CIS 6930:
IoT Security

Lecture 3

Prof. Kaushal Kafle

Spring 2025

Derived from slides by Adwait Nadkarni, William Enck, Micah Sherr and Patrick McDaniel

Class Notes and Clarifications

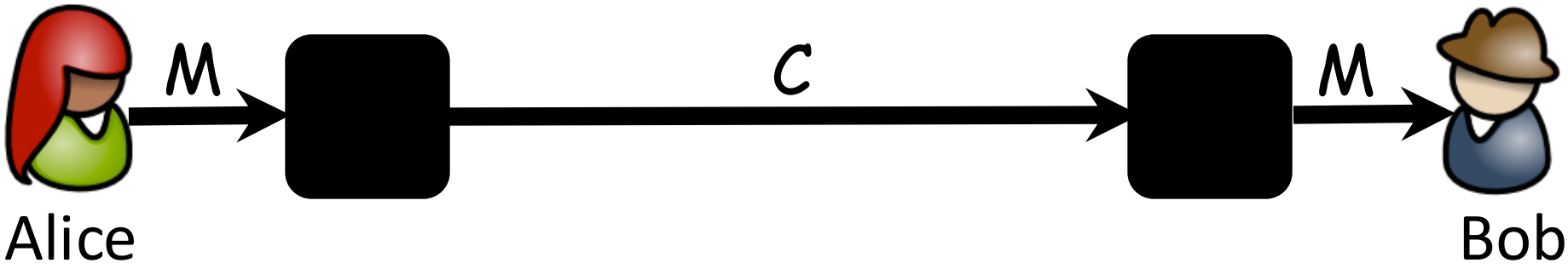
- Proposal submission is next week.
  - How many already have your team?
- Regarding the paper reviews:
  - [REVIEW] papers are the ones you should review!
  - Your verdicts should match your strengths/weaknesses and comments to authors.
    - i.e., your paper score should be *justified in your review*.
  - Strengths and weaknesses are meant as short bullet points. You clarify them in detail in the comments to authors.
  - You should look for constructive criticism, and propose ways in which the authors can improve the paper based on your criticism.

Modern Cryptography

Two flavors of confidentiality

- **Unconditional** or **probabilistic security**: cryptosystem offers provable guarantees, irrespective of computational abilities of an attacker
  - given ciphertext, the probabilities that bit i of the plaintext is 0 is p and the probability that it is 1 is (1-p)
  - e.g., one-time pad
  - often requires key sizes that are equal to size of plaintext
- **Conditional** or **computational security**: cryptosystem is secure assuming a computationally bounded adversary, or under certain hardness assumptions (e.g., P<>NP)
  - e.g., DES, 3DES, AES, RSA, DSA, ECC, DH, MD5, SHA
  - Key sizes are much smaller (~128 bits)
- Almost all deployed modern cryptosystems are conditionally secure

Recall: Encryption and Decryption



C=E(M)
M=D(C)
i.e.,
M=D(E(M))

where

M = plaintext
C = ciphertext
E(x) = encryption function
D(y) = decryption function

Kerckhoffs' Principles

- Modern cryptosystems use a key to control encryption and decryption
- Ciphertext should be undecipherable without the correct key
- Encryption key may be different from decryption key.
- **Kerckhoffs' principles** [1883]:
  - Assume Eve knows cipher algorithm
  - Security should rely on choice of key
  - If Eve discovers the key, a new key can be chosen

Kerckhoffs' Principles

- Kerckhoffs' Principles are contrary to the principle of "**security by obscurity**", which relies only upon the secrecy of the algorithm/cryptosystem
  - If security of a keyless algorithm compromised, cryptosystem becomes permanently useless (and unfixable)
  - Algorithms relatively easy to reverse engineer

Key Sizes

- Original DES used 56-bit keys, 3DES uses 168-bit keys
- AES uses 128-, 192- or 256-bit keys
- Are these numbers big enough?
  - DES has 2^{56} = 72,057,594,037,927,936 possible keys
    - In Feb 1998, distributed.net cracked DES in 41 days
    - In July 1998, the Electronic Frontier Foundation (EFF) and distributed.net cracked DES in 56 hours using a $250K machine
    - In Jan 1999, the team did in less than 24 hours
  - **Each additional bit adds 2X brute-force work factor (exponential security for linear keysize increase)**
  - There are approximately 2^{250} atoms in the universe, so don't expect 256-bit keys to be brute forced anytime in the foreseeable future *(with conventional computing)*.
- Takeaway: 128-keys are reasonably secure

$$2^{256} =$$

115,792,089,237,316,195,
423,570,985,008,687,907,
853,269,984,665,640,564,
039,457,584,007,913,129,
639,936

Cryptanalysis

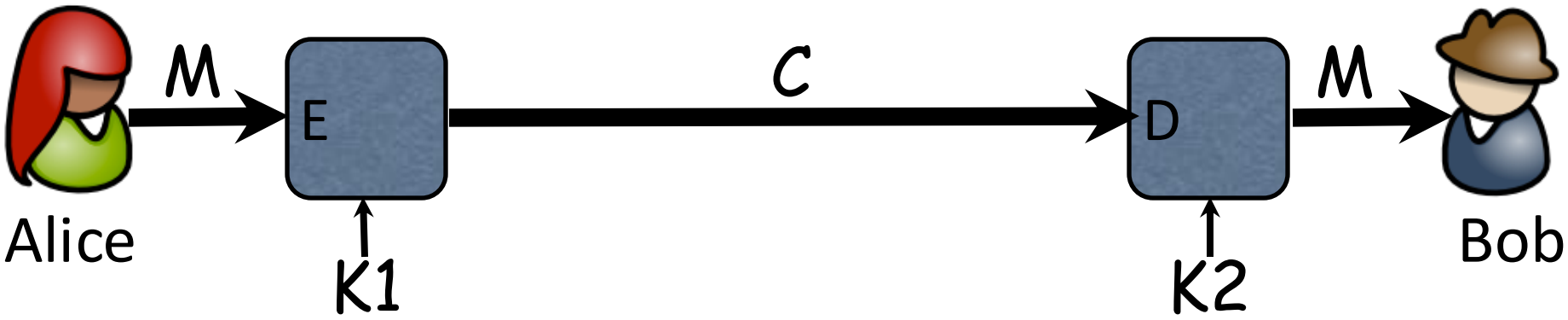
- Goal: learn the key
- Classifications:
  - **ciphertext-only** attack: Eve has access only to ciphertext
  - **known-plaintext** attack: Eve has access to plaintext and corresponding ciphertext
  - **chosen-plaintext** attack: Eve can choose plaintext and learn ciphertext
  - **chosen-ciphertext** attack: Eve can choose ciphertext and learn plaintext

Which of these are passive/active attacks?

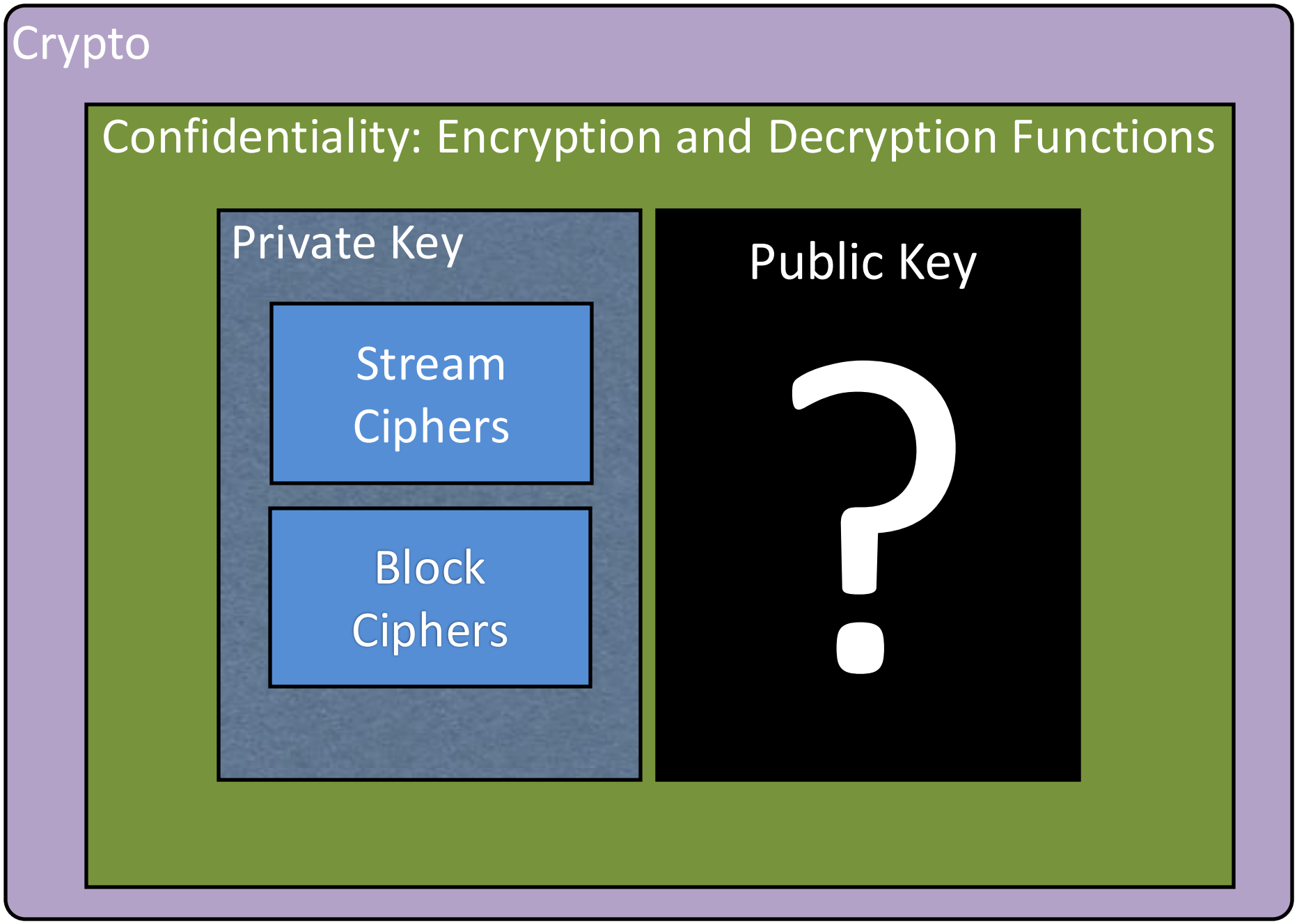
Other cryptanalysis ...

- Brute force cryptanalysis
  - Just keep trying different keys and check result
- Not covered in this class:
  - Linear cryptanalysis
    - Construct linear equations relating plaintext, ciphertext and key bits that have a high bias
    - Use these linear equations in conjunction with known plaintext-ciphertext pairs to derive key bits
  - Differential cryptanalysis
    - Study how differences in an input can affect the resultant difference at the output
    - Use chosen plaintext to uncover key bits

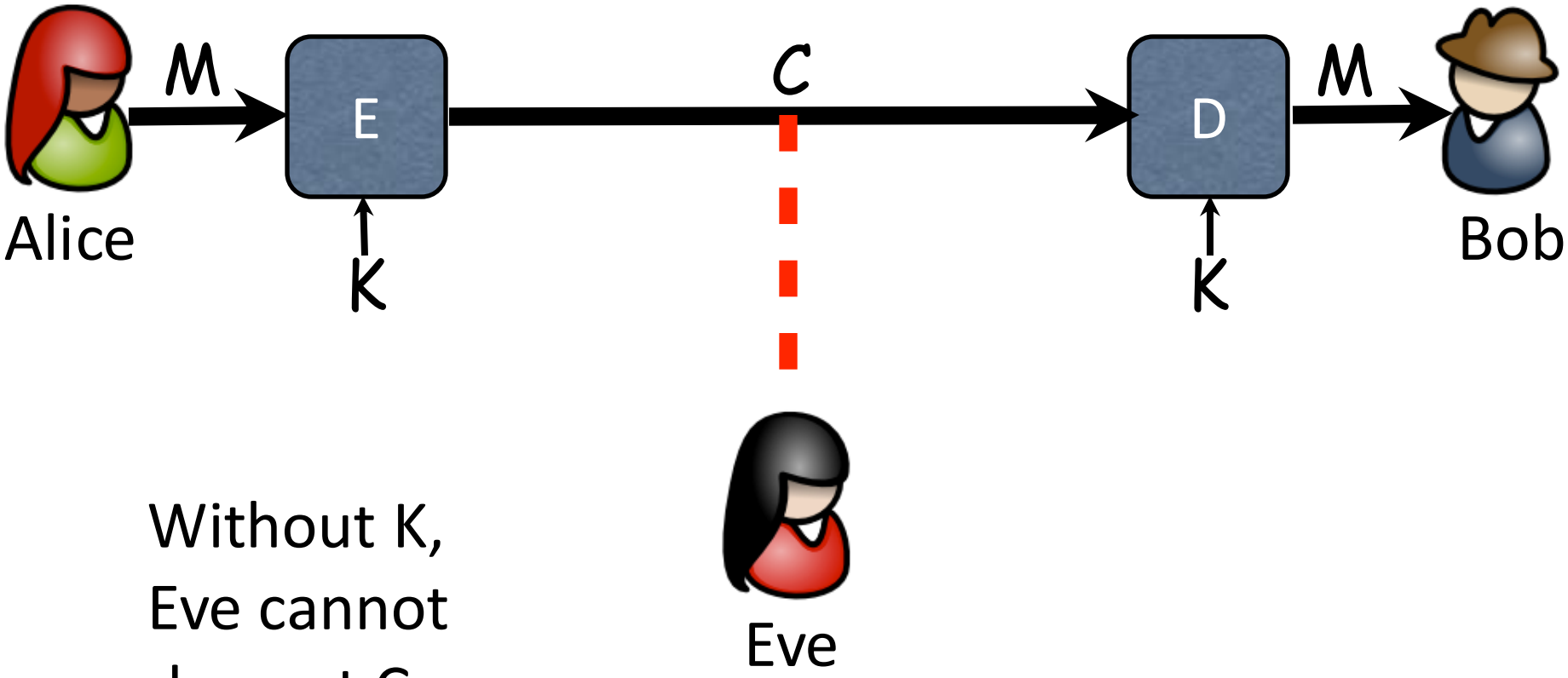
Symmetric and Asymmetric Crypto



- **Symmetric crypto:** (also called **private key crypto**)
  - Alice and Bob share the same key (K=K1=K2)
  - K used for both encrypting and decrypting
  - Doesn't imply that encrypting and decrypting are the same algorithm
  - Also called **private key** or **secret key** cryptography, since knowledge of the key reveals the plaintext
- **Asymmetric crypto:** (also called **public key crypto**)
  - Alice and Bob have different keys
  - Alice encrypts with K1 and Bob decrypts with K2
  - Also called **public key** cryptography, since Alice and Bob can publicly post their *public* keys


Crypto
Confidentiality: Encryption and Decryption Functions
Private Key
Stream Ciphers
Block Ciphers
Public Key
?

Secret Key Crypto



Without K, Eve cannot decrypt C

Block ciphers vs. Stream ciphers

- **Stream Ciphers**
  - Combine (e.g., XOR) plaintext with pseudorandom stream of bits
  - Pseudorandom stream generated based on key
  - XOR with same bit stream to recover plaintext
  - E.g., RC4, FISH
- **Block Ciphers**
  - Fixed block size
  - Encrypt block-sized portions of plaintext
  - Combine encrypted blocks (more on this later)
  - E.g., DES, 3DES, AES

Stream Ciphers

- E(M1) = M1 $\oplus$ C(K)
  - [C(K) = pseudorandom stream produced using key K]
- Useful when plaintext arrives as a stream (e.g., 802.11's WEP)
- Vulnerable if used incorrectly

Stream Ciphers

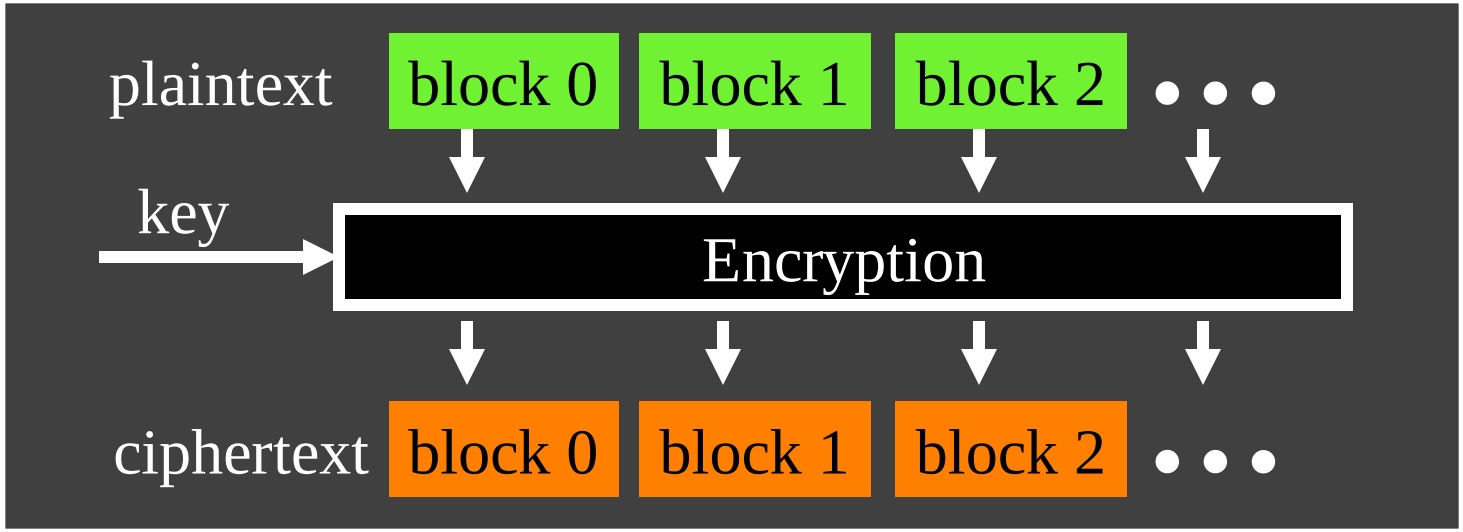
- **Key reuse:** [C(K) = pseudorandom stream produced using key K]
  - E(M1) = M1 ⊕ C(K)
  - E(M2) = M2 ⊕ C(K)
  - Suppose Eve knows ciphertexts E(M1) and E(M2)
  - E(M1) ⊕ E(M2) = M1 ⊕ C(K) ⊕ M2 ⊕ C(K) = M1 ⊕ M2
  - M1 and M2 can be derived from M1 ⊕ M2 using frequency analysis
- Countermeasure is to use IV (**initialization vector**)
  - IV sent in clear and is combined with K to produce pseudorandom sequence
  - E.g., replace C(K) with C(K⊕IV)
  - IVs should never be reused and should be sufficiently large
  - WEP broken partly because IVs were insufficiently large
  - modern stream ciphers take IVs, but it's up to the programmer to generate them

Stream Ciphers

- **Substitution Attack:**
  - M = “Pay me $100.00”
  - E(M) = M ⊕ C(K)
  - Suppose Eve knows M and E(M) but doesn't know K
  - She can substitute M for M' by replacing E(M) with:
    - E'(M) = **E(M) ⊕ M ⊕ M'** = M ⊕ C(K) ⊕ M ⊕ M' = C(K) ⊕ M'
    - Eve can then replace E(M) with E'(M), which Bob will decrypt message as M' (“Pay me $900.00”)
  - *Encryption alone does not provide integrity:* Countermeasure is to include message authentication code (more on this later) that helps detect manipulation (i.e., provides integrity and authenticity)

Block ciphers: Generic Block Encryption

- Converts one input plaintext block of fixed size b bits to an output ciphertext block also of *b* bits
- Benefits of large *b*? of short *b*?
  - Think cryptanalysis!
- Block and key size are *separate parameters*
- E.g., AES, DES

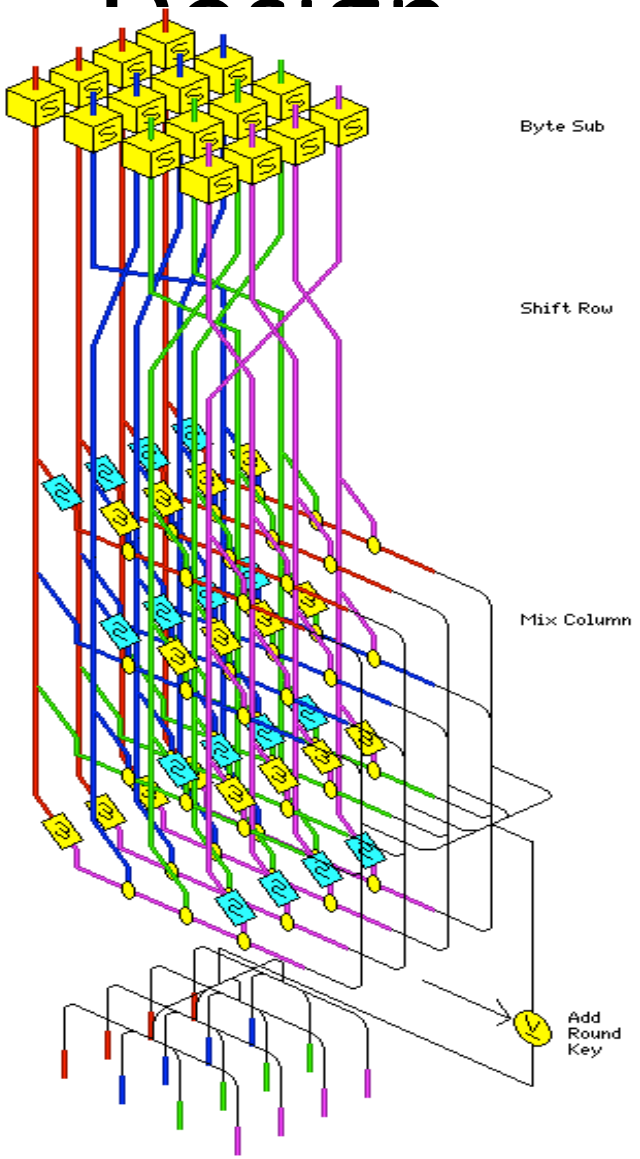


Two Principles for Cipher Design

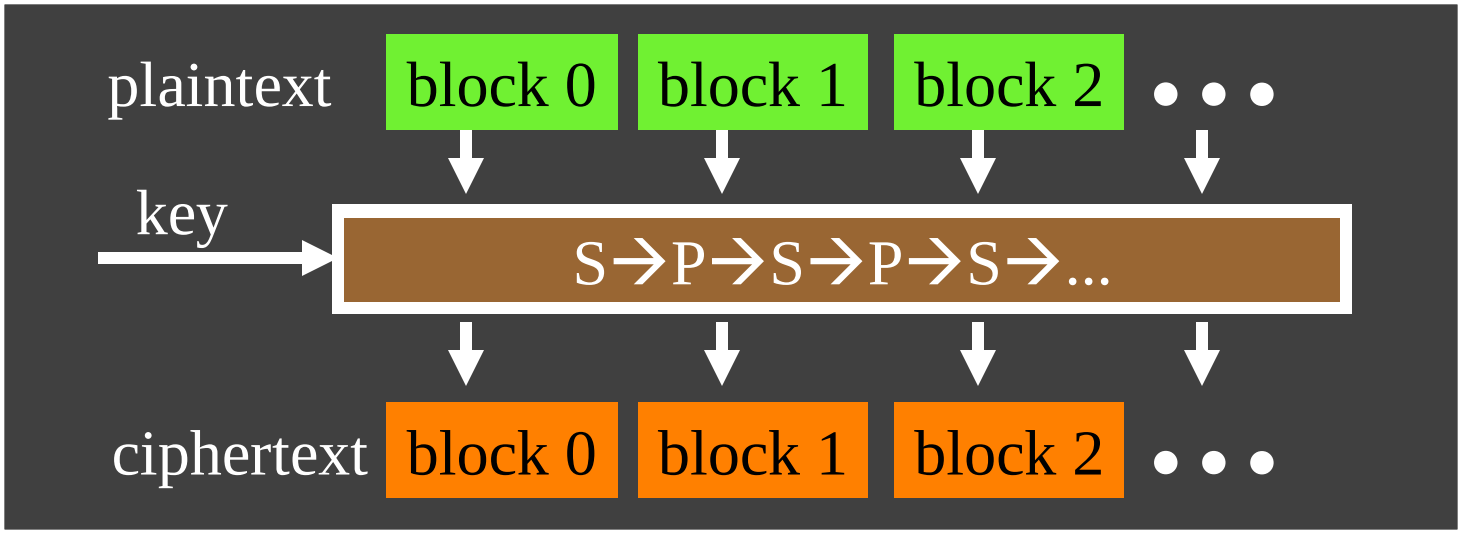
- *Confusion*: Make the relationship between the <plaintext, key> input and the <ciphertext> output as complex (non-linear) as possible
  - Mainly accomplished by *substitution*
- *Diffusion*: Spread the influence of each input bit across many output bits
  - Mainly accomplished by *permutation*
- Idea: use *multiple*, *alternating* permutations and subsitutions
  - *S→P→S→P→S→... or P→S→P→S→P→...*
  - Does it have to alternate?, e.g., S→S→S→P→P→P→S→S→...

-> Stream ciphers do not have diffusion! Why?

Two Principles for Cipher Design



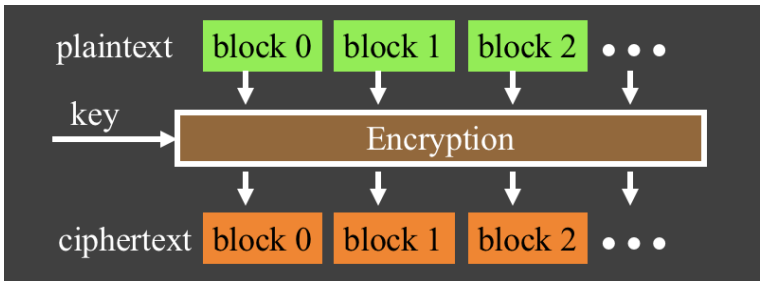
Two Principles for Cipher Design



- Can I *predictably* change the plaintext, by changing the ciphertext?
  - No. The relationship is too complex.

Modes of Operation

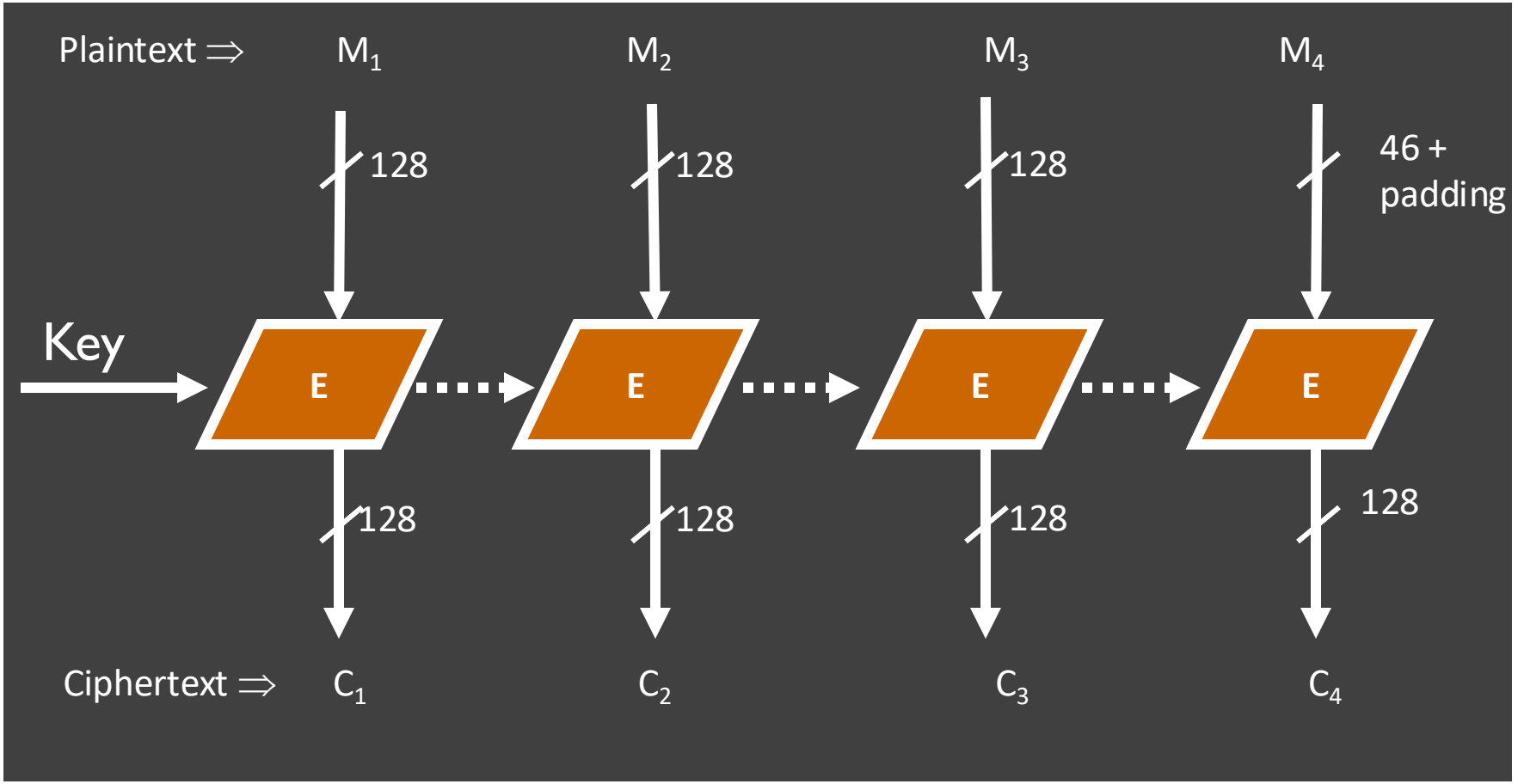
- Most ciphers work on blocks of fixed (small) size
- How to encrypt long messages?
- Modes of operation
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
  - CTR (Counter)
  - (there are many more; we will look at 3 for a bare minimum understanding)



Issues for Block Chaining Modes

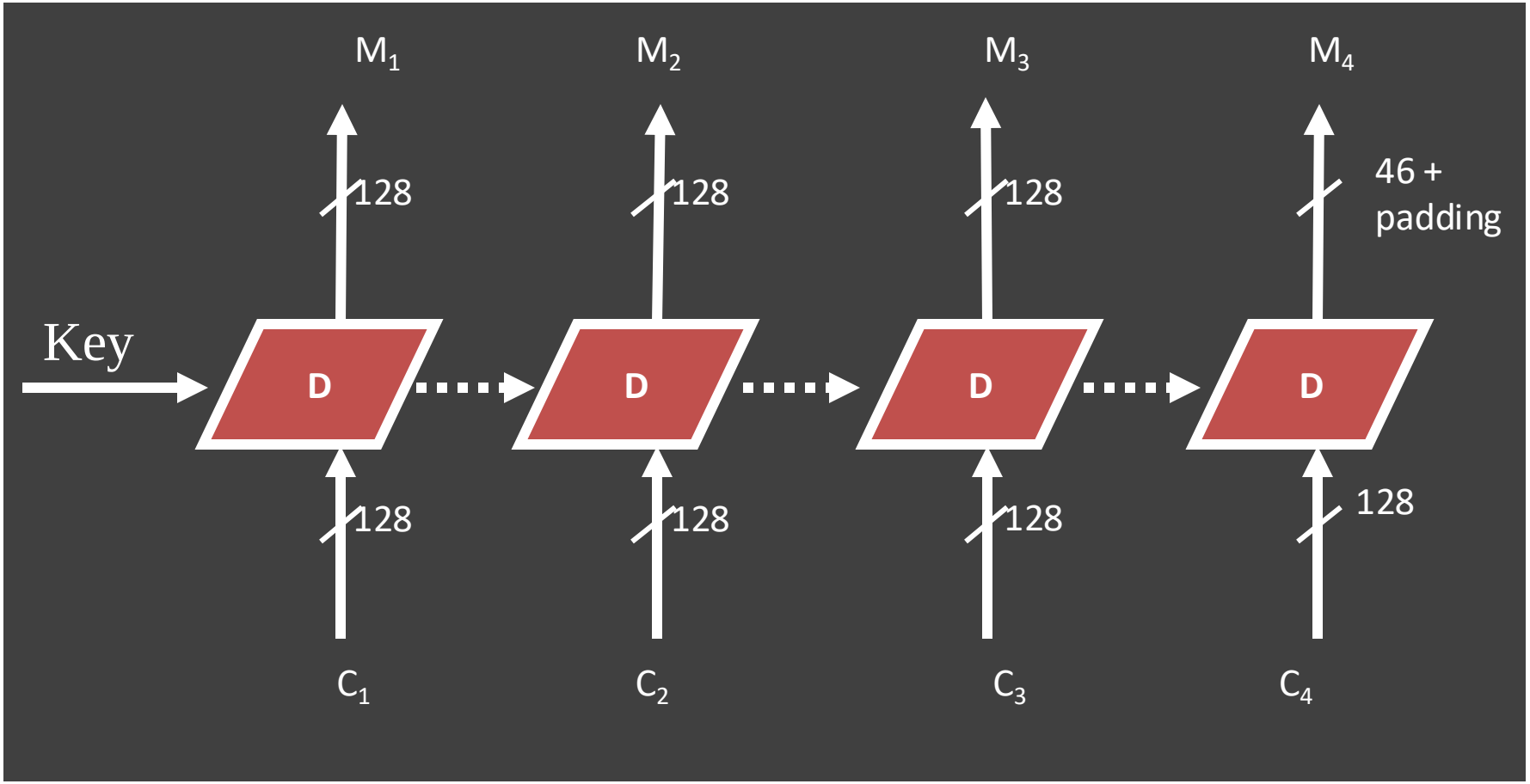
- *Information leakage*: Does it reveal info about the plaintext blocks?
- *Ciphertext manipulation*: Can an attacker modify ciphertext block(s) in a way that will produce a predictable/desired change in the decrypted plaintext block(s)?
  - Note: assume the structure of the plaintext is known, e.g., first block is employee #1 salary, second block is employee #2 salary, etc.
- *Parallel/Sequential*: Can blocks of plaintext (ciphertext) be encrypted (decrypted) in parallel?
- *Error Propagation*: If there is an error in a plaintext (ciphertext) block, will there be an encryption (decryption) error in more than one ciphertext (plaintext) block?

Electronic Code Book (ECB)



- The easiest mode of operation; each block is independently encrypted

ECB Decryption

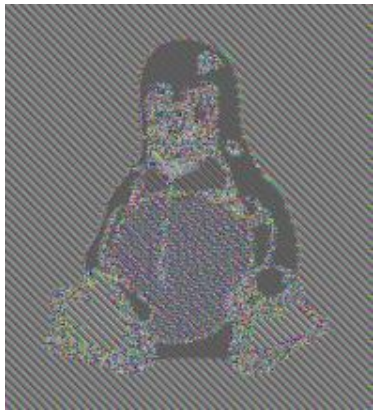


- Each block is independently decrypted

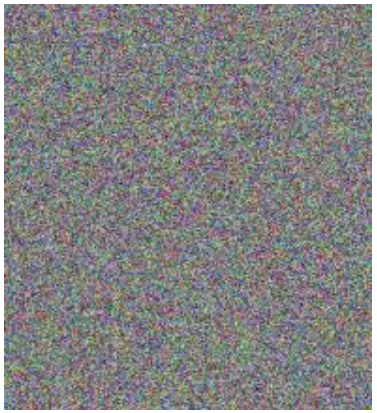
ECB Issues

- *Information leaks*: two ciphertext blocks that are the same
- *Manipulation*: switch ciphertext with predictable results on plaintext (e.g., shuffle).
- *Parallel*: yes
- *Error Propagate*: no

Plaintext



ECB



Other modes

4:13 Ode to ECB

by Ben Nagy

Oh little one, you're growing up
You'll soon be writing C
You'll treat your ints as pointers
You'll nest the ternary
You'll cut and paste from github
And try cryptography
But even in your darkest hour
Do not use ECB

CBC's BEASTly when padding's abused
And CTR's fine til a nonce is reused
Some say it's a CRIME to compress then encrypt
Or store keys in the browser (or use javascript)
Diffie Hellman will collapse if hackers choose your g
And RSA is full of traps when e is set to 3
Whiten! Blind! In constant time! Don't write an RNG!
But failing all, and listen well: Do not use ECB

They'll say "It's like a one-time-pad!
The data's short, it's not so bad
the keys are long—they're iron clad
I have a PhD!"
And then you're front page Hacker News
Your passwords cracked—Adobe Blues.
Don't leave your penguin showing through,
Do not use ECB

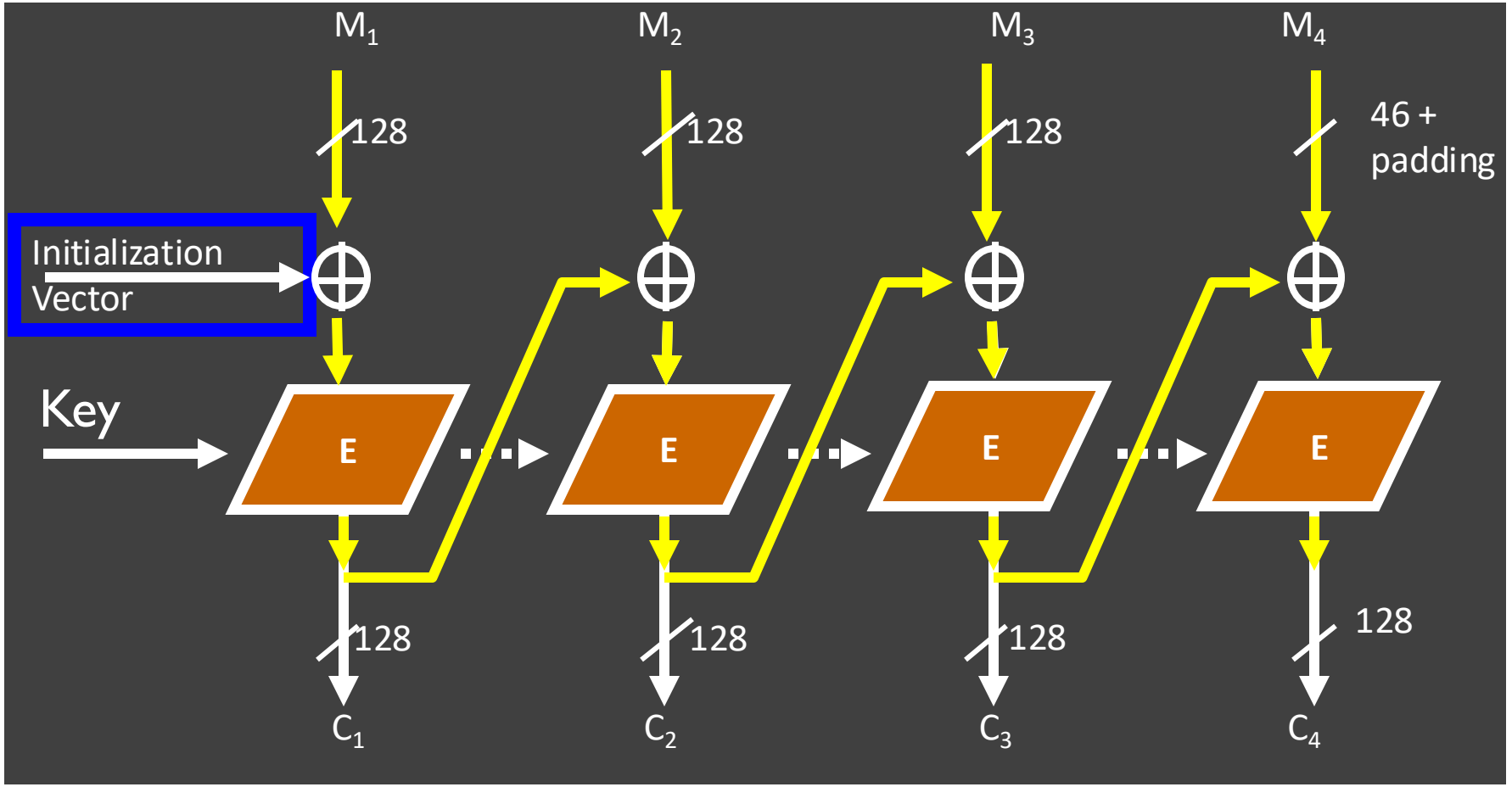
Sometimes it can seem like there's ECB everywhere. ECB on TV, ECB in music, it's endless. But that doesn't make it safe. Or right. So tune out and avoid ECB, no matter what your friends, the TV, or your favourite cryptographer tells you.

Canadian Joke Council

True Bugs Wait ♡

@natashenka
#truebugswait

Cipher Block Chaining (CBC)

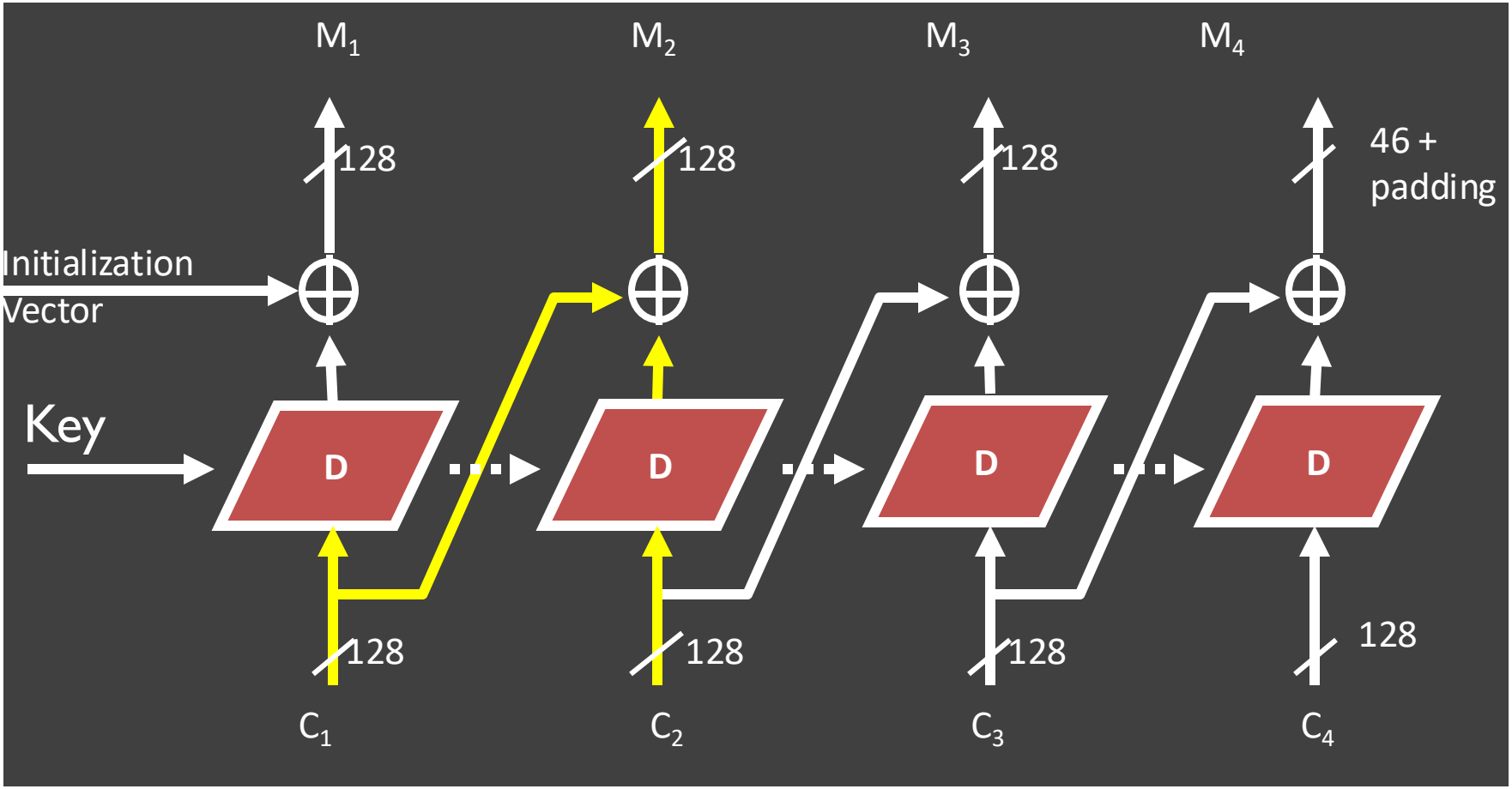


- Chaining dependency: each ciphertext block depends on all preceding plaintext blocks

Initialization Vectors

- Initialization Vector (IV)
  - Used along with the key; not secret
  - For a given plaintext, changing either the key, or the IV, will produce a different ciphertext
  - Why is that useful?
- IV generation and sharing
  - Random; may transmit with the ciphertext
  - Incremental; predictable by receivers

CBC Decryption

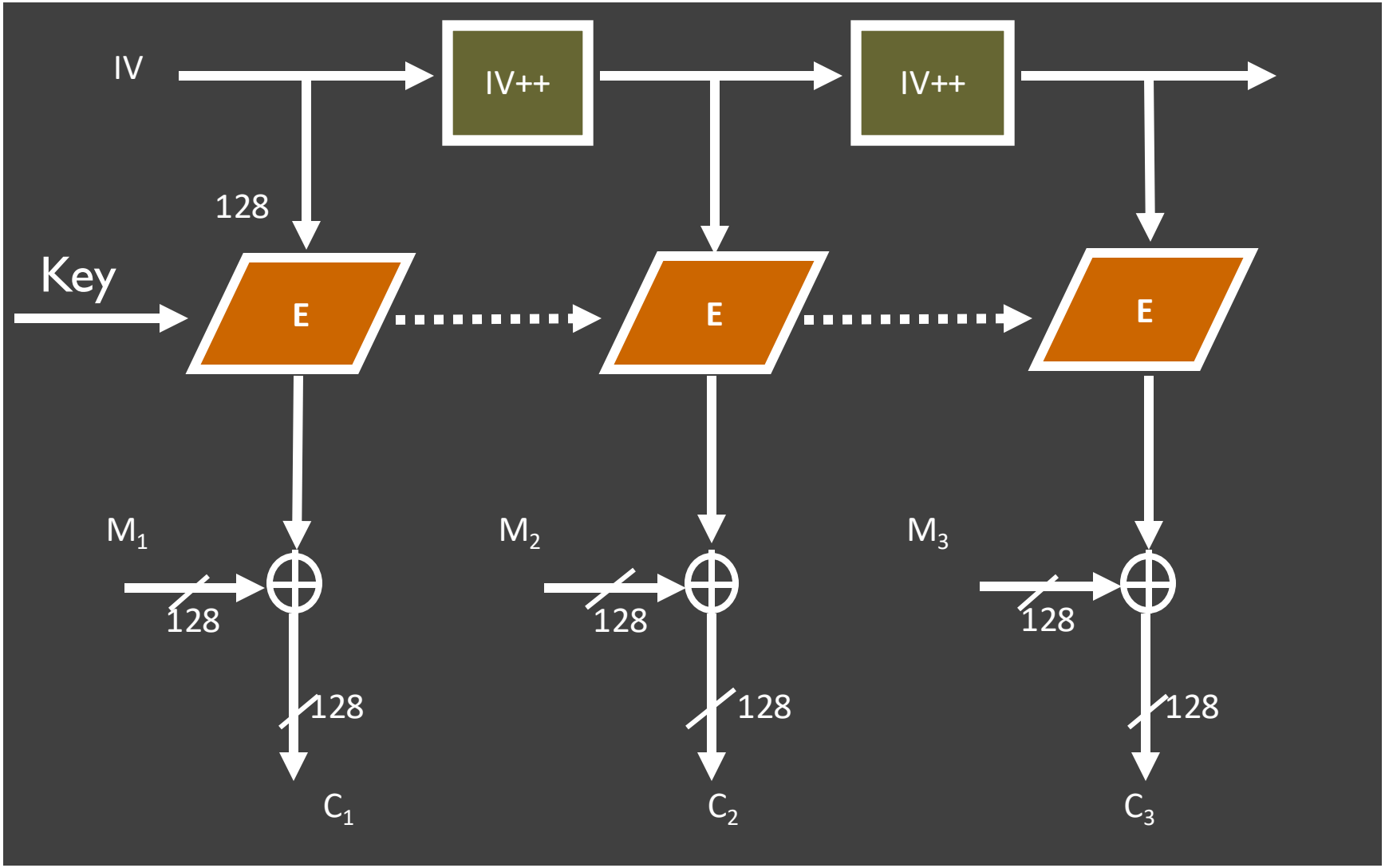


- How many ciphertext blocks does each plaintext block depend on?

CBC Properties

- Does information leak?
  - Identical plaintext blocks will produce different ciphertext blocks
- Can ciphertext be manipulated profitably?
  - Yes
- Parallel processing possible?
  - no (encryption), yes (decryption)
- Do ciphertext errors propagate?
  - yes (encryption), a little (decryption)

Counter Mode (CTR)



CTR Mode Properties

- Does information leak?
  - Identical plaintext block produce different ciphertext blocks
- Can ciphertext be manipulated profitably
  - Yes!
- Parallel processing possible
  - Yes (both generating pad and XORing)
- Do ciphertext errors propagate?
  - No.
- Allow decryption the ciphertext at any location
  - Ideal for random access to ciphertext

What encryption does and does not

- Does:
  - confidentiality
- Doesn't do:
  - data integrity
  - source authentication
- **Need:** ensure that data is not altered and is from an authenticated source