IERG 4130 – 2026 Fall

香港中文大學
CUHK

Block Cipher Modes of Operation

Yajin ZHOU

https://yajin.org

Credits: the slides are from CS161 of UC Berkeley,

Summary: One-Time Pads



- **One-time pads**
  - Symmetric encryption scheme: Alice and Bob share a secret key.
  - Encryption and decryption: Bitwise XOR with the key.
  - No information leakage if the key is never reused.
  - Information leaks if the key is reused.
  - Impractical for real-world usage

Summary: Block Ciphers

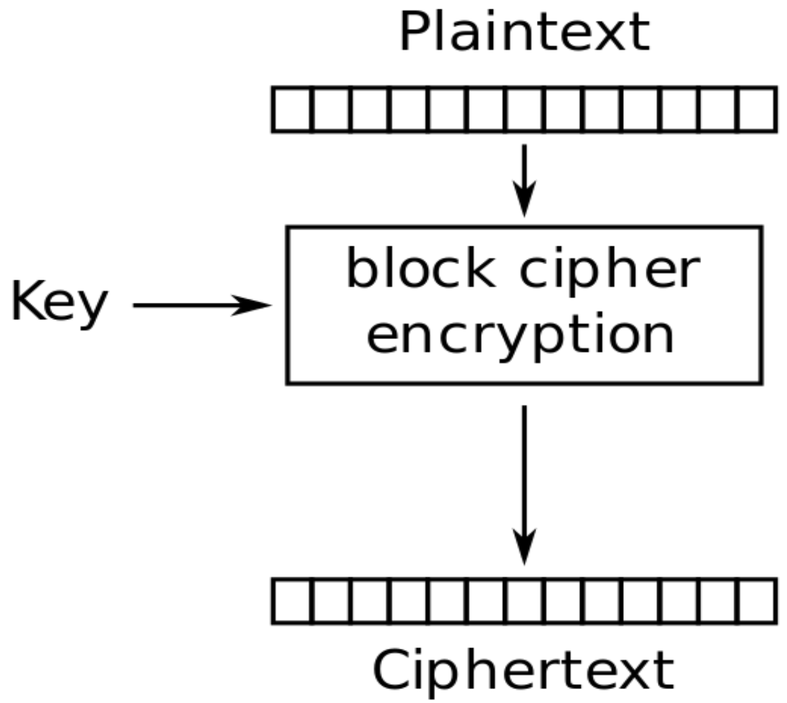


- **Encryption: input a *k*-bit key and *n*-bit plaintext, receive *n*-bit ciphertext**
- **Decryption: input a *k*-bit key and *n*-bit ciphertext, receive *n*-bit plaintext**
- **Correctness: when the key is fixed, $E_K(M)$ should be bijective**
- **Security**
  - Without the key, E_K(m) is computationally indistinguishable from a random permutation
  - Brute-force attacks take astronomically long and are not possible
- **Efficiency: algorithms use XORs and bit-shifting (very fast)**
- **Implementation: AES is the modern standard**
- **Issues**
  - Not IND-CPA secure because they're deterministic
  - Can only encrypt *n*-bit messages

Scratchpad: Let's design it together

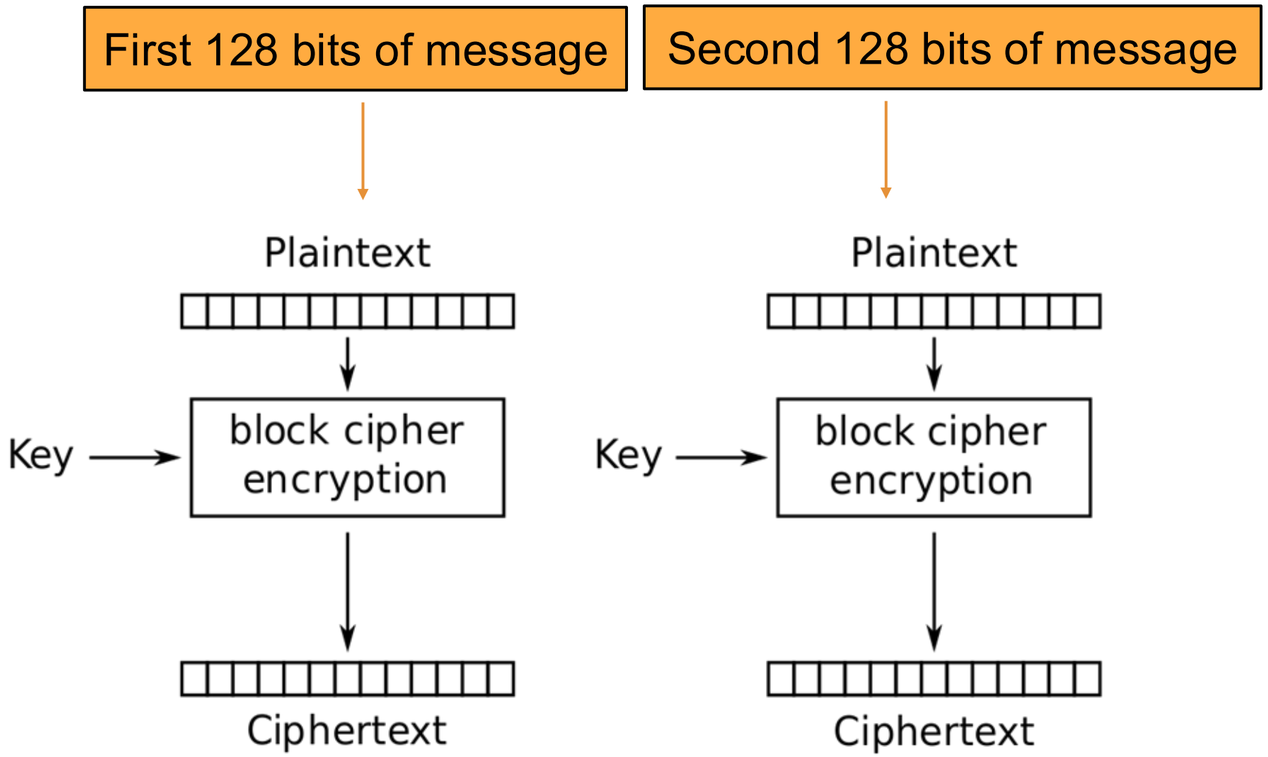
Here's an AES block. Remember that it can only encrypt 128-bit messages.

How can we use AES to encrypt a longer message (say, 256 bits?)



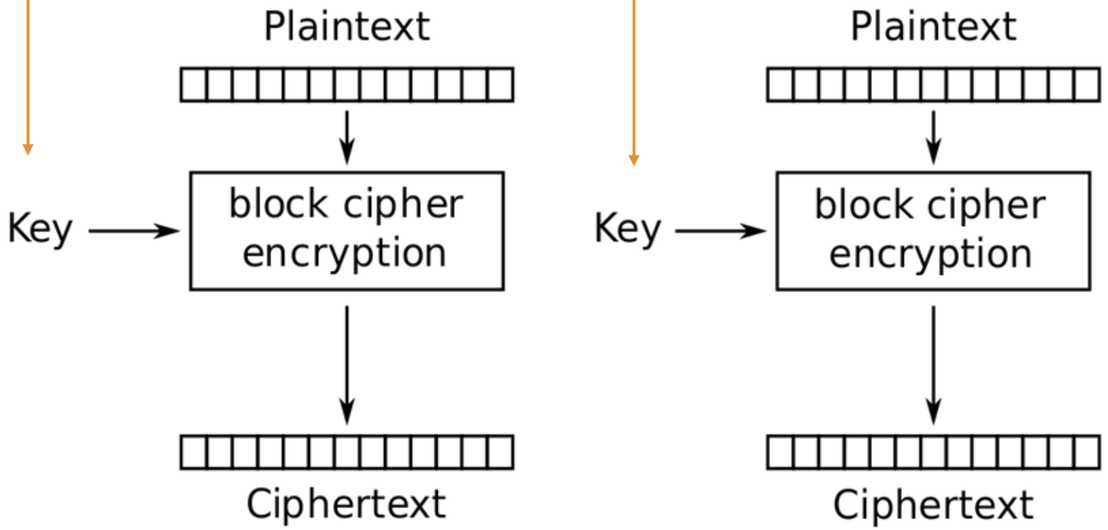
Scratchpad: Let's design it together

Idea: Let's use AES twice!



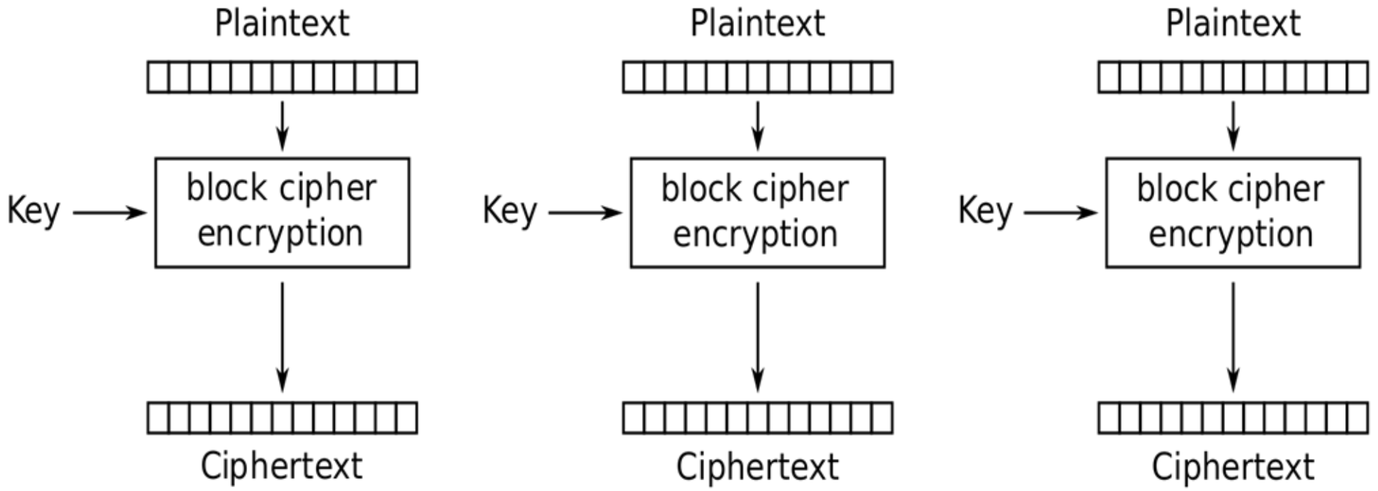
Scratchpad: Let's design it together

Note that we are using the same key twice. We want to avoid a situation like one-time pads where we need very long keys.



ECB Mode

- **We've just designed electronic code book (ECB) mode**
  - Enc(K, M) = C_1 || C_2 || … || C_m
  - Assume m is the number of blocks of plaintext in M, each of size n
- **AES-ECB is not IND-CPA secure. Why?**
  - Because ECB is deterministic

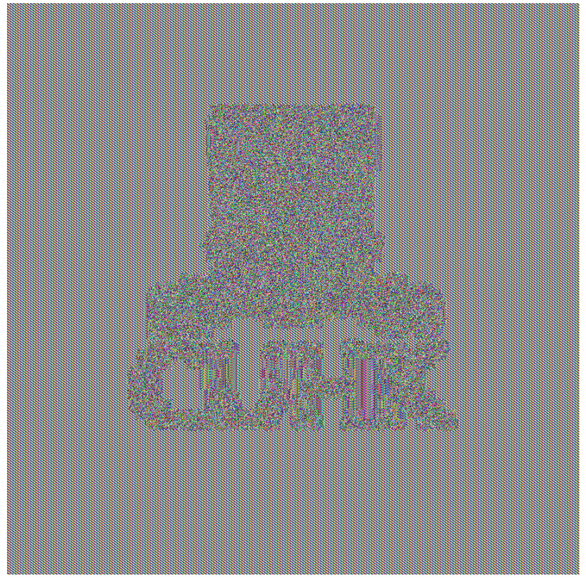


Electronic Codebook (ECB) mode encryption

ECB Mode

- Key is random key: dfa1de38d95a90ffd4ff40f113daa23f
  - It leaks the structure of the logo (why?)

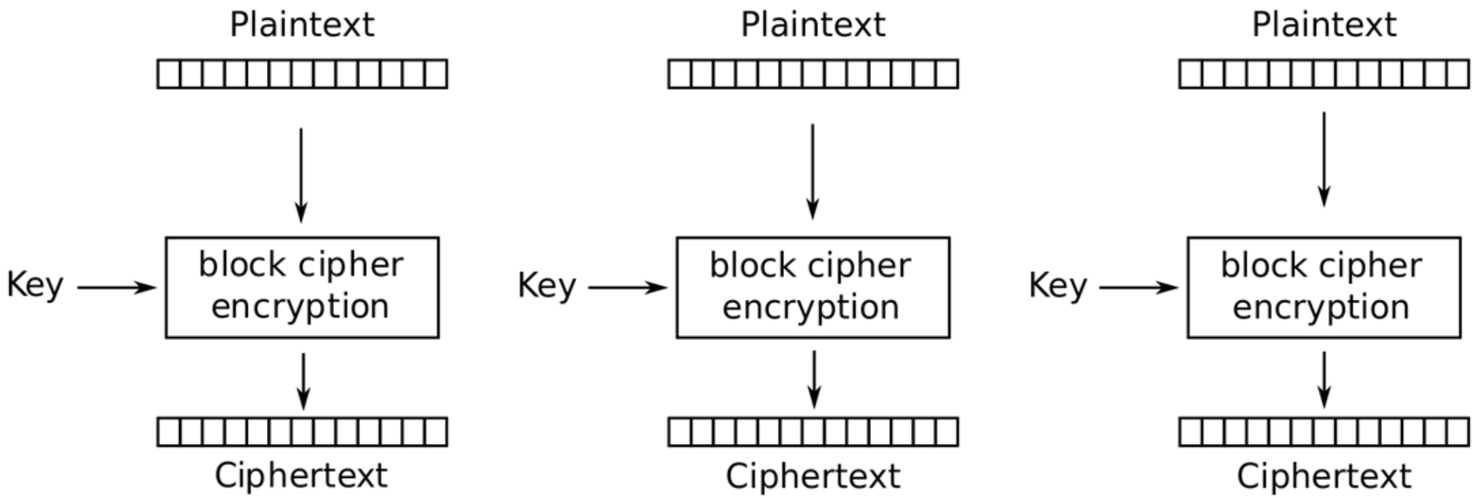




Scratchpad: Let's design it together

Here's ECB mode. It's not IND-CPA secure because it's deterministic.

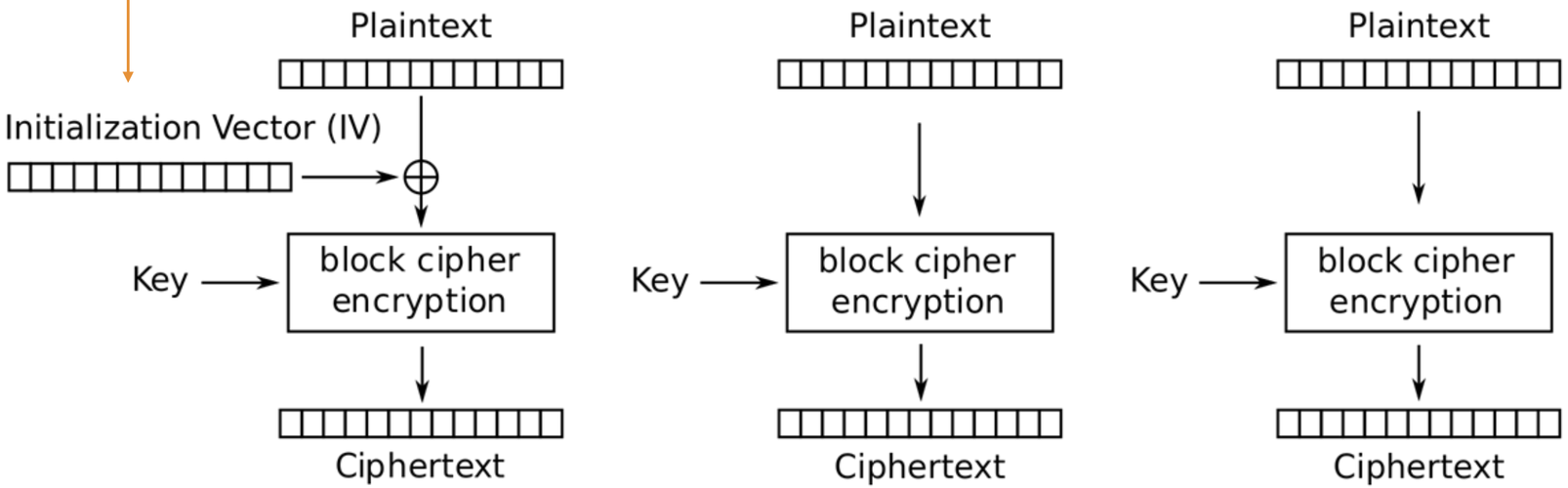
Let's fix that by adding some randomness.



Scratchpad: Let's design it together

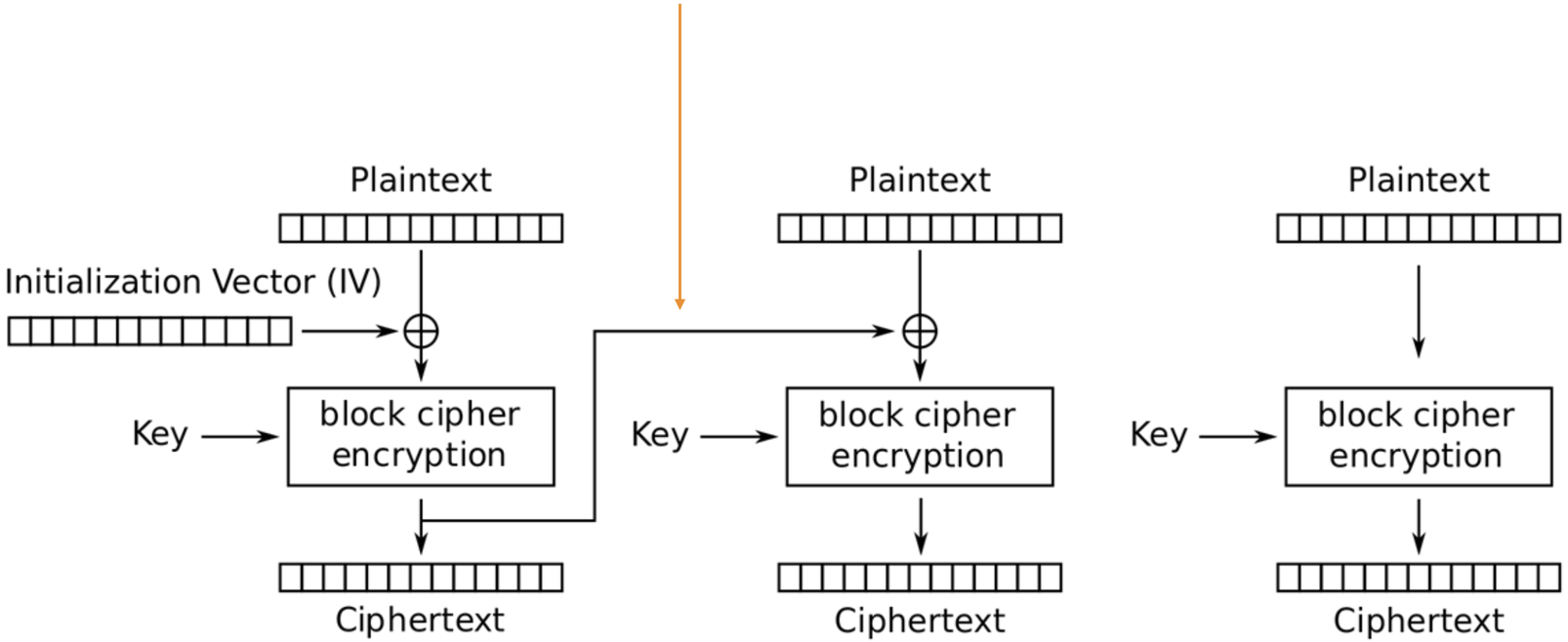
The Initialization Vector (IV) is different for every encryption. Now the first ciphertext block will be different for every encryption!

Okay, but the other blocks are still deterministic...



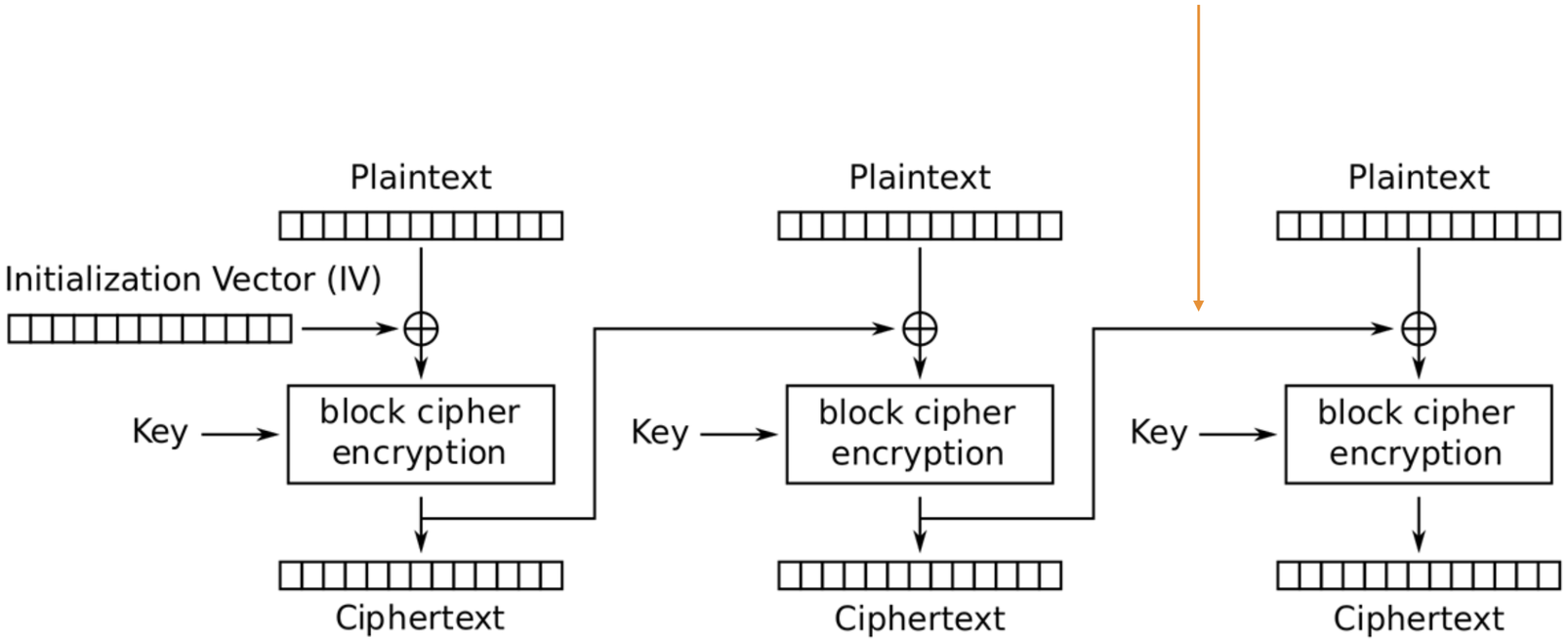
Scratchpad: Let's design it together

Idea: The first ciphertext block was computed with some randomness. Let's use it to add randomness to the second block.



Scratchpad: Let's design it together

Now the second ciphertext block has some randomness in it. Let's use it to add randomness to the third block.

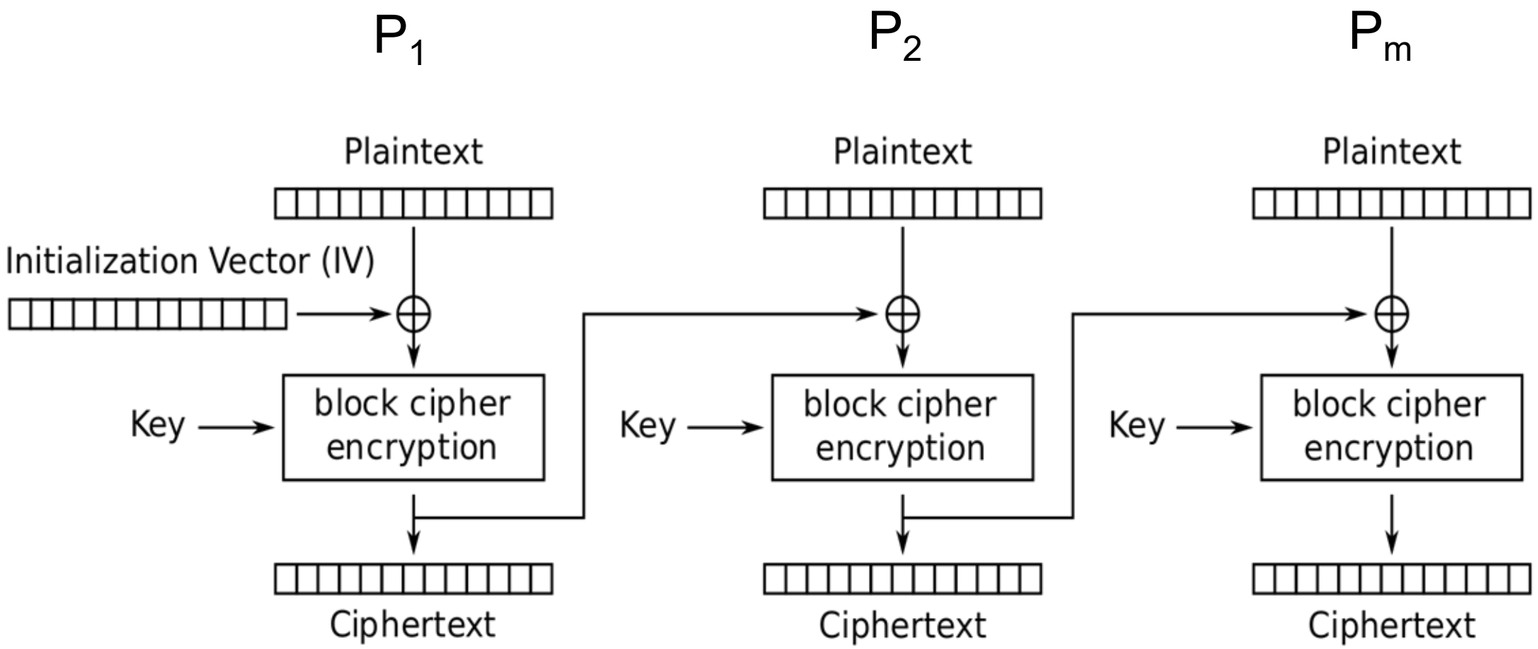


CBC Mode



- **We've just designed cipher block chaining (CBC) mode**
- **$C_i = E_K(M_i \oplus C_{i-1})$; C_0 = IV**
- **Enc(K, M):**
  - Split M in m plaintext blocks $P_1 \ldots P_m$ each of size n
  - Choose a random IV
  - Compute and output (IV, C_1, …, C_m) as the overall ciphertext
- **How do we decrypt?**

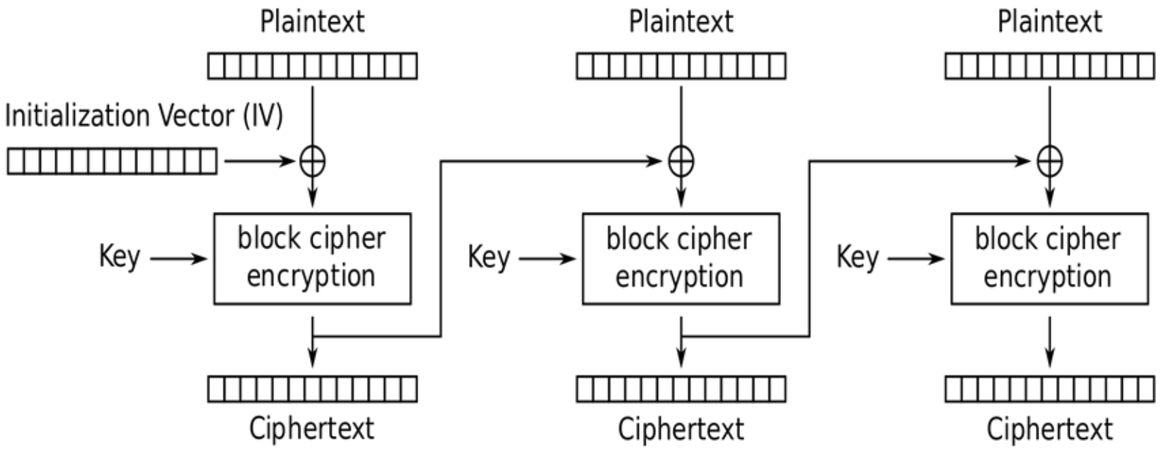
CBC Mode



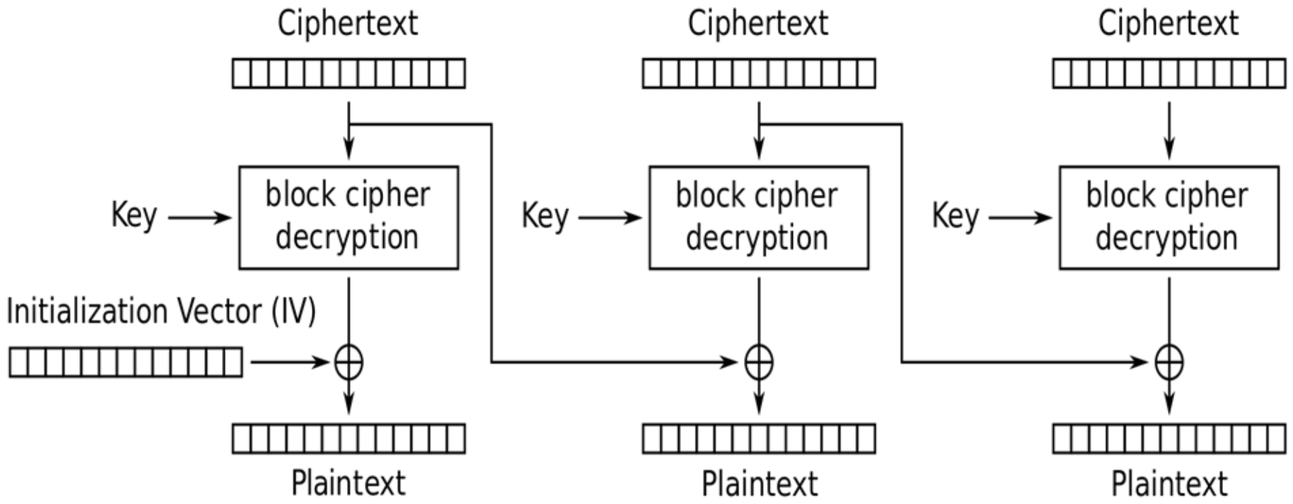
Cipher Block Chaining (CBC) mode encryption

CBC Mode: Decryption

- **How do we decrypt CBC mode?**
  - Parse ciphertext as (IV, C_1, …, C_m)
  - Decrypt each ciphertext and then XOR with IV or previous ciphertext



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

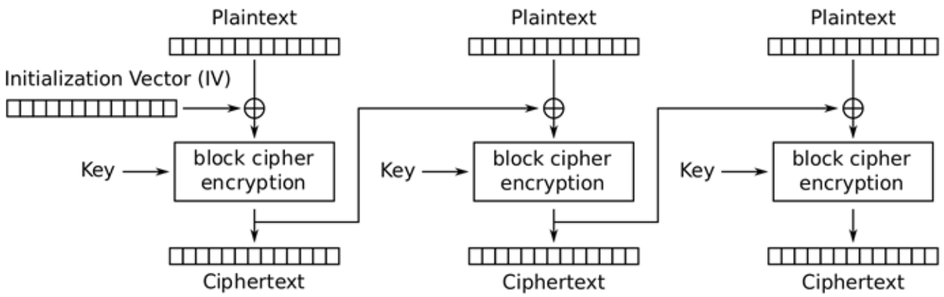
CBC Mode: Decryption

- **How do we decrypt CBC mode?**

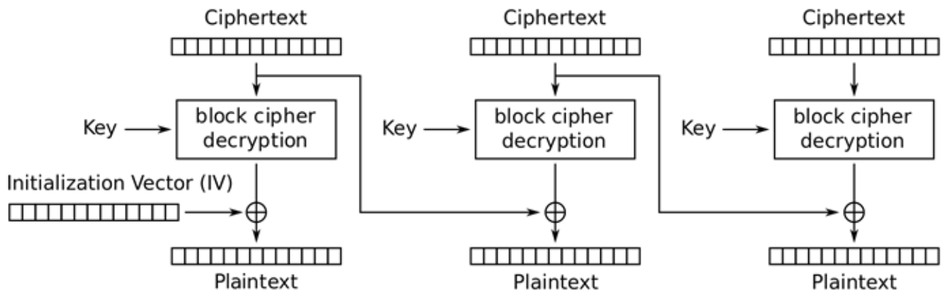
C_i	=	$E_K(M_i \oplus C_{i-1})$	Definition of encryption
$D_K(C_i)$	=	$D_K(E_K(M_i \oplus C_{i-1}))$	Decrypting both sides
$D_K(C_i)$	=	$M_i \oplus C_{i-1}$	Decryption and encryption cancel
$D_K(C_i) \oplus C_{i-1}$	=	$M_i \oplus C_{i-1} \oplus C_{i-1}$	XOR both sides with C_{i-1}
$D_K(C_i) \oplus C_{i-1}$	=	M_i	XOR property

CBC Mode: Efficiency & Parallelism

- **Can encryption be parallelized?**
  - No, we have to wait for block i to finish before encrypting block i+1
- **Can decryption be parallelized?**
  - Yes, decryption only requires ciphertext as input



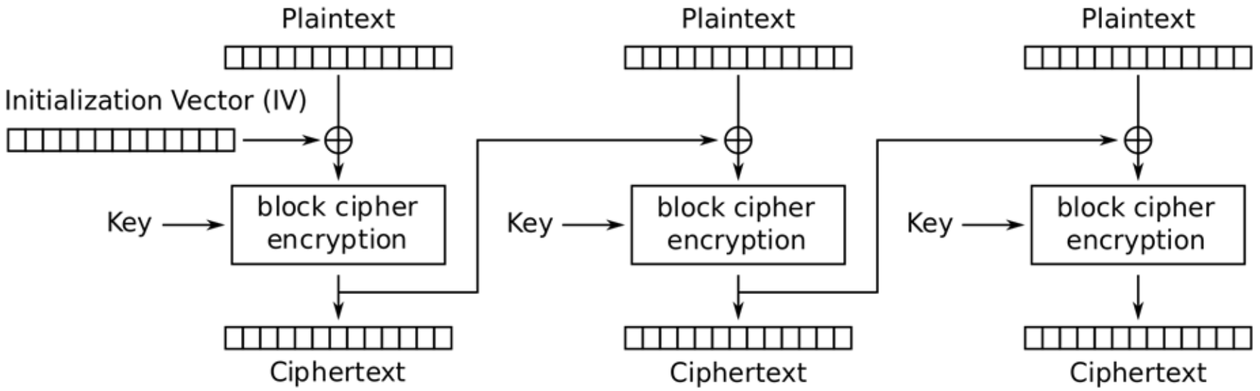
Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

CBC Mode: Padding

- **What if you want to encrypt a message that isn't a multiple of the block size?**
  - AES-CBC is only defined if the plaintext length is a multiple of the block size
- **Solution: Pad the message until it's a multiple of the block size**
  - **Padding**: Add dummy bytes at the end of the message until it's the proper length



Cipher Block Chaining (CBC) mode encryption

CBC Mode: Padding

- **What padding scheme should we use?**
  - Padding with 0 bits?
    - Doesn’t work: What if our message already ends with 0’s?
  - Padding with 1 bits?
    - Same problem

CBC Mode: Padding

- **We need a scheme that can be unpadded without ambiguity**
  - One scheme that works: Append a 1 bit, then pad with 0 bits
    - If plaintext is multiple of n, you still need to pad with an entire block
  - Example: suppose n = 8
    - 10110 -> 10110 100
      - How to recover: search the first 1 on the right side
      - How to distinguish with the original value 10110 100
    - 10110100 -> 10110100 10000000

CBC Mode: Padding



- **We need a scheme that can be unpadded without ambiguity**
  - Another scheme: Pad with the number of padding bytes
    - So if you need 1 byte, pad with **01**; if you need 3 bytes, pad with **03 03 03**
    - If you need 0 padding bytes, pad an entire dummy block
    - This is called PKCS #7

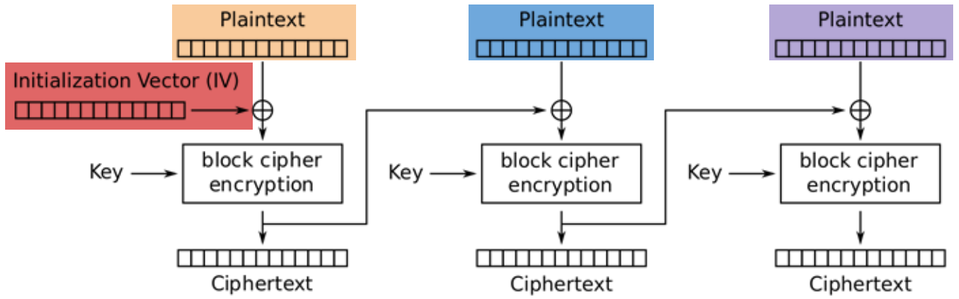
CBC Mode: Security



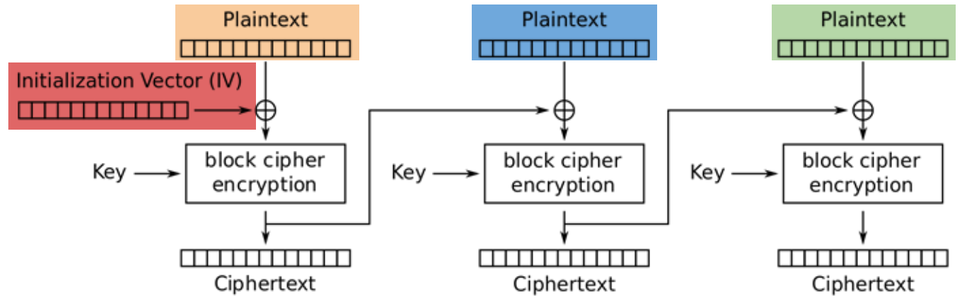
- AES-CBC is IND-CPA secure, assuming the IV is randomly generated and never reused
- **If you reuse the IV, the scheme becomes deterministic: No more IND-CPA security**

CBC Mode: IV Reuse

- **Consider two three-block messages: $P_1P_2P_3$ and $P_1P_2P_4$**
  - The first two blocks are the same for both messages, but the last block is different
  - What if we encrypt them with the same IV?
- When the IV is reused, CBC mode reveals **when two messages start with the same plaintext blocks, up to the first different plaintext block**



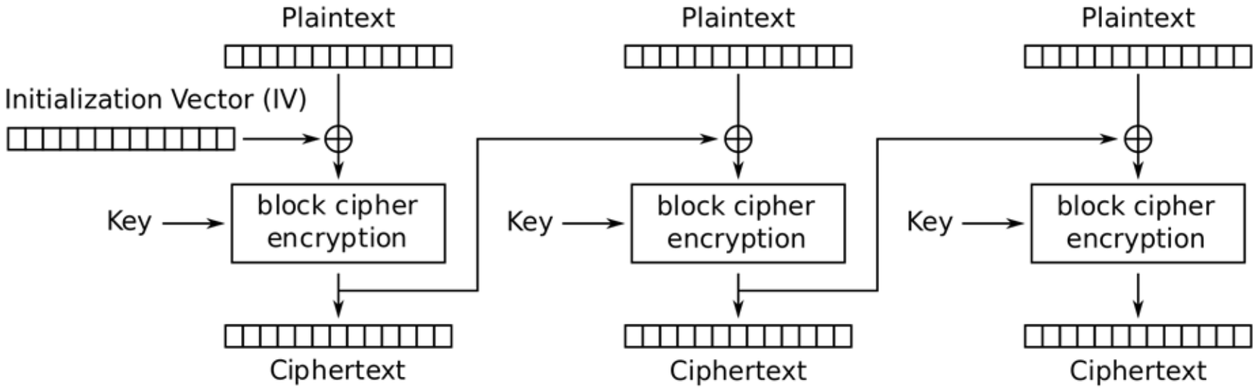
Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode encryption

CBC Mode is IND-CPA (when used correctly)

- **Enc(*K*, *M*):**
  - Split M in m plaintext blocks $P_1 \ldots P_m$ each of size *n*
  - Choose random IV, compute and output (IV, C_1, …, C_m) as the overall ciphertext
- **CBC mode is IND-CPA-secure**
  - If there exists an attacker that wins in the IND-CPA game, then there exists an attacker that breaks the block cipher security. Proof is out of scope.

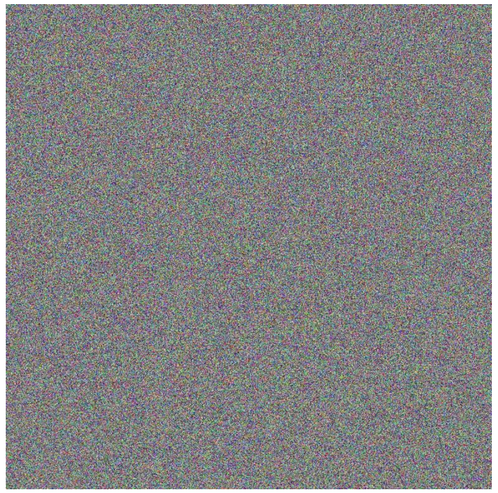


Cipher Block Chaining (CBC) mode encryption

CBC Mode

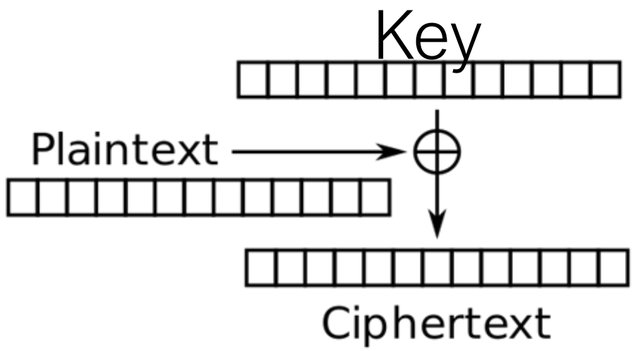
- Key is random key: dfa1de38d95a90ffd4ff40f113daa23f
- IV：b49e92b2248a8a7a148e92f798c9f508



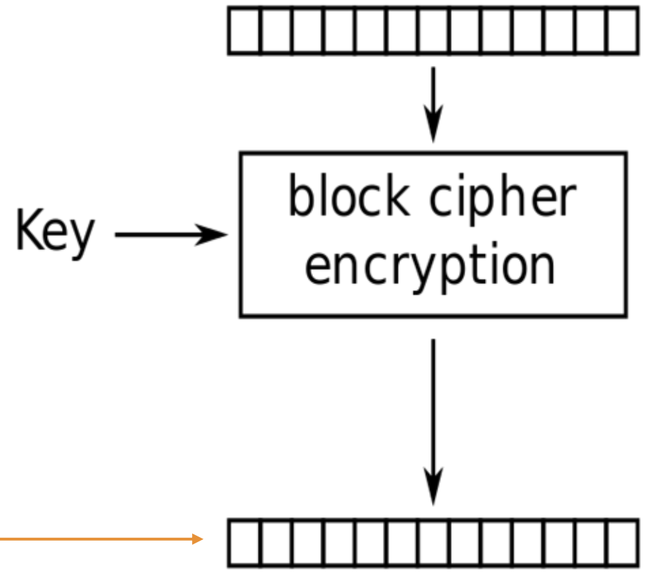


CTR Mode Scratchpad: Let's design it together

One-time pads are secure if we never reuse the key.



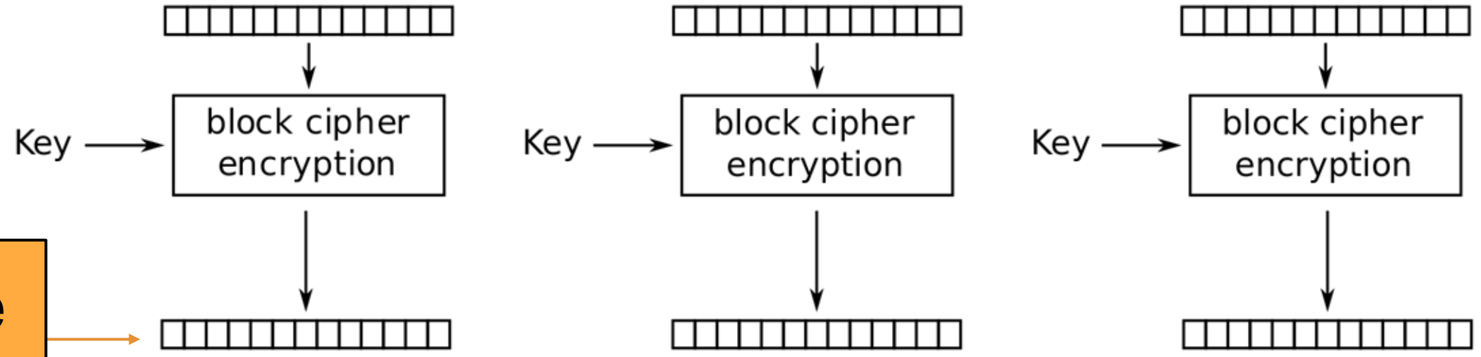
CTR Mode Scratchpad: Let’s design it together



If the attacker doesn’t know the key, the block cipher output looks random.

New idea: Let's use block ciphers to simulate a one-time pad!

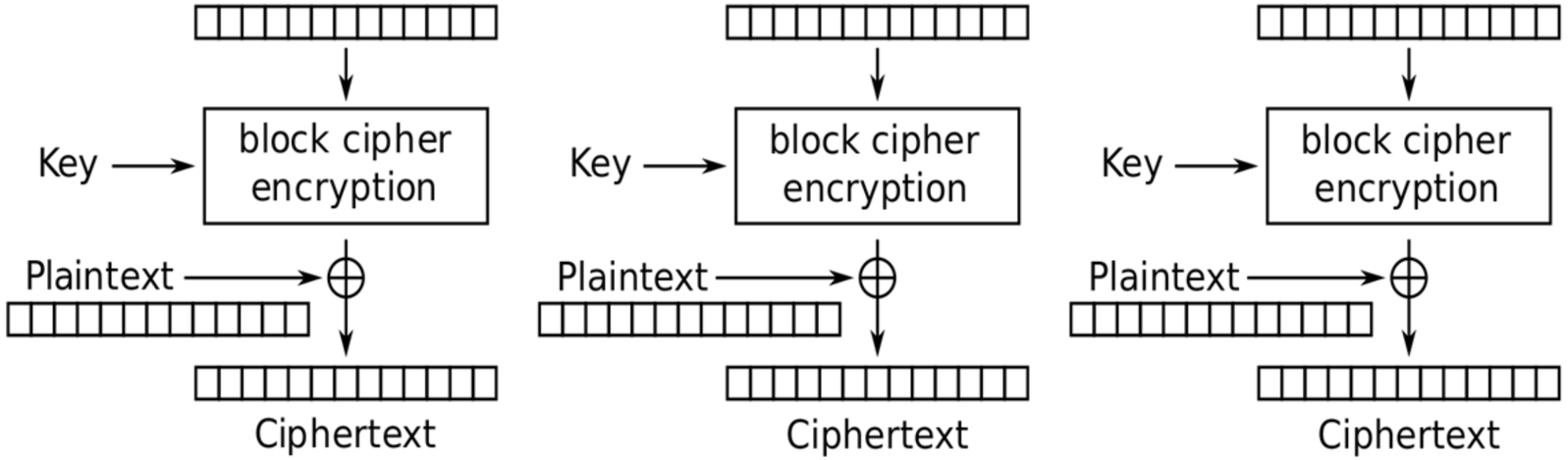
CTR Mode Scratchpad: Let's design it together



If the attacker doesn't know the key, all of these outputs look random.

CTR Mode Scratchpad: Let's design it together

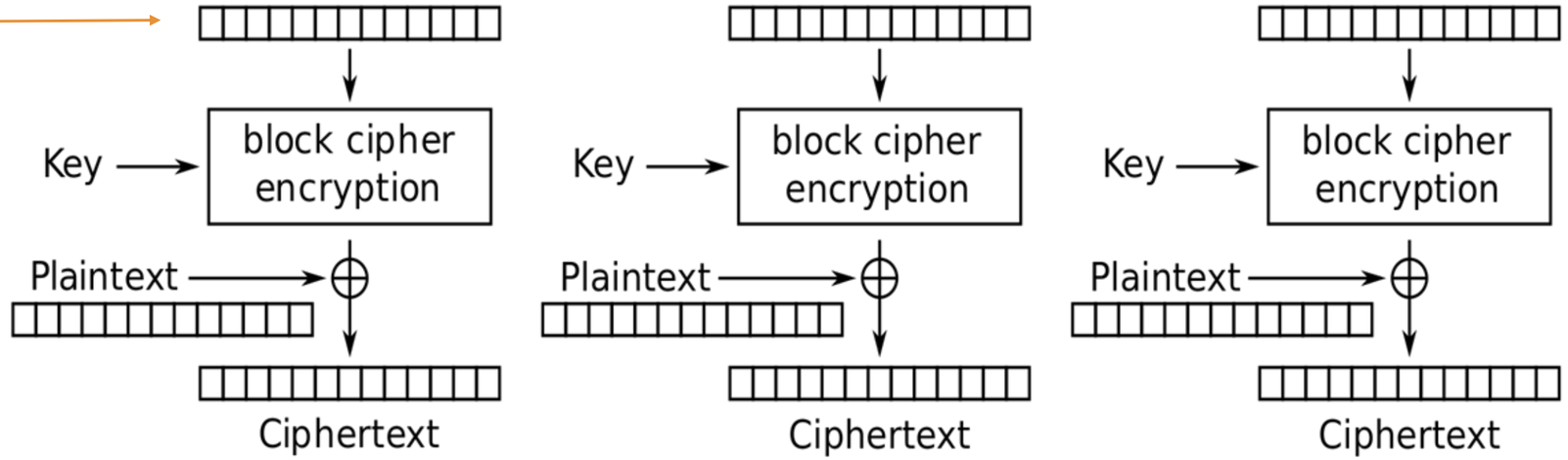
Idea: Use this random-looking output as a one-time pad!



Remember one-time pads: XOR the pad with plaintext to get ciphertext

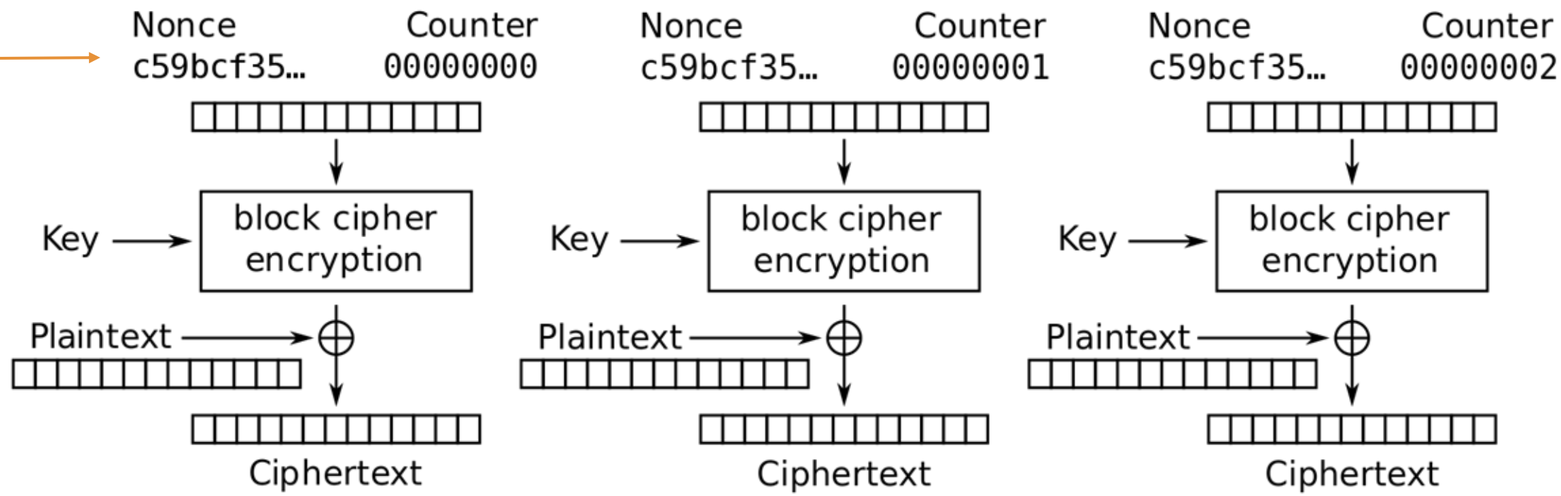
CTR Mode Scratchpad: Let's design it together

What do we use as input to the block cipher?



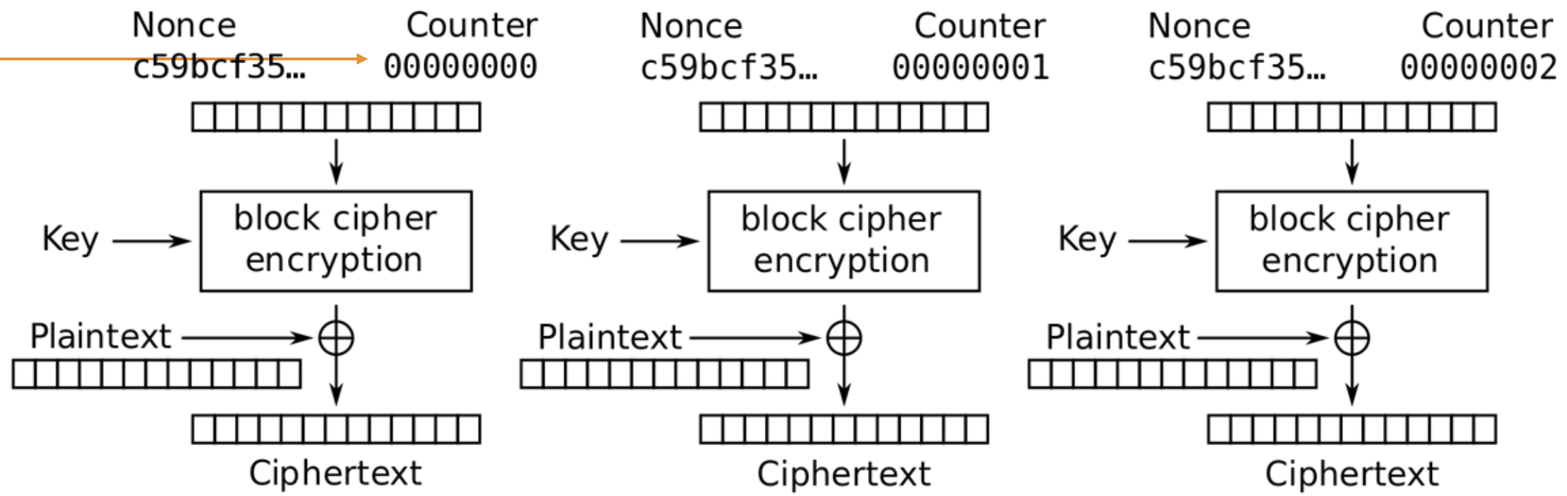
CTR Mode Scratchpad: Let's design it together

IND-CPA schemes need randomness, so let's put a random **nonce** here!



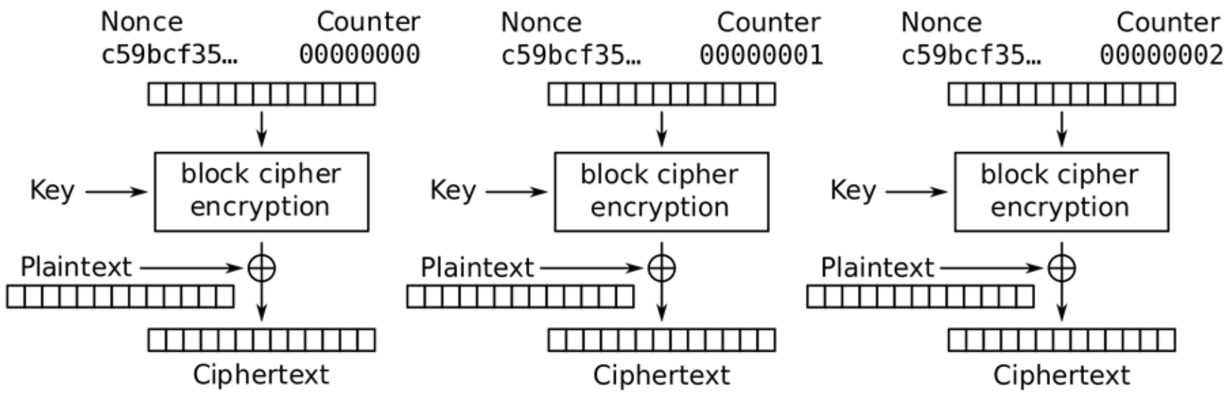
CTR Mode Scratchpad: Let's design it together

The **counter** increments per block to ensure each block cipher output is different



CTR (Counter) Mode

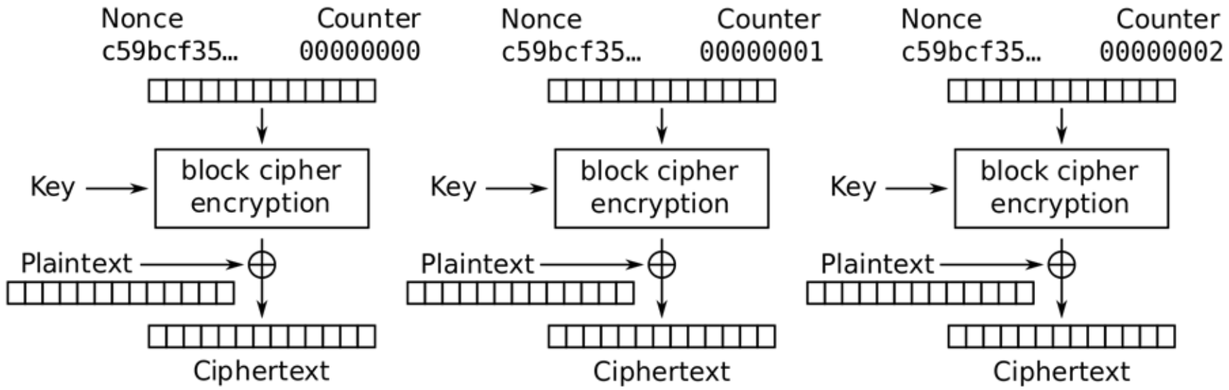
- The random value is named the nonce here, but the idea is the same as the IV in CBC mode
- **Overall ciphertext is (Nonce, C_1, …, C_m)**



C_1 Counter (CTR) mode encryption C_m

CTR (Counter) Mode

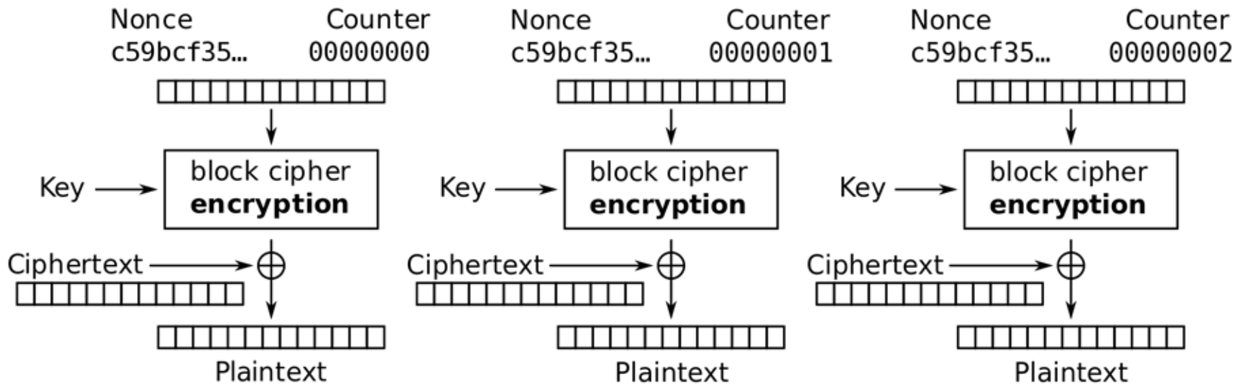
- **Enc(K, M):**
  - Split M in plaintext blocks $P_1 ... P_m$ (each of block size n)
  - Choose random nonce
  - Compute and output (Nonce, C_1, …, C_m)
- **How do you decrypt?**



C_1 Counter (CTR) mode encryption C_m

CTR (Counter) Mode

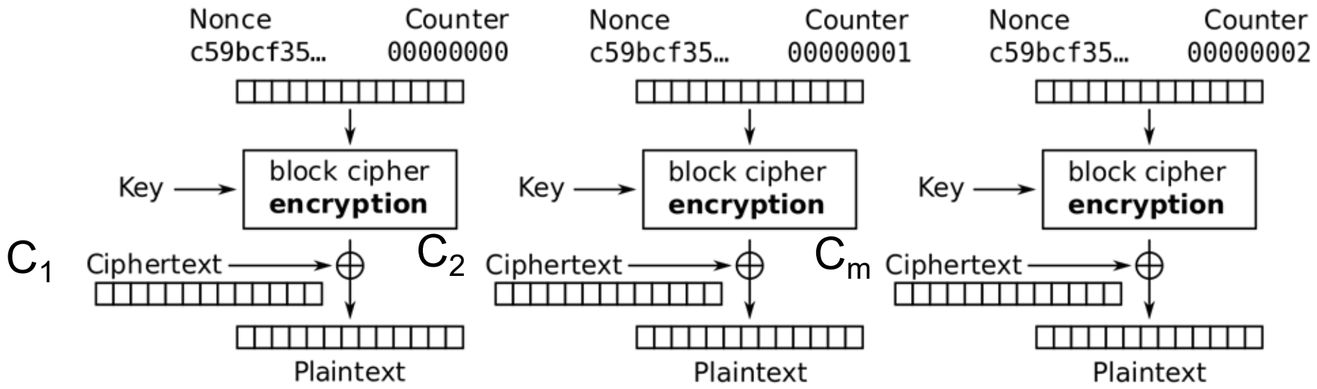
- Recall one-time pad: XOR with ciphertext to get plaintext
- We are only using block cipher encryption, not decryption



Counter (CTR) mode decryption

CTR (Counter) Mode

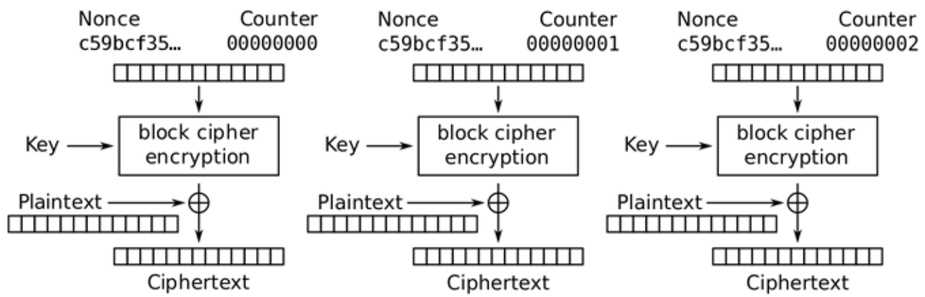
- **Dec(K, C):**
  - Parse C into (nonce, C_1, …, C_m)
  - Compute P_i by XORing Ci with output of E_k on nonce and counter
  - Concatenate resulting plaintexts and output M = P_1 … P_m



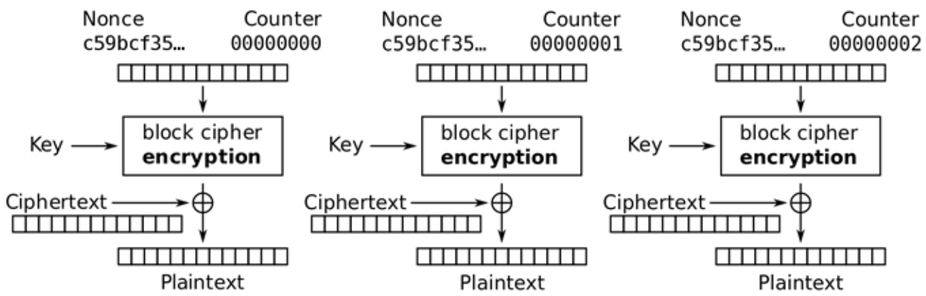
Counter (CTR) mode decryption

CTR Mode: Efficiency

- **Can encryption be parallelized?**
  - Yes
- **Can decryption be parallelized?**
  - Yes



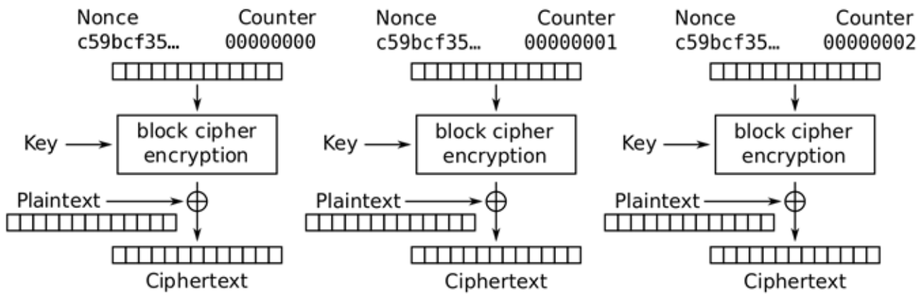
Counter (CTR) mode encryption



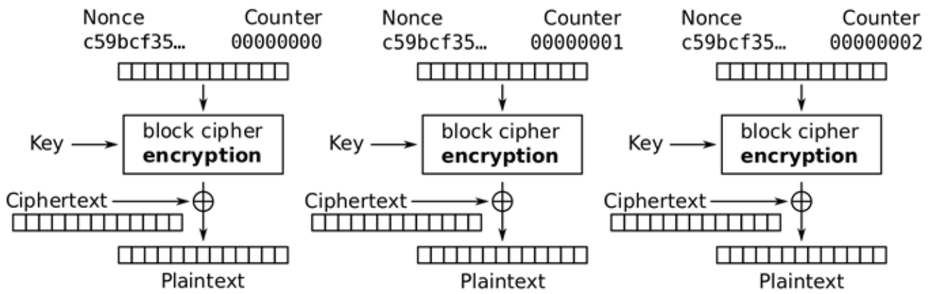
Counter (CTR) mode decryption

CTR Mode: Padding

- **Do we need to pad messages?**
  - No! We can just cut off the parts of the XOR that are longer than the message.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

CTR Mode: Security

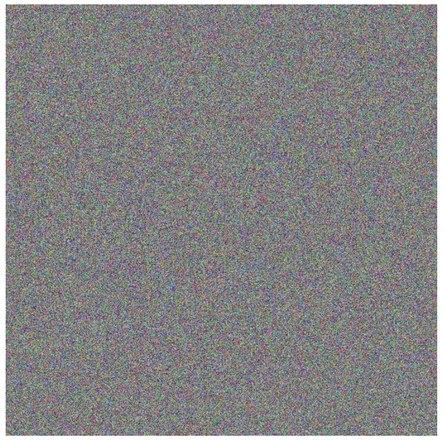


- AES-CTR is IND-CPA secure, assuming the nonce is randomly generated and never reused
- **Reusing a nonce is equivalent to reusing a key in a one-time pad (very bad)**
  - Recall: Key reuse in a one-time pad is catastrophic: usually leaks enough information for an attacker to deduce the entire plaintext

CTR Mode

- Key is random key: dfa1de38d95a90ffd4ff40f113daa23f
- Nonce / counter block：5c8e0ea7646c64fd7b70259677fd74ba





IVs and Nonces



- **Initialization vector** (**IV**): A random, but public, one-use value to introduce randomness into the algorithm
  - For CTR mode, we say that you use a **nonce** (number used once), since the value has to be unique, not necessarily random.
  - In this class, we use IV and nonce interchangeably
- **Never reuse IVs**
  - In some algorithms, IV/nonce reuse leaks limited information (e.g. CBC)
  - In some algorithms, IV/nonce reuse leads to catastrophic failure (e.g. CTR)

IVs and Nonces



- **Thinking about the consequences of IV/nonce reuse is hard**
- **What if the IV/nonce is not reused, but the attacker can predict future values?**
  - Now you have to think about more attacks
  - We'll analyze this more in discussion: it really depends on the encryption function
- **Solution: Randomly generate a new IV/nonce for every encryption**
  - If the nonce is 128 bits or longer, the probability of generating the same IV/nonce twice is astronomically small (basically 0)
  - Now you don't ever have to think about IV/nonce reuse attacks!

Comparing Modes of Operation



- If you need high performance, CTR mode is arguably better, because you can parallelize both encryption and decryption
- **If you're paranoid about security, CBC mode is arguably better**
- **Theoretically, CBC and CTR mode are equally secure if used properly**
  - However, if used improperly (IV/nonce reuse), **CBC only leaks partial information, and CTR fails catastrophically**
    - Consider human factors: Systems should be as secure as possible even when implemented *incorrectly*
  - IV failures on CTR mode have resulted in multiple real-world security incidents!

Comparing Modes of Operation



- CTR:
  - $C_1 = P_1 \oplus K$
  - $C_2 = P_2 \oplus K$
  - $C_1 \oplus C_2 = P_1 \oplus P_2$
  - If attack knows C1, then he can infer C2
- When the IV is reused, CBC mode reveals **when two messages start with the same plaintext blocks, up to the first different plaintext block**

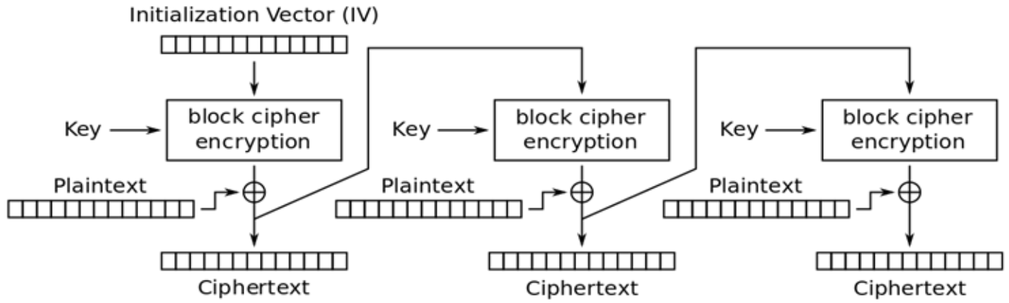
Other Modes of Operation



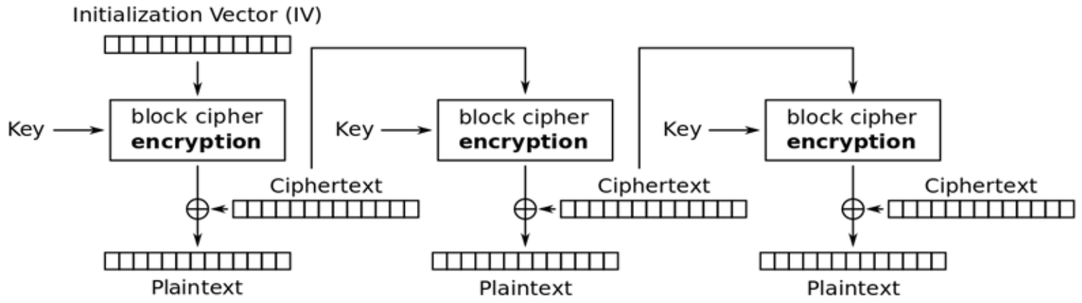
- **Other modes exist besides CBC and CTR**
- **Trade-offs:**
  - Do we need to pad messages?
  - How robust is the scheme if we use it incorrectly?
  - Can we parallelize encryption/decryption?

CFB

- **Also IND-CPA**
- **Try to analyze the trade-offs yourself:**
  - Do we need to pad messages?
  - How robust is the scheme if we use it incorrectly?
  - Can we parallelize encryption/decryption?



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

CFB

- **Try to analyze the trade-offs yourself:**
  - Do we need to pad messages?
    - No
  - How robust is the scheme if we use it incorrectly?
    - Similar effects as CBC mode, but a bit worse if you reuse the IV
  - Can we parallelize encryption/decryption?
    - Only decryption is parallelizable

Lack of Integrity and Authenticity



- Block ciphers are designed for *confidentiality* (IND-CPA)
- If an attacker tampers with the ciphertext, we are not guaranteed to detect it
- Remember Mallory: An *active* manipulator who wants to tamper with the message

Lack of Integrity and Authenticity

- **Consider CTR mode**
- **What if Mallory tampers with the ciphertext using XOR?**

M	P	a	y		M	a	l		$	1	0	0
	0x50	0x61	0x79	0x20	0x4d	0x61	0x6c	0x20	0x24	0x31	0x30	0x30
						⊕						
EK(i)	0x8a	0xe3	0x5e	0xcf	0x3b	0x40	0x46	0x57	0xb8	0x69	0xd2	0x96
						=						
C	0xda	0x82	0x27	0xef	0x76	0x21	0x2a	0x77	0x9c	0x58	0xe2	0xa6

Lack of Integrity and Authenticity

- **Suppose Mallory knows the message *M***
- **How can Mallory change the *M* to say Pay Mal $900?**

M	P	a	y		M	a	l		$	**1**	0	0
	0x50	0x61	0x79	0x20	0x4d	0x61	0x6c	0x20	0x24	**0x31**	0x30	0x30
						⊕						
EK(i)	0x8a	0xe3	0x5e	0xcf	0x3b	0x40	0x46	0x57	0xb8	**0x69**	0xd2	0x96
						=						
C	0xda	0x82	0x27	0xef	0x76	0x21	0x2a	0x77	0x9c	**0x58**	0xe2	0xa6

Lack of Integrity and Authenticity

$C_i = M_i \oplus Pad_i$	$0x58 = 0x31 \oplus Pad_i$	Definition of CTR
$Pad_i = M_i \oplus C_i$	$Pad_i = 0x58 \oplus 0x31$ $= 0x69$	Solve for the ith byte of the pad
$C'_i = M'_i \oplus Pad_i$	$C'_i = 0x39 \oplus 0x69$ $= 0x50$	Compute the changed ith byte

C	0xda	0x82	0x27	0xef	0x76	0x21	0x2a	0x77	0x9c	**0x58**	0xe2	0xa6
C’	0xda	0x82	0x27	0xef	0x76	0x21	0x2a	0x77	0x9c	**0x50**	0xe2	0xa6

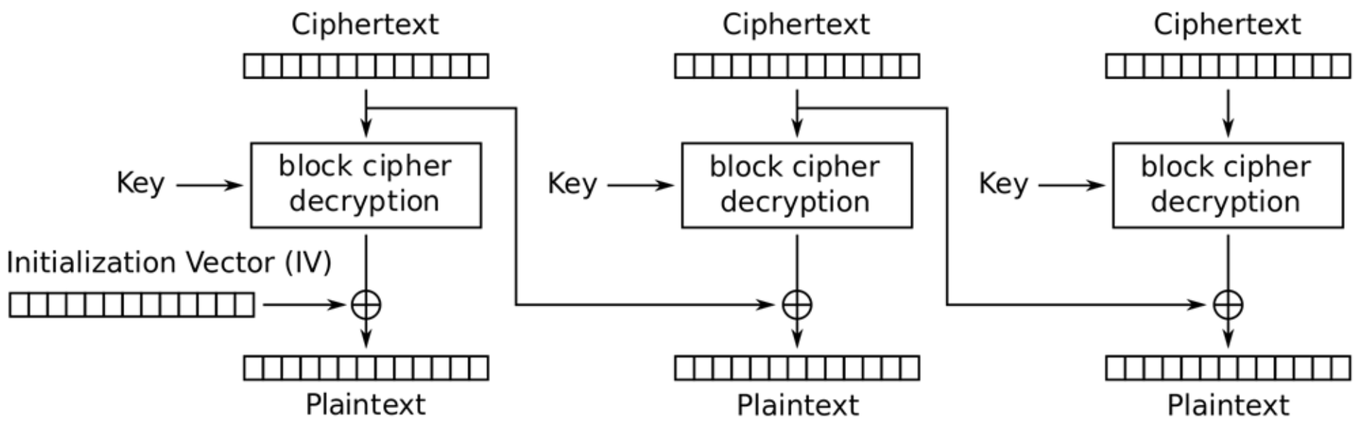
Lack of Integrity and Authenticity

- **What happens when we decrypt *C*'?**
  - The message looks like “Pay Mal $900” now!
  - Note: Mallory didn’t have to know the key; no integrity or authenticity for CTR mode!

C’	0xda	0x82	0x27	0xef	0x76	0x21	0x2a	0x77	0x9c	**0x50**	0xe2	0xa6
							⊕					
EK(i)	0x8a	0xe3	0x5e	0xcf	0x3b	0x40	0x46	0x57	0xb8	**0x69**	0xd2	0x96
							=					
P’	P	a	y		M	a	l		$	**9**	0	0
	0x50	0x61	0x79	0x20	0x4d	0x61	0x6c	0x20	0x24	**0x39**	0x30	0x30

Lack of Integrity and Authenticity

- **What about CBC?**
  - Altering a bit of the ciphertext causes some blocks to become random gibberish
  - However, Bob cannot prove that Alice did not send random gibberish, so it still does *not* provide integrity or authenticity



Cipher Block Chaining (CBC) mode decryption

Summary



- **ECB mode: Deterministic, so not IND-CPA secure**
- **CBC mode**
  - IND-CPA secure, assuming no IV reuse
  - Encryption is not parallelizable
  - Decryption is parallelizable
  - Must pad plaintext to a multiple of the block size
  - IV reuse leads to leaking the existence of identical blocks at the start of the message

Summary



- **CTR mode**
  - IND-CPA secure, assuming no IV reuse
  - Encryption and decryption are parallelizable
  - Plaintext does not need to be padded
  - Nonce reuse leads to losing all security
- **Lack of integrity and authenticity**