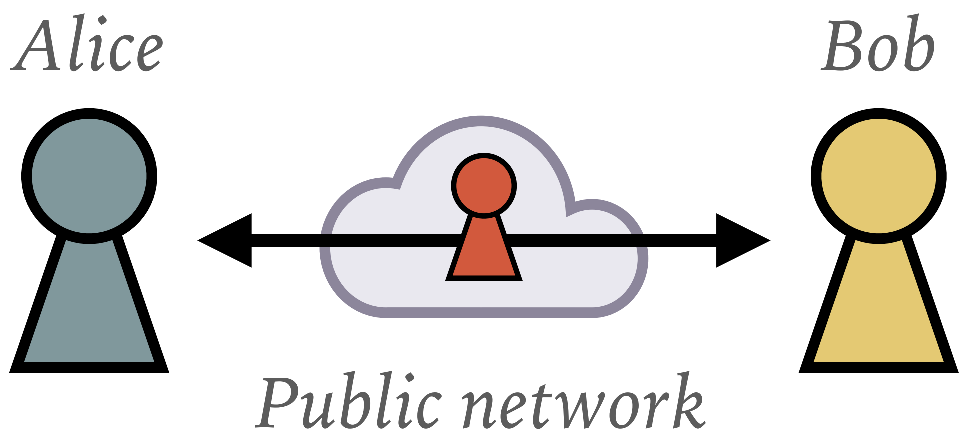
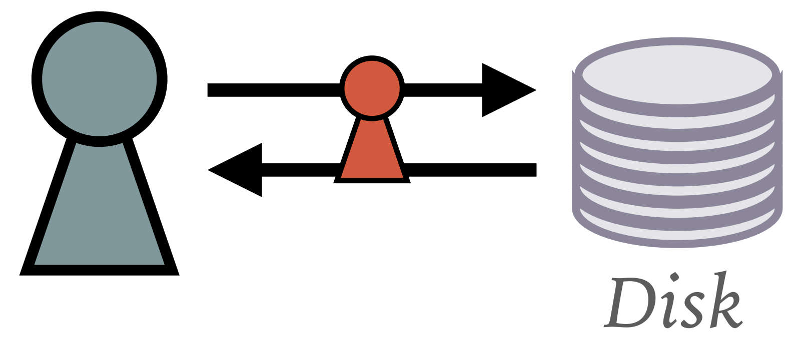
CRYPTOGRAPHY
INTRO

CMSC 414

MAR 13 2018

SCENARIOS AND GOALS



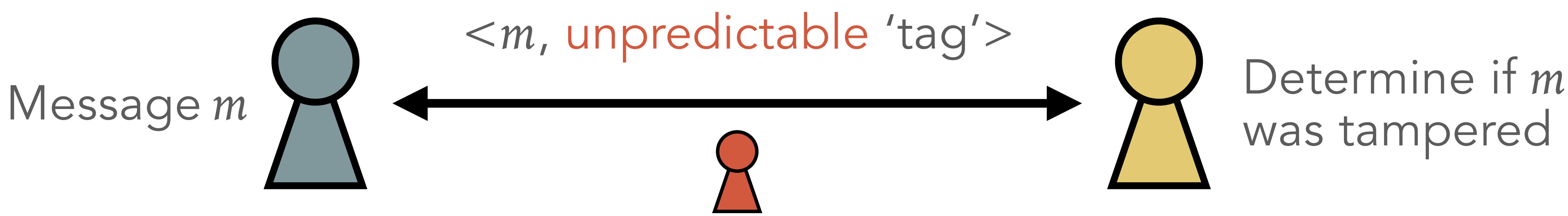


CONFIDENTIALITY Keep others from reading Alice's messages / data

INTEGRITY Keep others from undetectably tampering with Alice's messages / data

AUTHENTICITY Keep others from undetectably impersonating Alice (keep her to her word, too)

RANDOMNESS

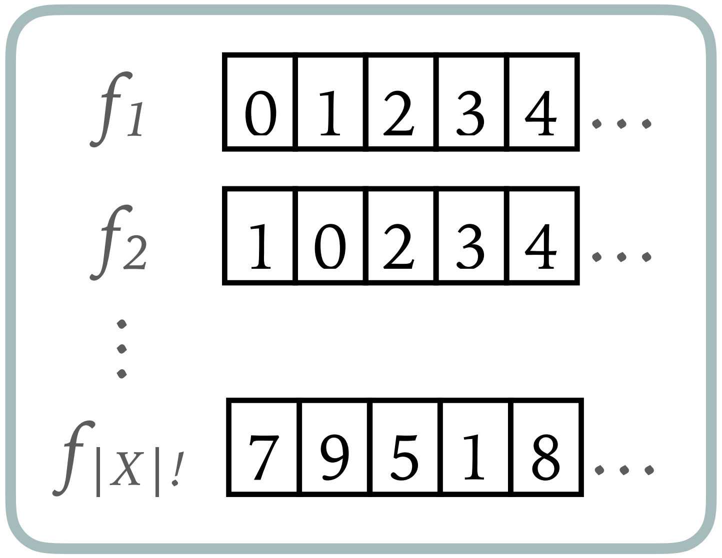


Ideally, to the attacker, it is indistinguishable from a string of bits chosen uniformly at random

This will be impossible with Alice and Bob having a shared secret

WHAT WE IDEALLY HAVE: RANDOM FUNCTIONS

Consider the set of all permutations $f_i : X \to X$



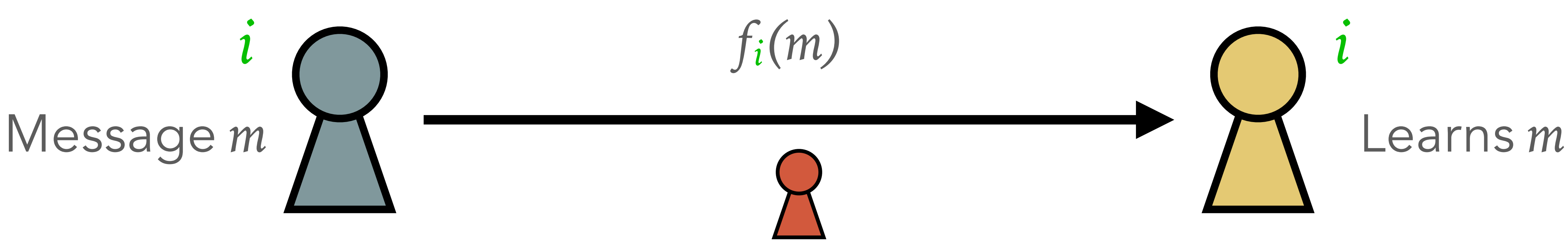
Think of X as all
128-bit bit strings

If you know i, then $f_i(x)$ is trivial to invert

If you *don't* know i, then $f_i(x)$ is one-way

"One-way trapdoor function"

Shared secret: index i chosen u.a.r.

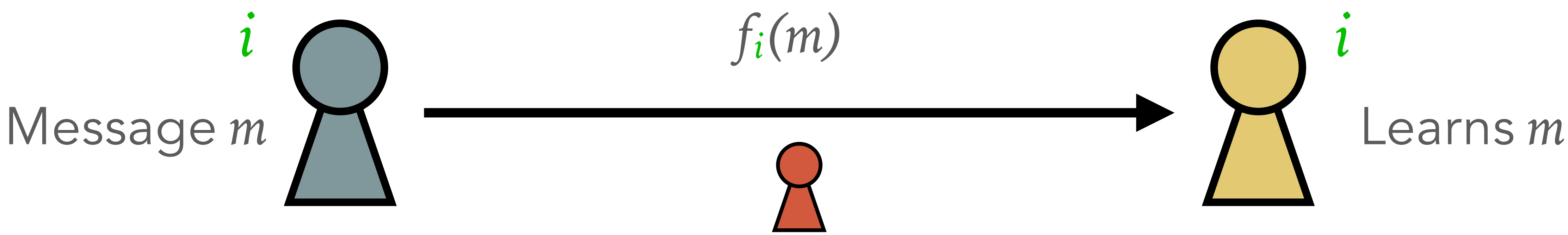


Without knowing i,
learns nothing about m

i is our key

WHAT WE IDEALLY HAVE: RANDOM FUNCTIONS

Shared secret: index i chosen u.a.r.



Without knowing i,
learns nothing about m

In essence, this protocol is saying "Let's use the i^{th} permutation function"

Infeasible to store all permutation functions

So instead cryptographers construct *pseudorandom functions*

HOW WE WILL BE COVERING CRYPTO

414 != 456

CMSC 456 (Crypto) covers how to build, analyze, and break cryptosystems

In this class, we will cover how to *use* them

BLACKBOXES

To this end, we'll cover several "blackboxes": what properties do they provide, and how can we responsibly put them together

Block ciphers | **MACs** | **Hash functions** | **Public key crypto**

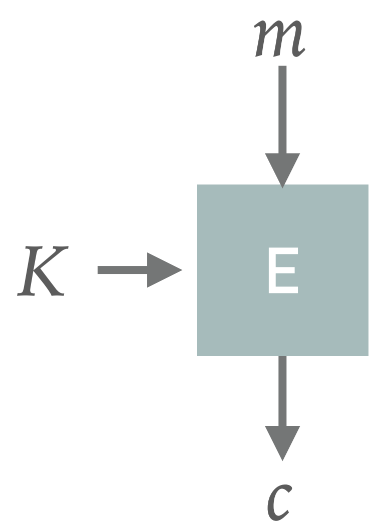
CMSC 456 opens these blackboxes up; it's awesome!
(but not what we're doing)

BLACKBOX #1:
BLOCK CIPHERS

BLOCK CIPHERS

ENCRYPTION

Key



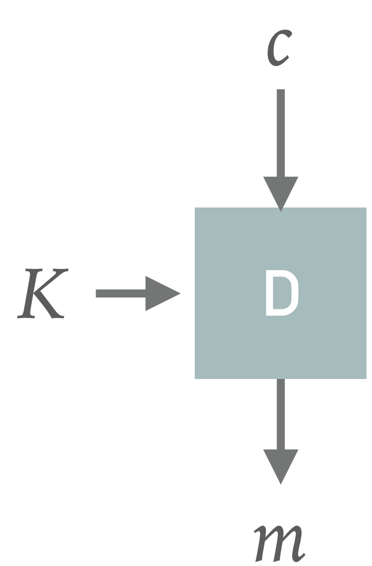
Plaintext ("message")

Ciphertext

Same fixed block size
(AES: 128 bits, 3DES: 64 bits)

AES key sizes:
128, 192, 256

DECRYPTION



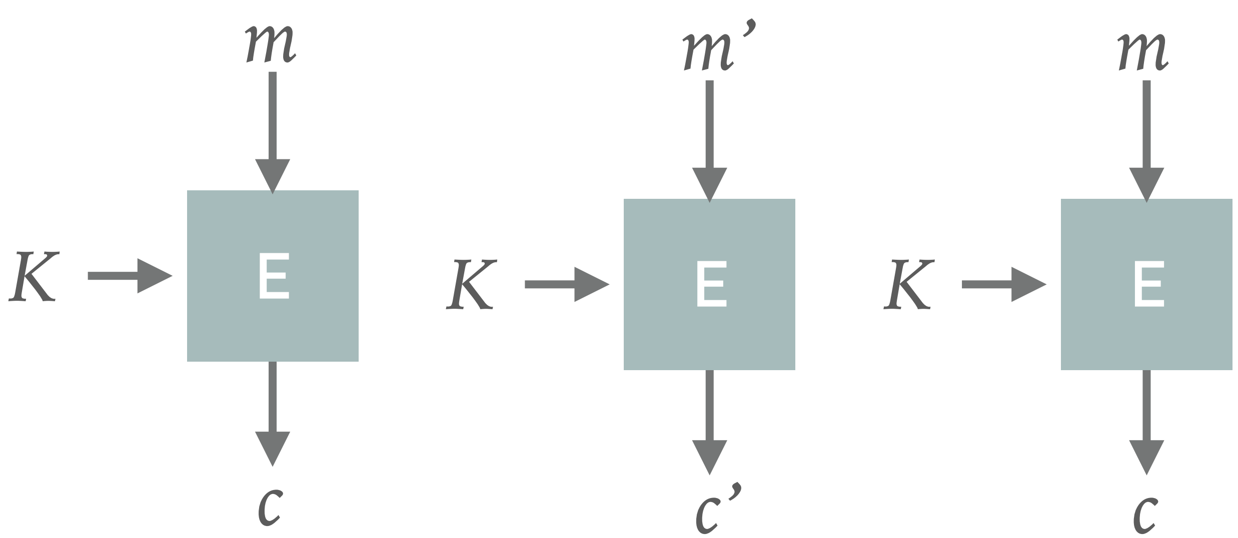
PROPERTY:
Block ciphers are deterministic
For a given m and K,
E(K,m) always returns the same c

PROPERTY: ***Small changes to the inputs cause big changes in the output***

Confusion: *Each bit of the ciphertext should depend on each bit of the key*

Diffusion: *Flipping a bit in m should flip each bit in c with Pr = 1/2*

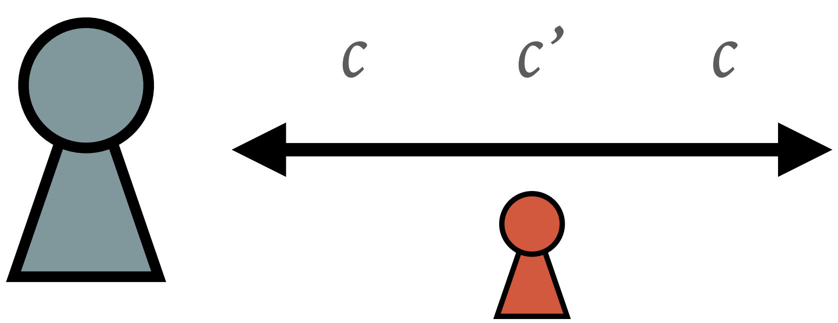
BLOCK CIPHERS ARE DETERMINISTIC



PROPERTY:

Block ciphers are deterministic

For a given m and K,
E(K,m) always returns the same c

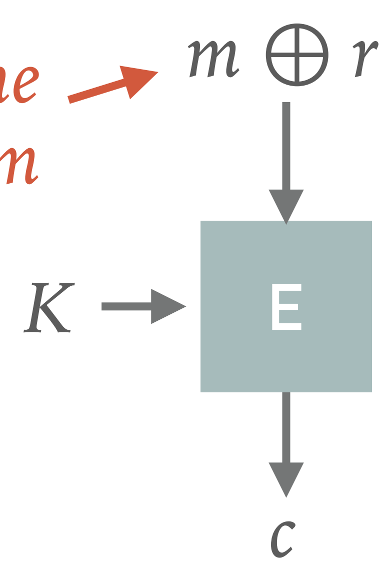


An eavesdropper could determine when messages are re-sent

A FIX:

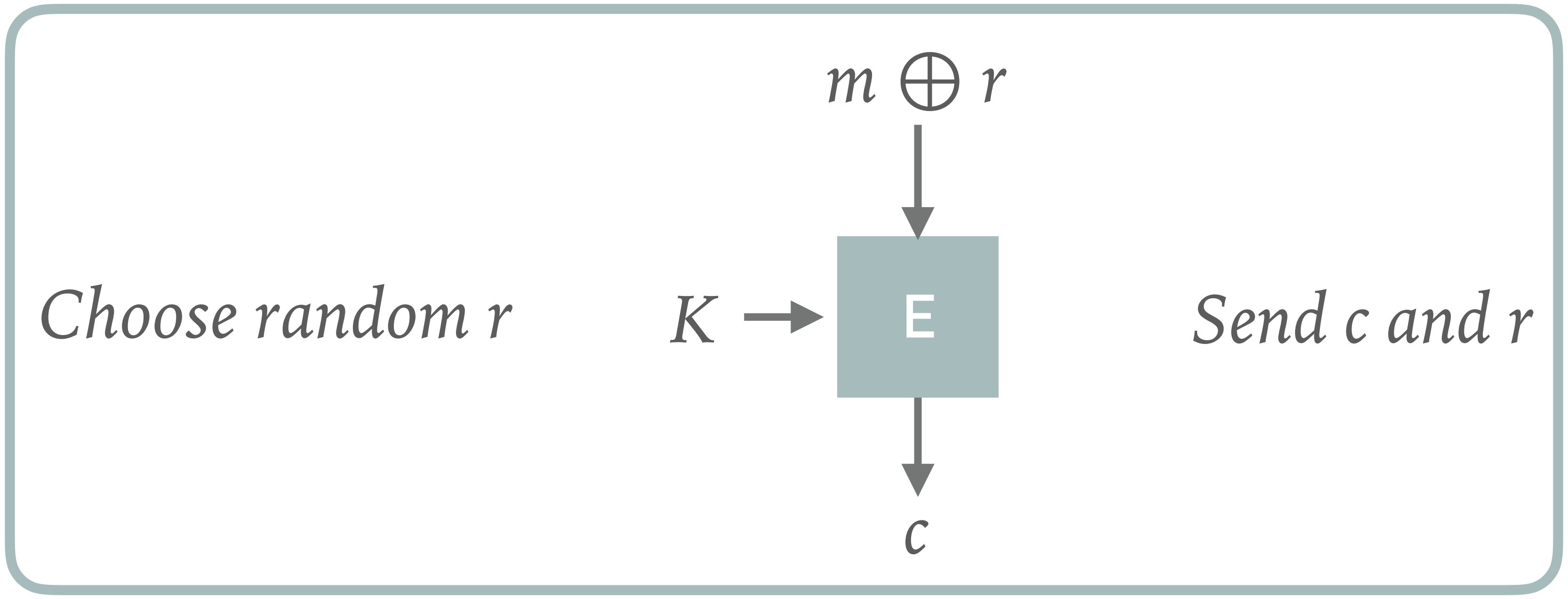
Note: $m \oplus r$ is the same size as m

Choose random r



Send c and r

INITIALIZATION VECTORS

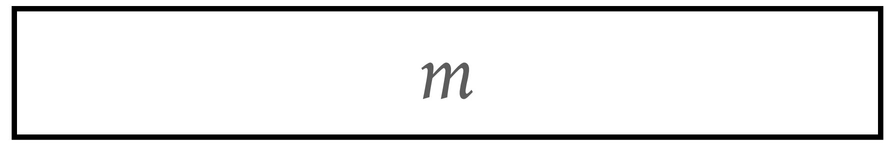


r just needs to be different each time

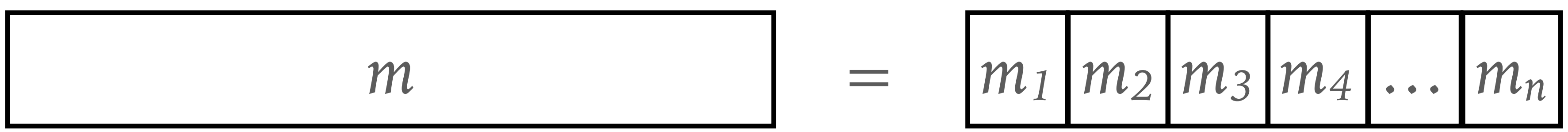
Random: Must send r with the message
This is good if messages can be reordered

Counter: Don't need to send r; the receiver can infer it from the message number
This is good if messages are delivered in-order

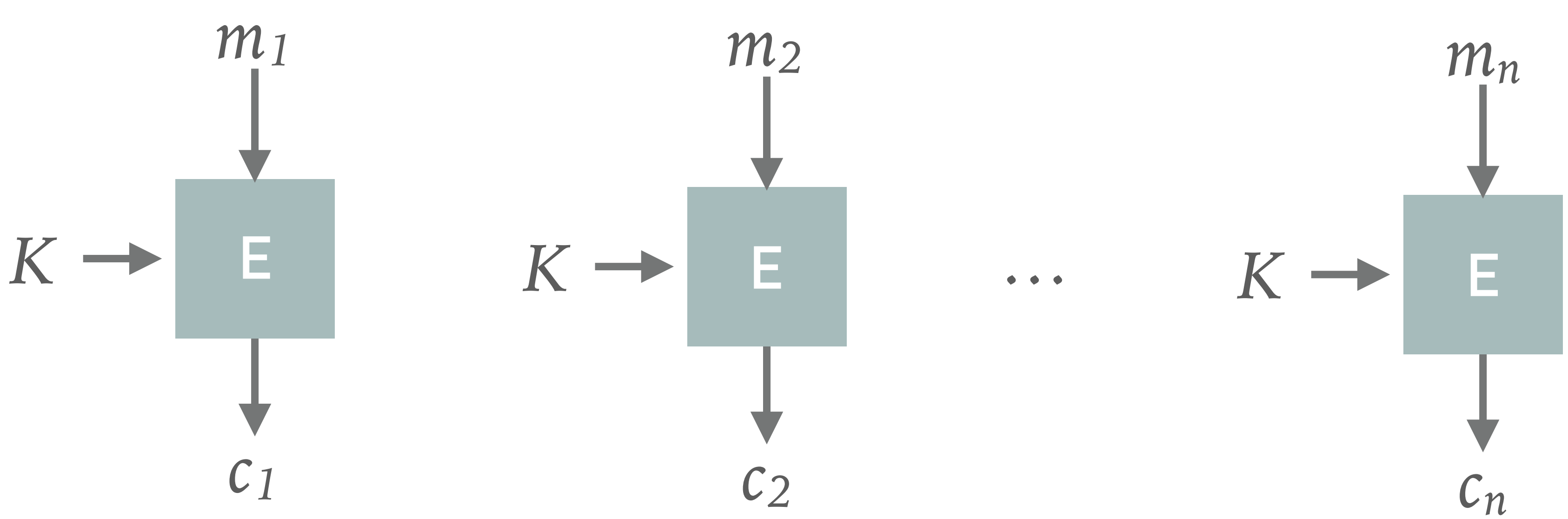
BLOCK CIPHERS HAVE FIXED SIZE



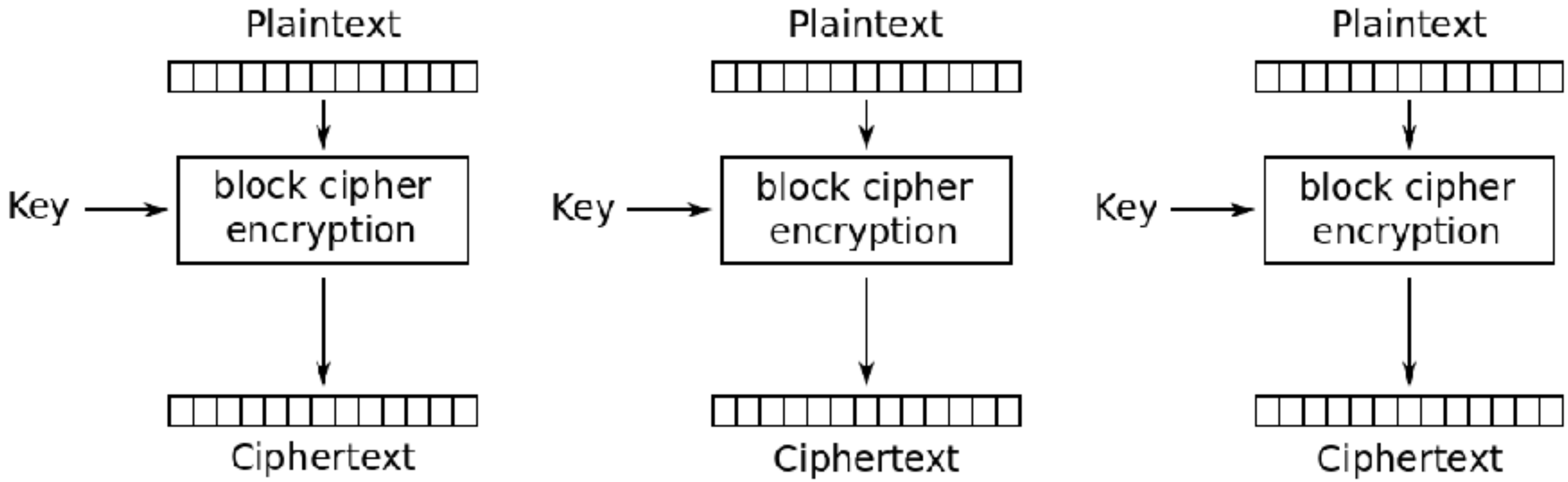
If we want to encrypt a message larger than the block size (128 bits), we simply break up the message into block-size-length pieces…



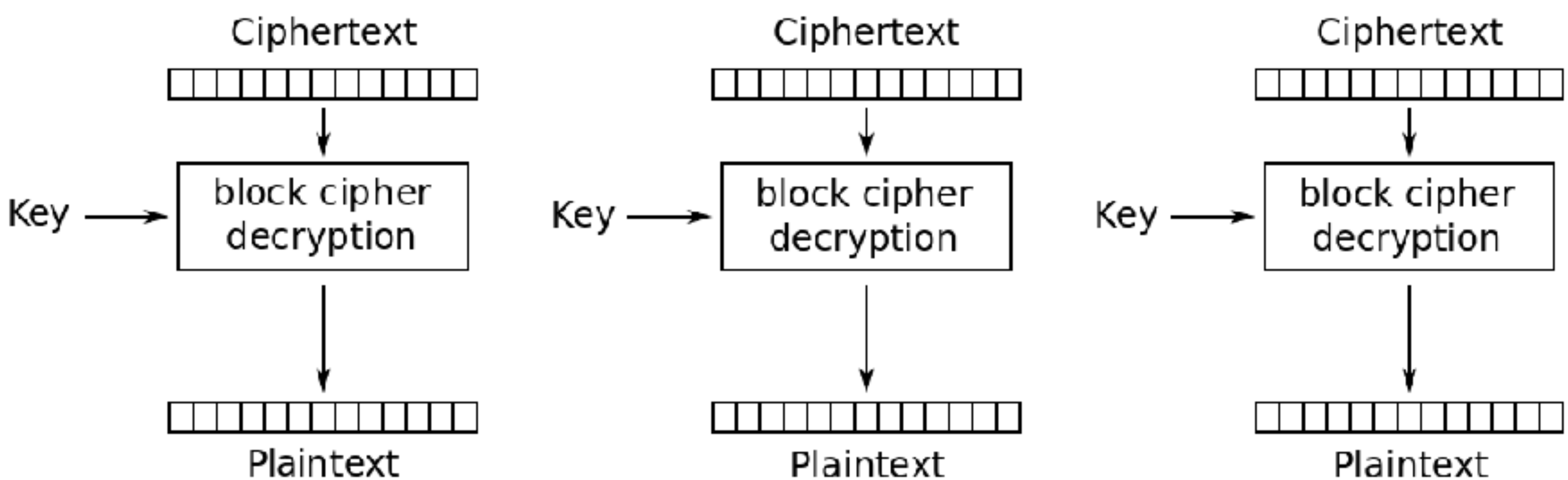
…and encrypt each block



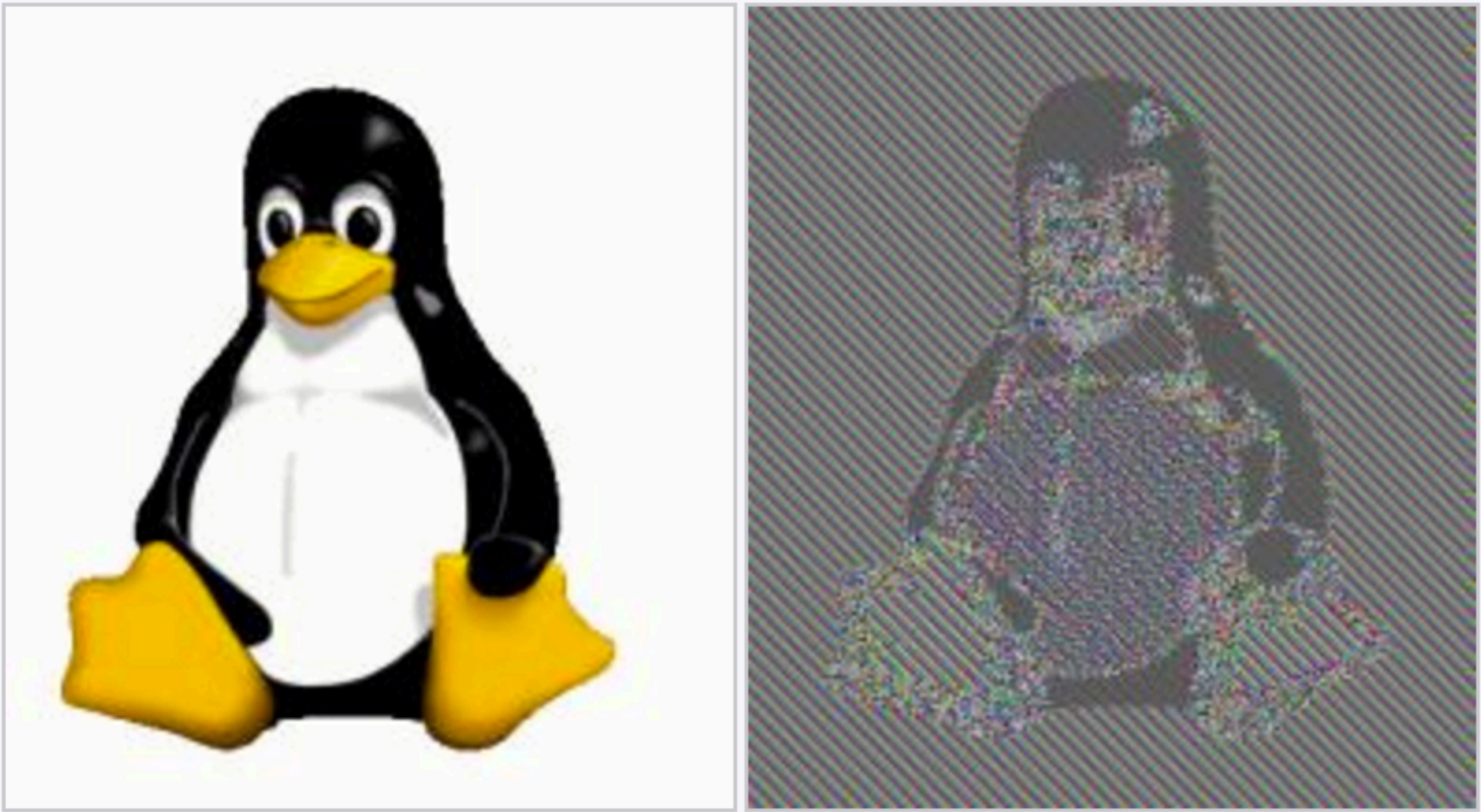
But recall: it can be deterministic. We must choose good initialization vectors. How?



Electronic Codebook (ECB) mode encryption

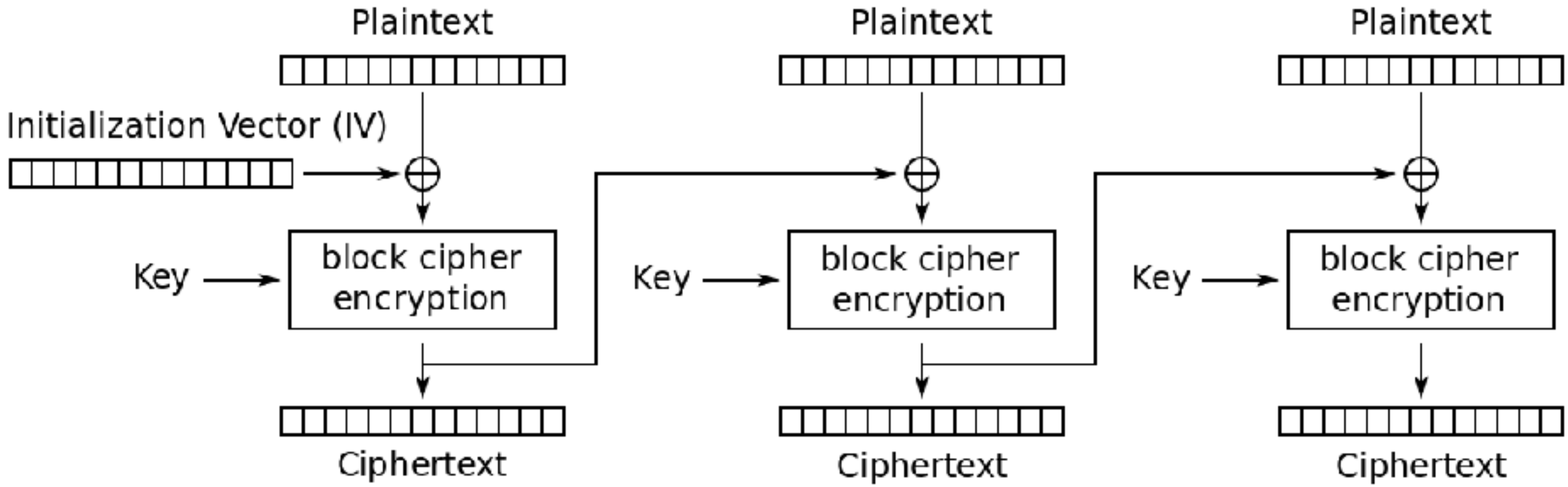


Electronic Codebook (ECB) mode decryption

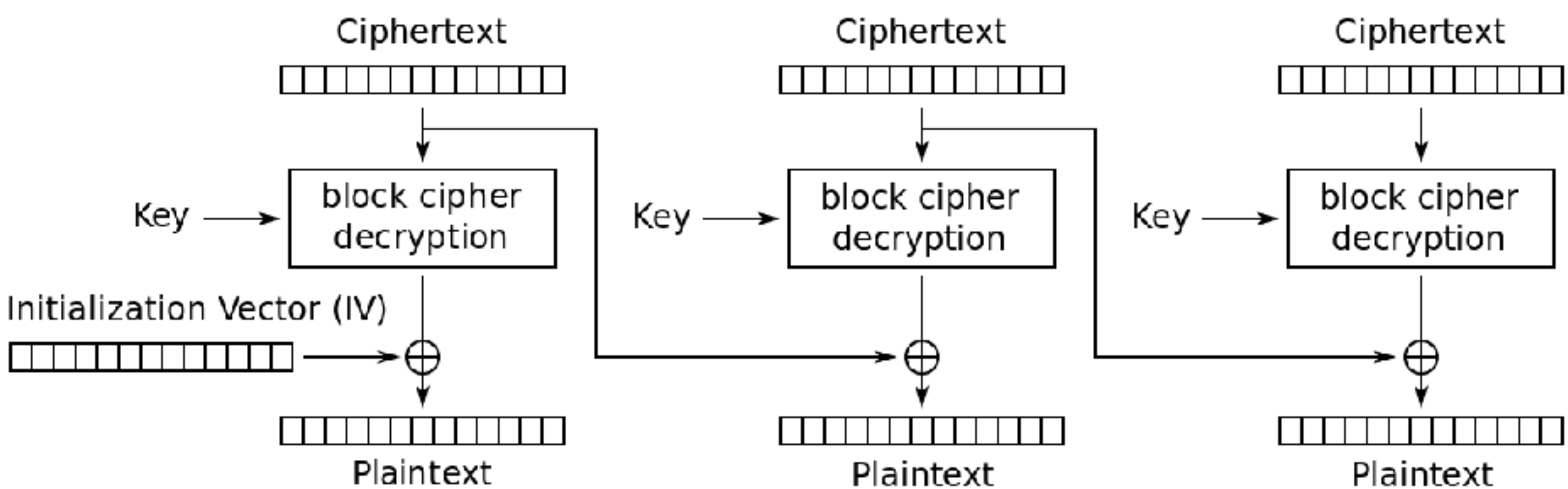


Original image   Encrypted using ECB mode

NEVER use ECB
(but over 50% of Android apps do)

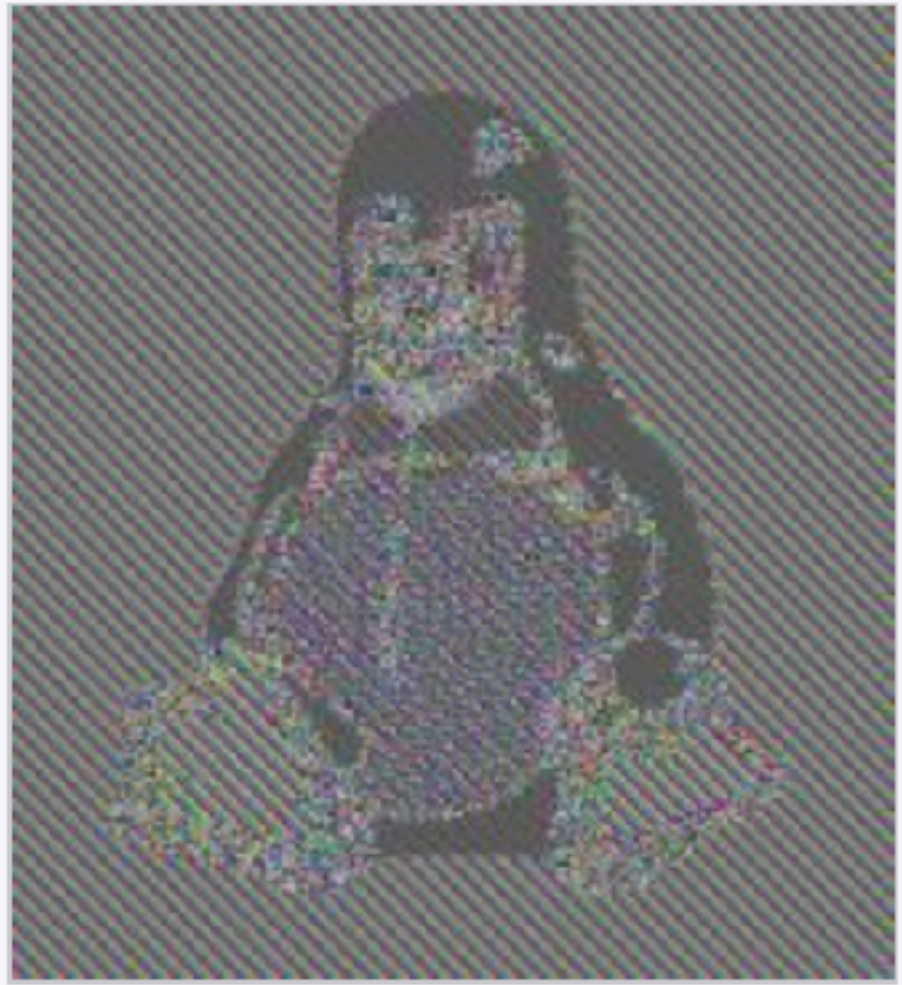


Cipher Block Chaining (CBC) mode encryption

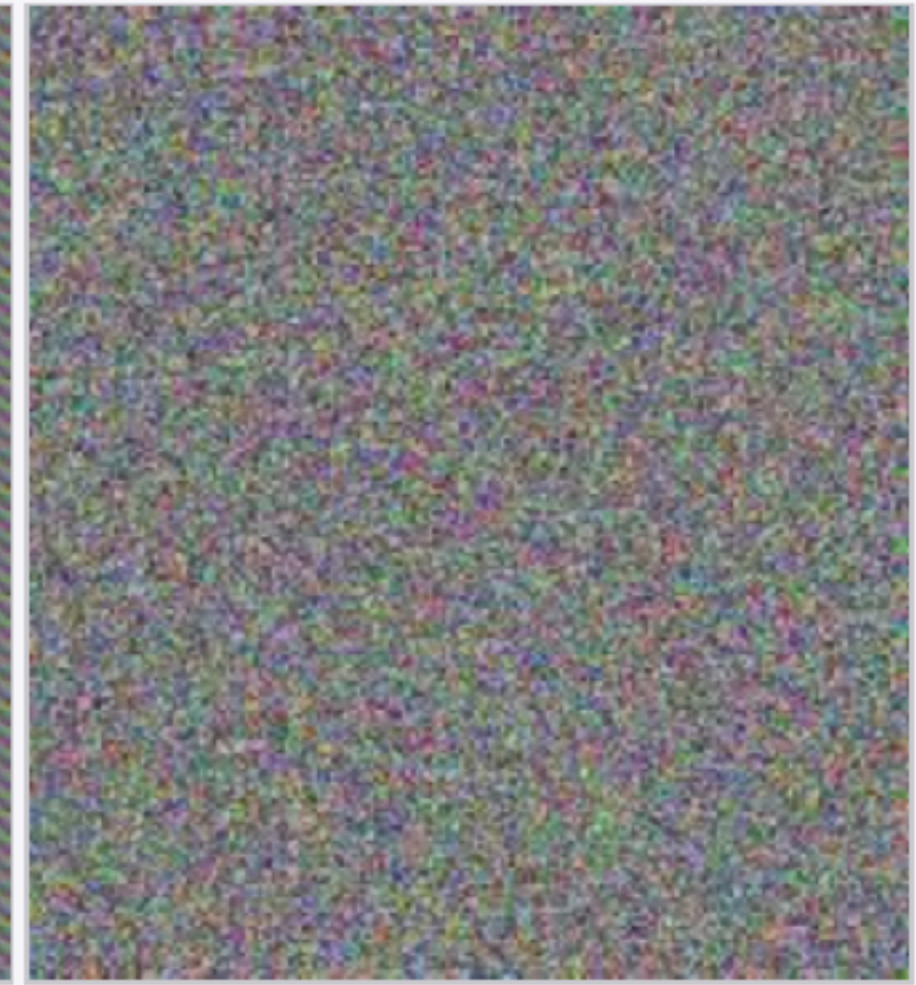


Cipher Block Chaining (CBC) mode decryption

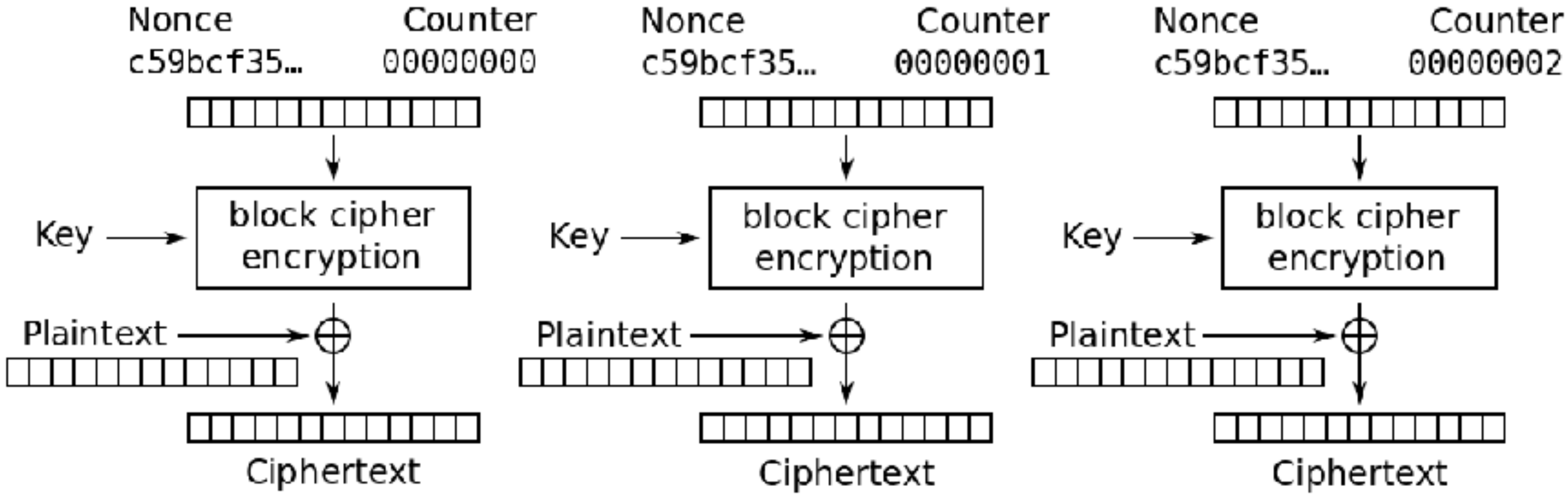
Original image



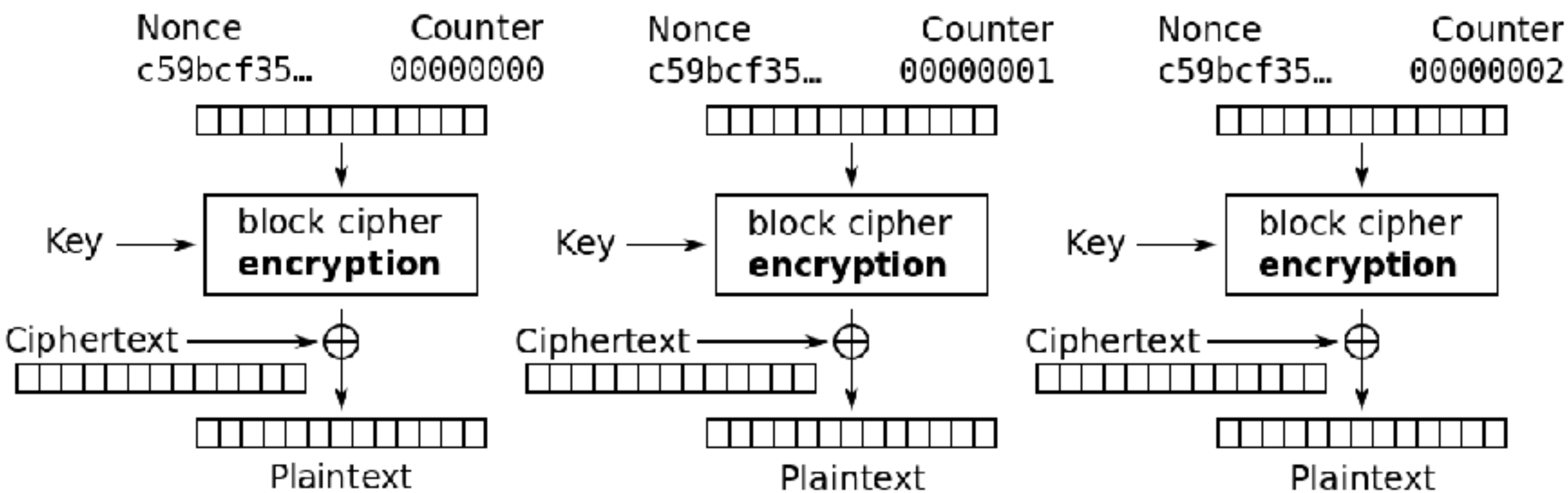
Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness



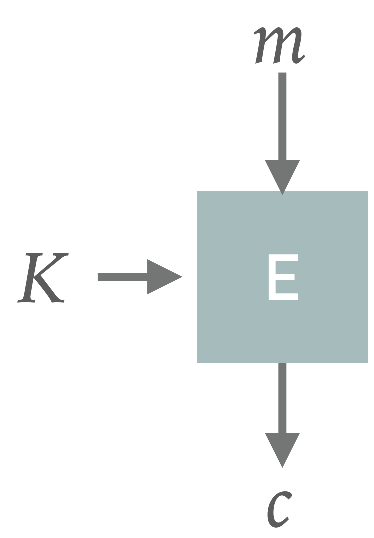
Counter (CTR) mode encryption



Counter (CTR) mode decryption

BLACKBOX #2:
MESSAGE AUTHENTICATION CODE (MAC)

MESSAGE AUTHENTICATION CODES

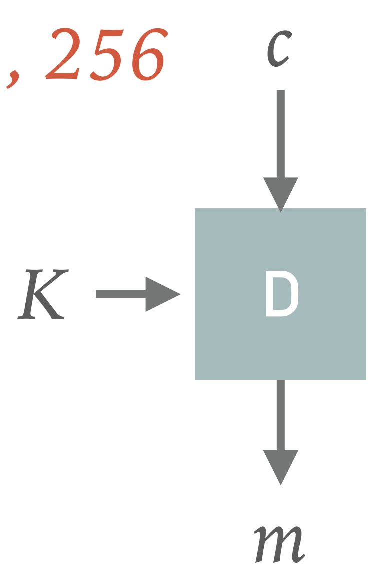


Plaintext

Same fixed ***block size***
(AES: 128 bits)

Ciphertext

AES key sizes:
128, 192, 256



Block ciphers are deterministic

For a given m and K,
E(K,m) always returns the same c

Confusion: *Each bit of the ciphertext should depend on each bit of the key*

Diffusion: *Flipping a bit in m should flip each bit in c with Pr = 1/2*

MESSAGE AUTHENTICATION CODES

- Sign: takes a key and a message and outputs a “tag”
  - Sgn(k,m) = t
- Verify: takes a key, a message, and a tag, and outputs Y/N
  - Vfy(k,m,t) = {Y,N}
- Correctness:
  - Vfy(k, m, Sgn(k, m)) = Y

ATTACKER'S GOAL: EXISTENTIAL FORGERY

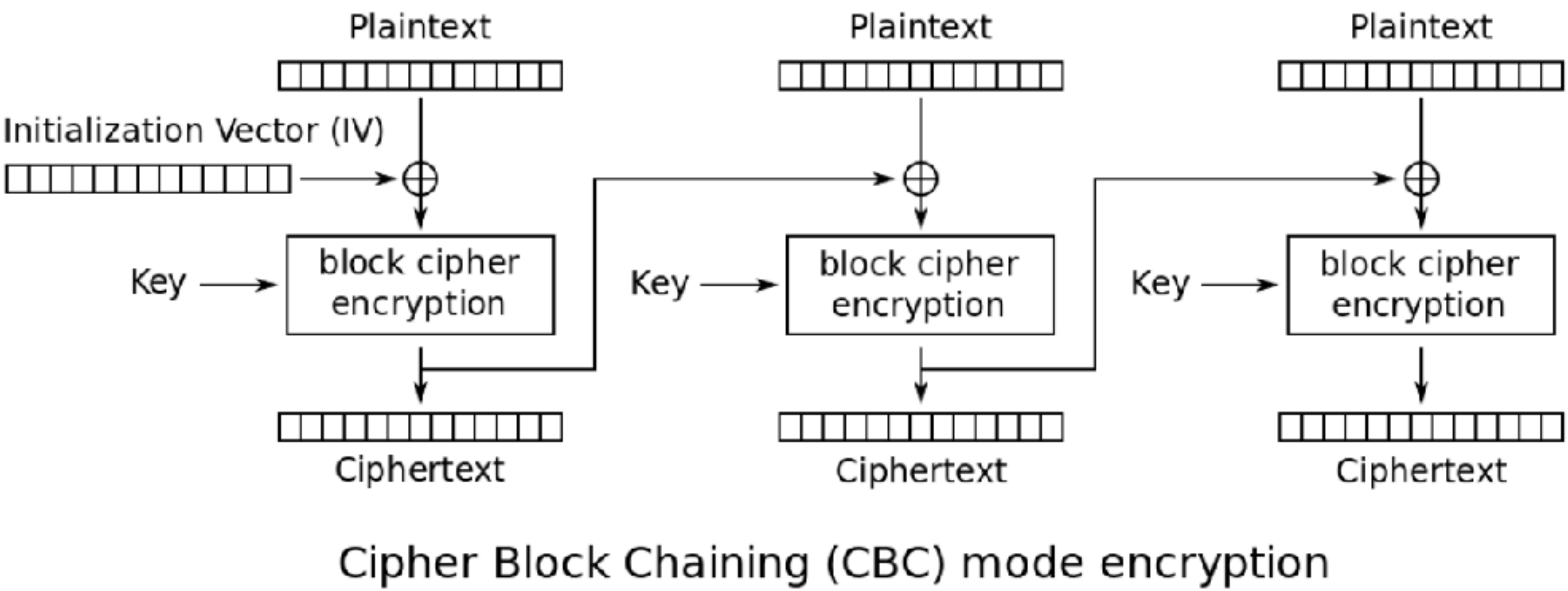
- A MAC is secure if an attacker cannot demonstrate an existential forgery despite being able to perform a chosen plaintext attack:

- Chose plaintext:
  - Attacker gets to choose m1, m2, m3, …
  - And in return gets a properly computed t1, t2, t3, …

- Existential forgery:
  - Construct a new (m,t) pair such that Vfy(k, m, t) = Y

ENCRYPTED CBC

Just take the last block in CBC *It's a trap!*



Use a separate key and encrypt the last block

BLACKBOX #3:
HASH FUNCTIONS

HASH FUNCTION PROPERTIES

- Very fast to compute
- Takes arbitrarily-sized inputs, returns fixed-sized output
- Pre-image resistant:
  Given H(m), hard to determine m
- Collision resistant
  Given m and H(m), hard to find m'≠ m s.t. H(m) = H(m')

Good hash functions: SHA family (SHA-256, SHA-512, …)

HASH MACS

- Sign(k, m):
  - opad = 0x5c5c5c…
  - ipad =0x363636…
  - $H((k \oplus opad) \| H((k \oplus ipad) \| m))$

- Verify:
  - Recompute and compare