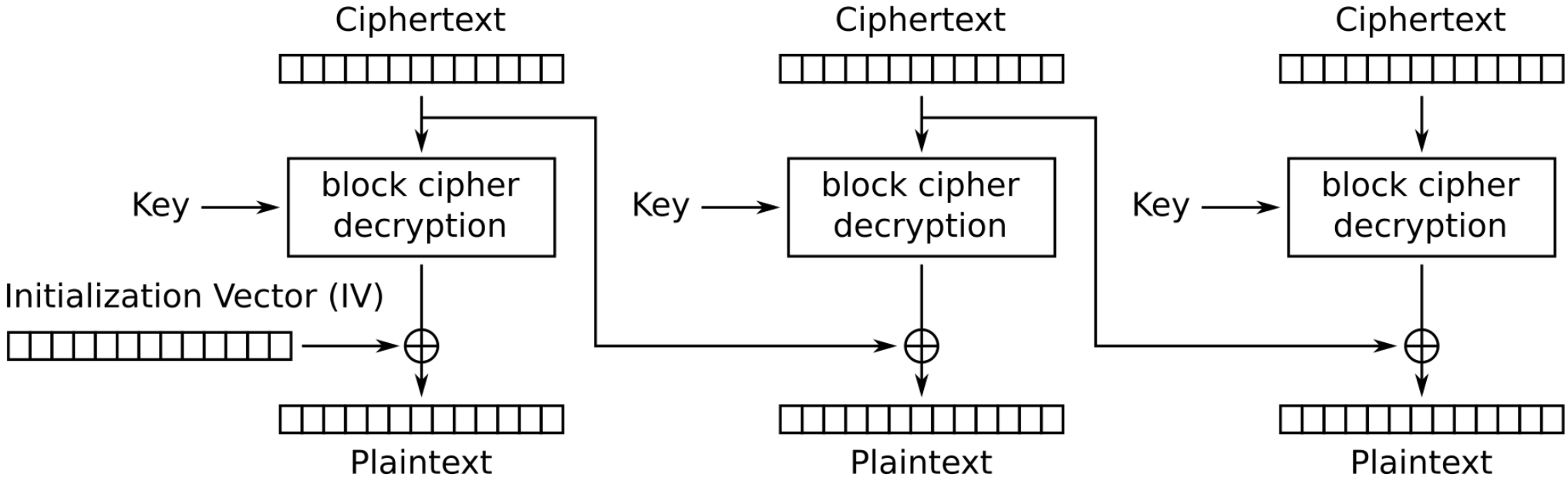
Padding Oracle Attack: changing the last byte of the penultimate ciphertext block changes the last byte of the final plaintext block.

There are only

256

possible bytes

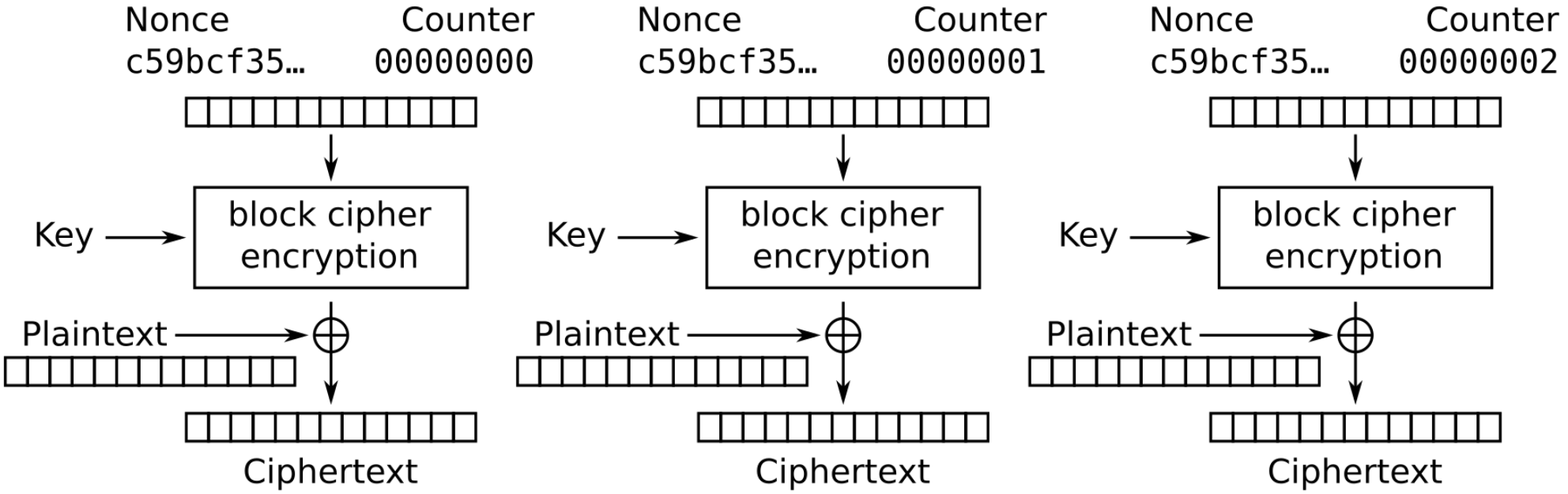
CBC mode: error propagation



Cipher Block Chaining (CBC) mode decryption

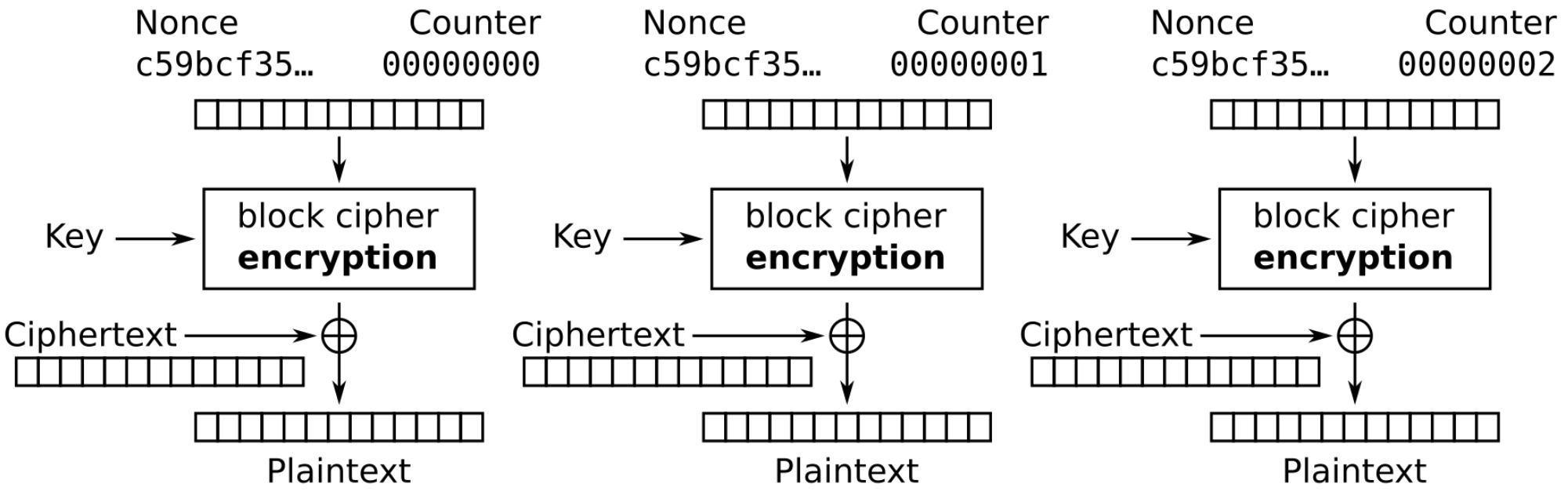
An error in the ciphertext corrupts next block

CTR mode



Counter (CTR) mode encryption

CTR mode



Counter (CTR) mode decryption

Parallelizable, no padding required!

GCM mode

- Combines counter mode with authentication (provides *integrity*)
- The mode considered “the best” (for now)

Cryptographic hashes

- *One-way*
- What were the three properties?

Pre-image resistance

- Given *h*, it should be infeasible to find a message *m* such that *h* = hash(*m*).
- “The hash should be one-way.”

Second pre-image resistance

- Given m_1, it should be infeasible to find an m_2 such that hash(m_1) = hash(m_2).
- “Given a message, you shouldn’t be able to find another message with the same hash.”

Collision resistance

- It should be infeasible to find two messages m_1 and m_2 such that hash(m_1) = hash(m_2).
- “You shouldn’t be able to find any pair of messages with the same hash.”

Breaking cryptographic hashes

- Wang Xiaoyun
- MD5 fails collision resistance
- SHA-1 if you have ~$100k

HMAC

- Normally you hash a message
- An HMAC takes a message and a *key*
- Symmetric algorithm for authentication

HMAC

$$\mathrm{HMAC}(K, m) = \mathrm{H}\left(\left(K' \oplus opad \right) \parallel \mathrm{H}\left(\left(K' \oplus ipad \right) \parallel m \right) \right)$$

$$K' = \begin{cases} \mathrm{H}(K) & \text{if } K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

```
opad = 0x5c5c5c5c…
ipad = 0x36363636…
```

Length-extension attacks

- Why not simply $H(k \,||\, m)$?
- An attacker who knows $H(m_1)$ and length(m_1) can calculate $H(m_1 \,||\, m_2)$ for carefully chosen m_2 without needing to know m_1.
- Let $m_1 = k \,||\, m$. We can extend m with an m_2 without knowing k.
- Does this affect password salting?

Problem with symmetric encryption

- I develop the new SnapTok app
- I want users to encrypt their uploads
- I embed an AES key into the app and the same key on my servers
- The user encrypts their uploads with this AES key
- Problem?

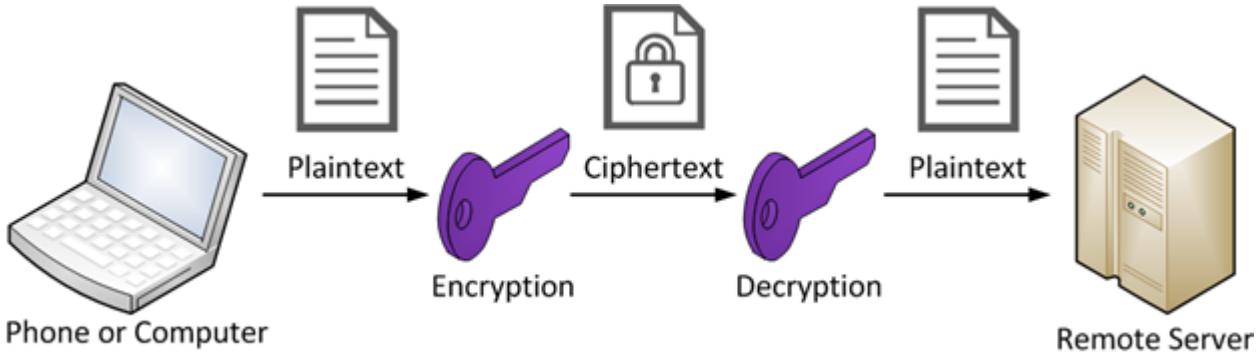
Problem with symmetric encryption

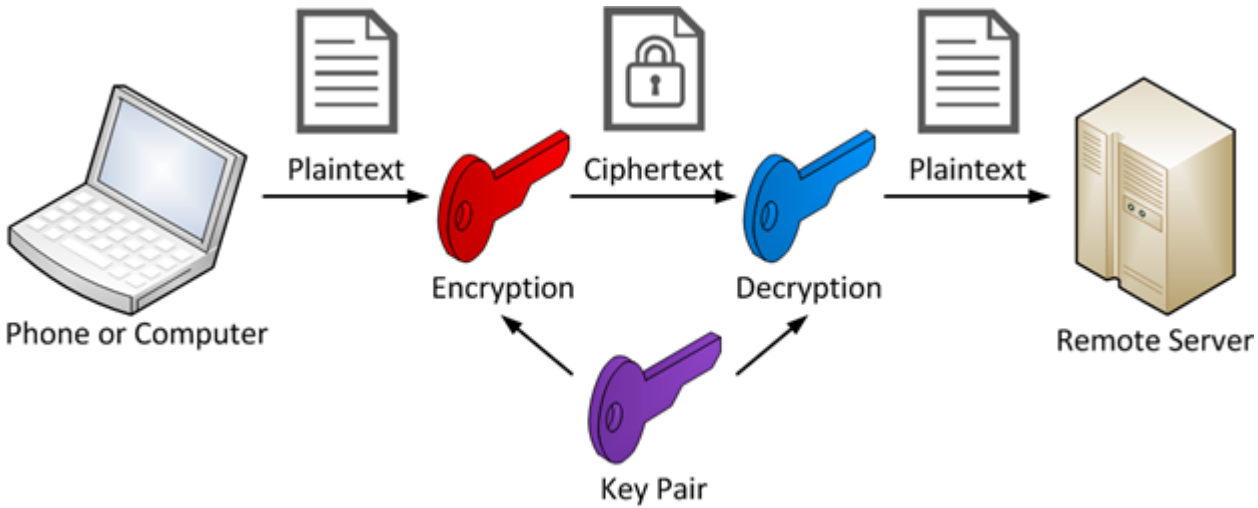
- How could we fix this? What do we wish we could do?

Alternatively

- I “sign” a contract with an HMAC and a key
- I give you the contract, the HMAC hash, and the key so you can verify the HMAC hash
- Problem?

Symmetric vs. asymmetric





Asymmetric cryptography

Encryption

- Public key: encrypt()
- Private key: decrypt()

Asymmetric cryptography

Digital signatures

- Public key: verify()
- Private key: sign()

Asymmetric cryptography

- Public key: not a secret
- Private key: must keep secret

Symmetric vs. asymmetric

- Symmetric: based on scrambling bits in ways we think provides confusion and diffusion
- Asymmetric: based on the difficulty of certain mathematical problems
- Both: outside of trivial cases (one-time pads), no proofs exist for their security

Symmetric vs. asymmetric

- Symmetric algorithms: Caesar cipher, substitution cipher, DES, 3DES, AES, etc.
- Asymmetric algorithms: RSA, DSA, ed25519 (elliptic curve)

Asymmetric algorithms

- Back to SnapTok
- I hard-code my public key into the app
- I use that to encrypt all of my users' data transmitted to my servers
- My servers have the private key
- Problem solved, right? Right!?

Why did we learn symmetric crypto?

- Asymmetric crypto is slow!
- You would need large keys to encrypt large messages!
- We still use symmetric crypto to encrypt everything
- But we use asymmetric crypto to handle the problem of symmetric key exchange

Why did we learn symmetric crypto?

- Asymmetric crypto is slow!
- You would need large keys to sign large messages!
- We still use hashes to hash everything
- But we use asymmetric crypto to handle the problem of authenticating the hash

Assignment 5

- Deadline: October 28, 2025, 10:00pm
- Real-world XSS attacks
  https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment5.pdf