WILLIAM & MARY
CHARTERED 1693

CSCI 445: Mobile Application Security

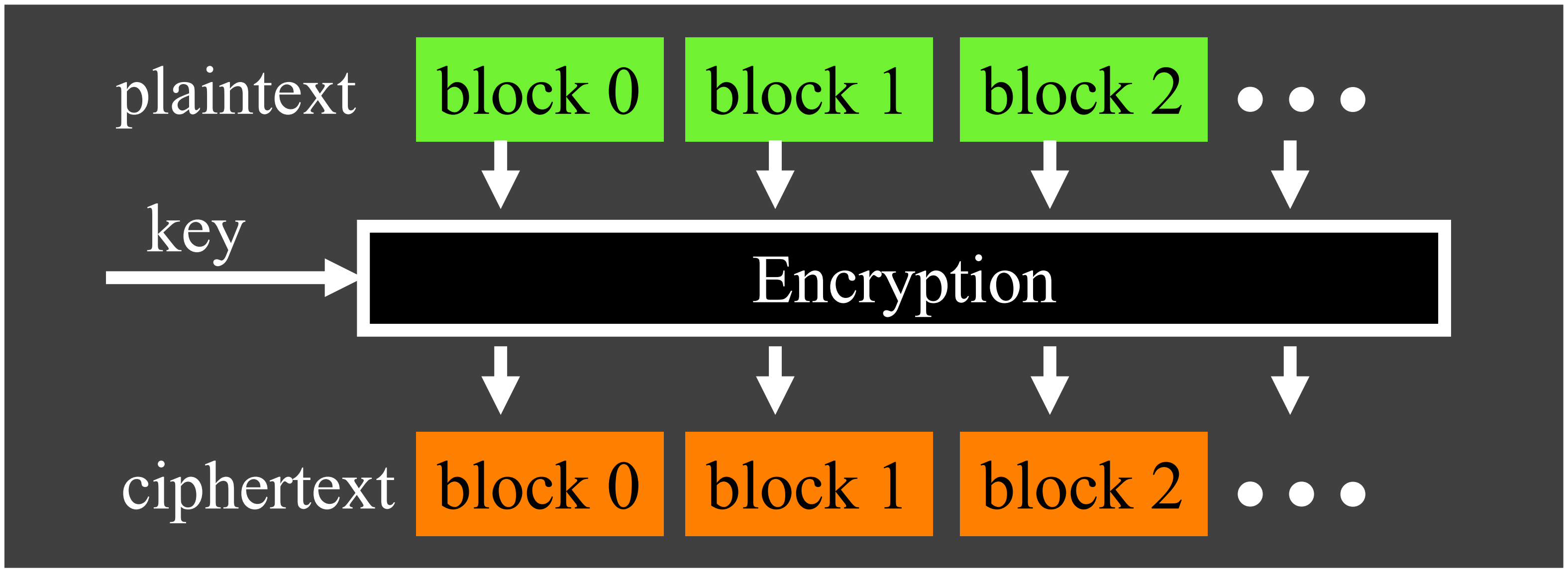
Lecture 4

Prof. Adwait Nadkarni

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

Block ciphers: Generic Block Encryption

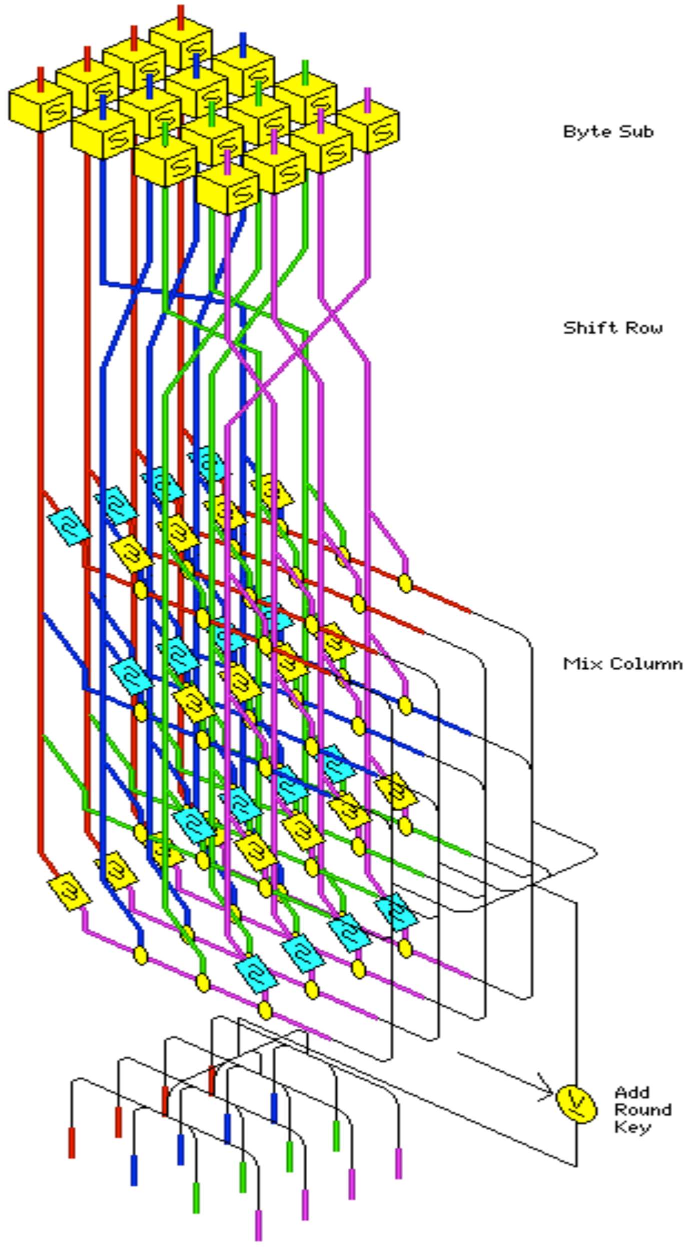
- Converts one input plaintext block of fixed size b bits to an output ciphertext block also of *b* bits
- Benefits of large *b*? of short *b*?
- Block and key size are *separate parameters*
- E.g., AES, DES, DESX *(won't go into much detail for this class)*



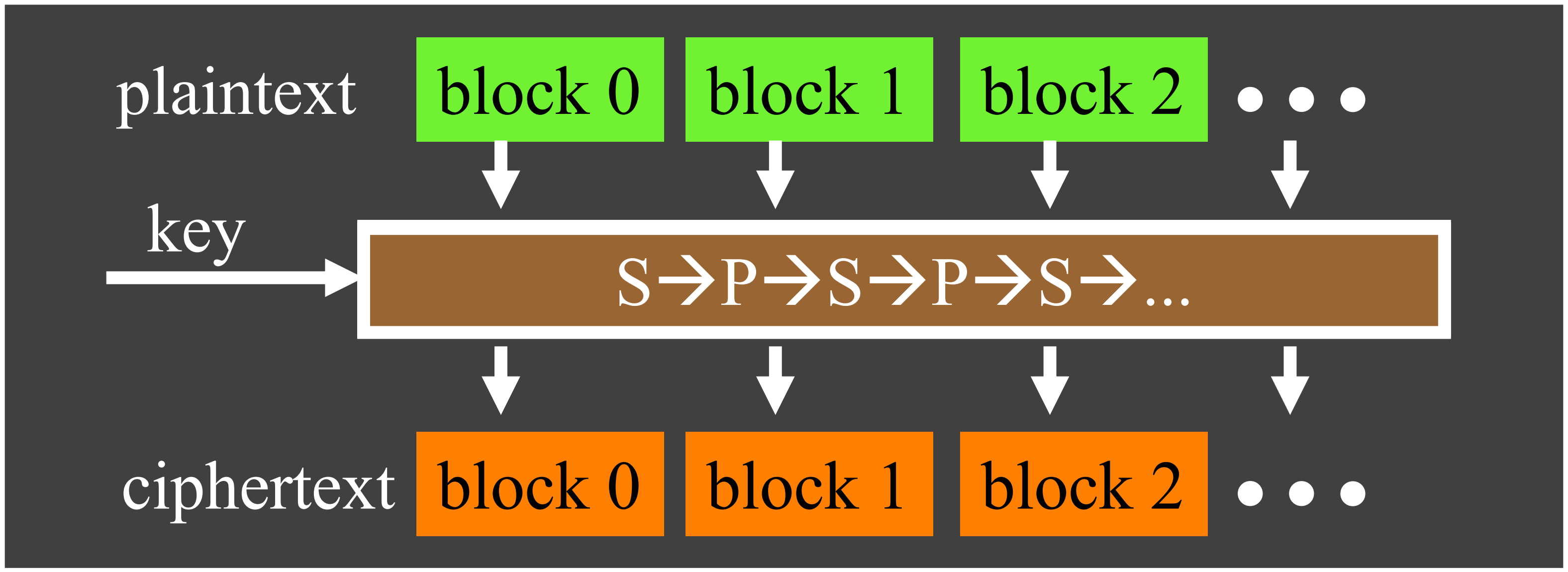
Two Principles for Cipher Design

- *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→…

Two Principles for Cipher Design



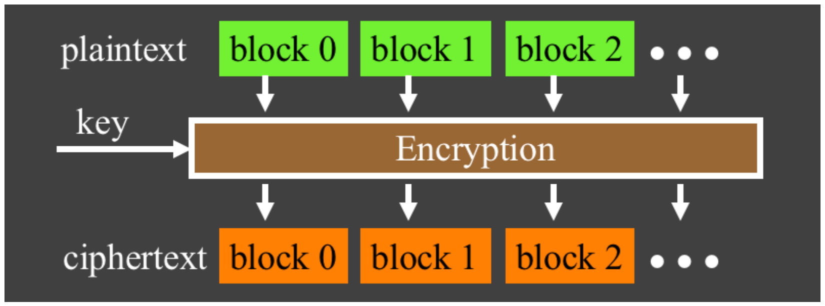
Two Principles for Cipher Design



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

Modes of Operation

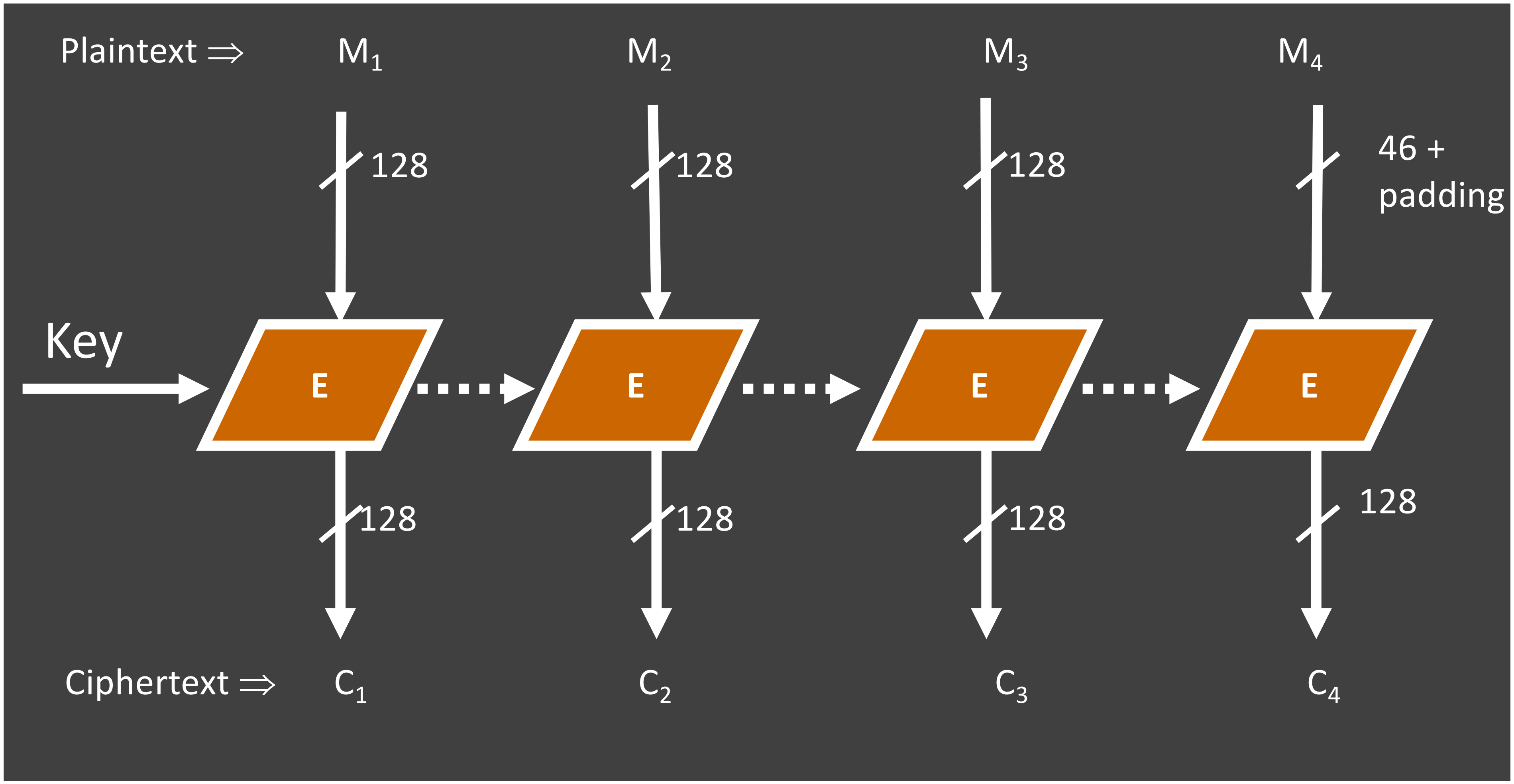
- Most ciphers work on blocks of fixed (small) size
- How to encrypt long messages?
- Modes of operation
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
  - CTR (Counter)
  - (there are many more; we will look at 3 for a bare minimum understanding)



Issues for Block Chaining Modes

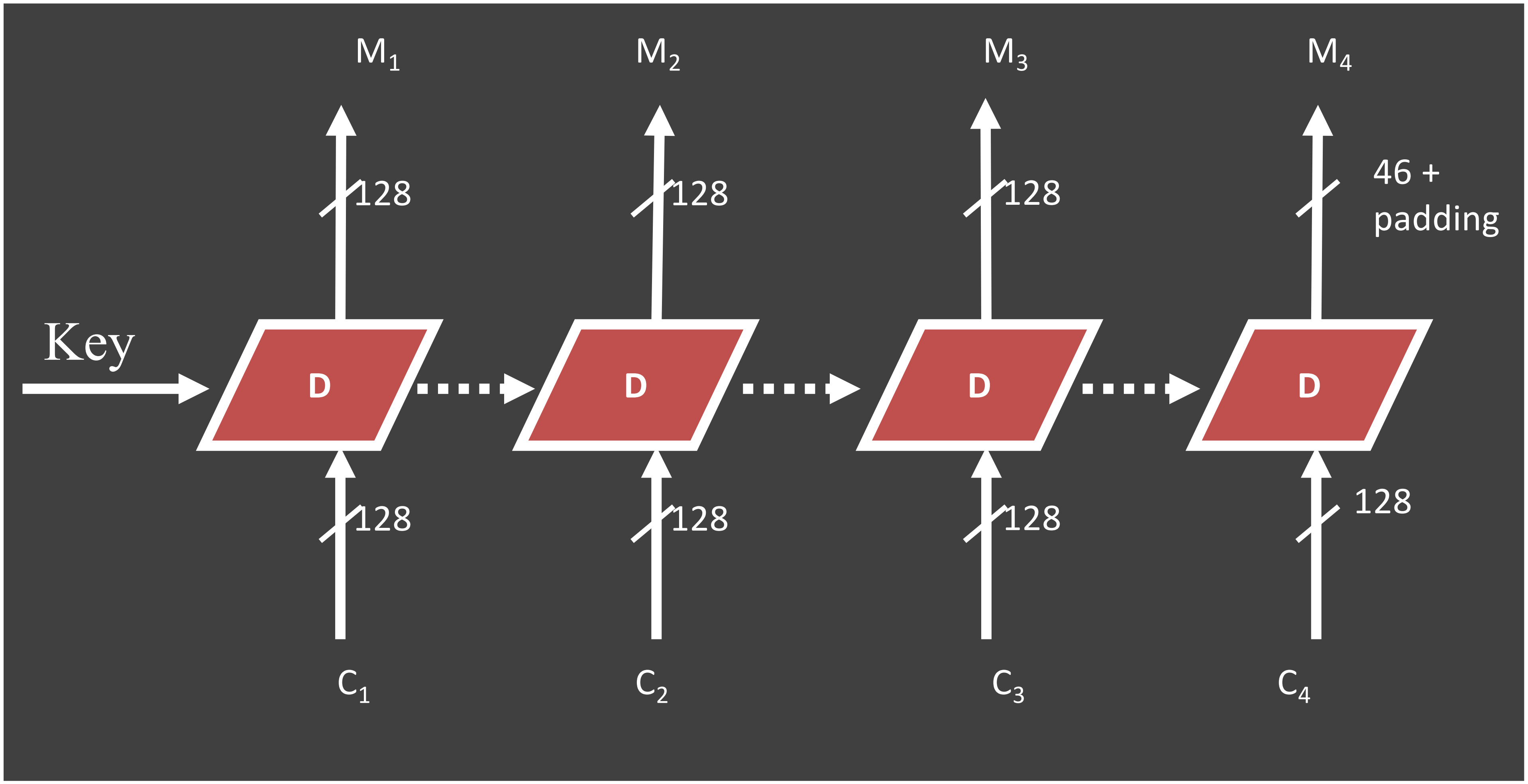
- *Information leakage*: Does it reveal info about the plaintext blocks?
- *Ciphertext manipulation*: Can an attacker modify ciphertext block(s) in a way that will produce a predictable/desired change in the decrypted plaintext block(s)?
  - Note: assume the structure of the plaintext is known, e.g., 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

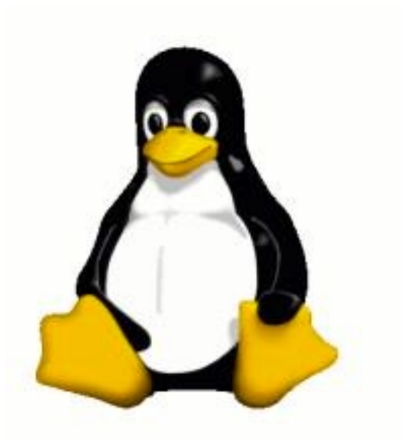
ECB Decryption

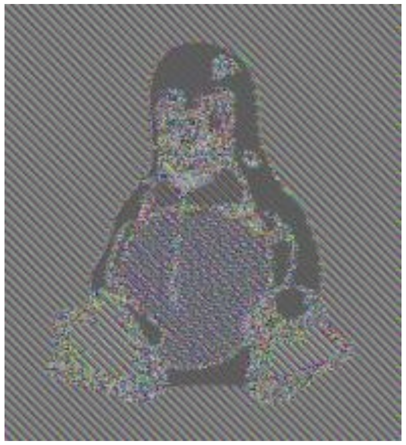


- Each block is independently decrypted

ECB Issues

- *Information leaks*: two ciphertext blocks that are the same
- *Manipulation*: switch ciphertext with predictable results on plaintext (e.g., shuffle).
- *Parallel*: yes
- *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

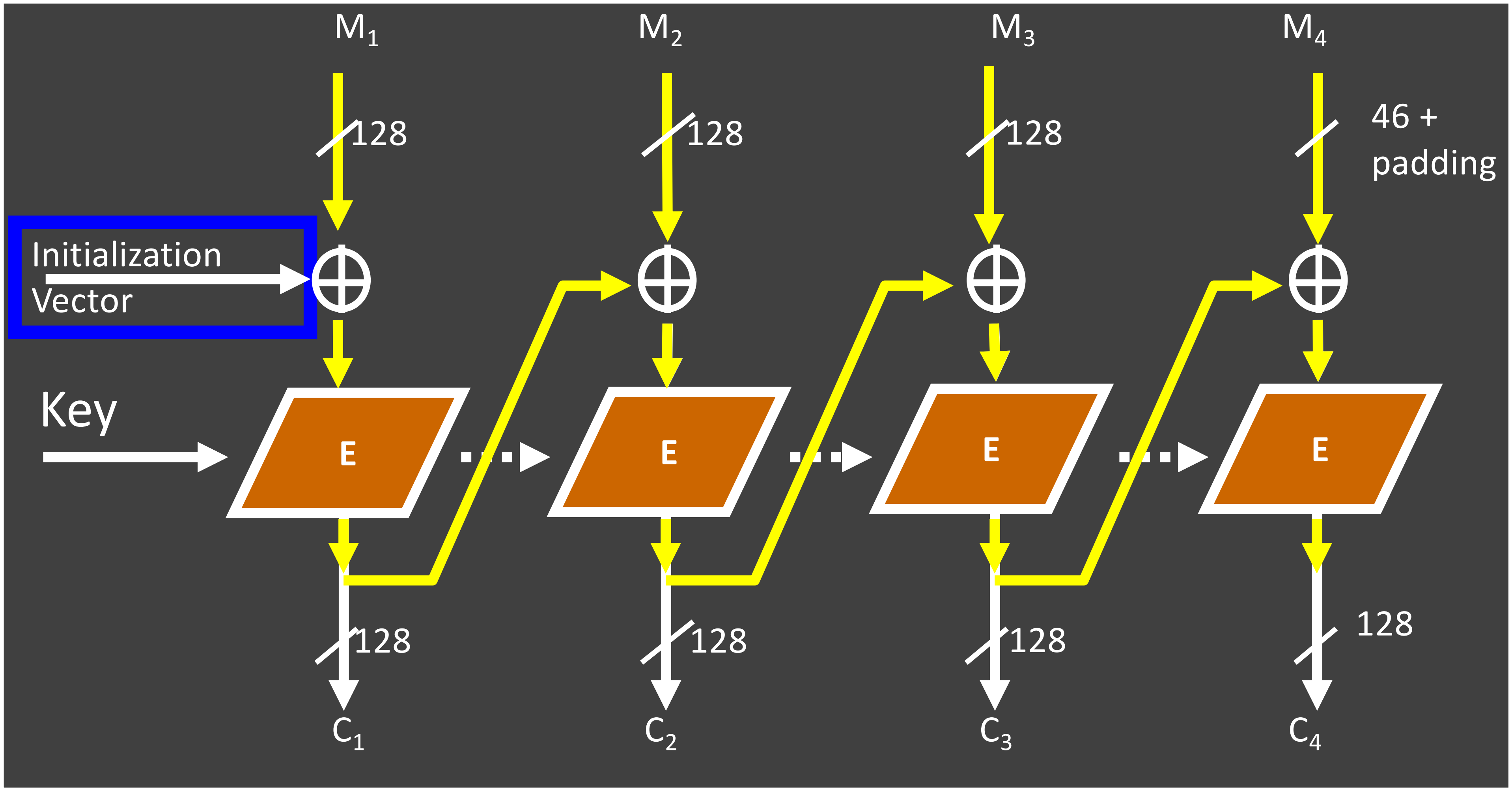
Sometimes it can seem like there's ECB everywhere. ECB on TV, ECB in music, it's endless. But that doesn't make it safe. Or right. So tune out and avoid ECB, no matter what your friends, the TV, or your favourite cryptographer tells you.

Canadian Joke Council

True Bugs Wait ♡

@natashenka
#truebugswait

Cipher Block Chaining (CBC)

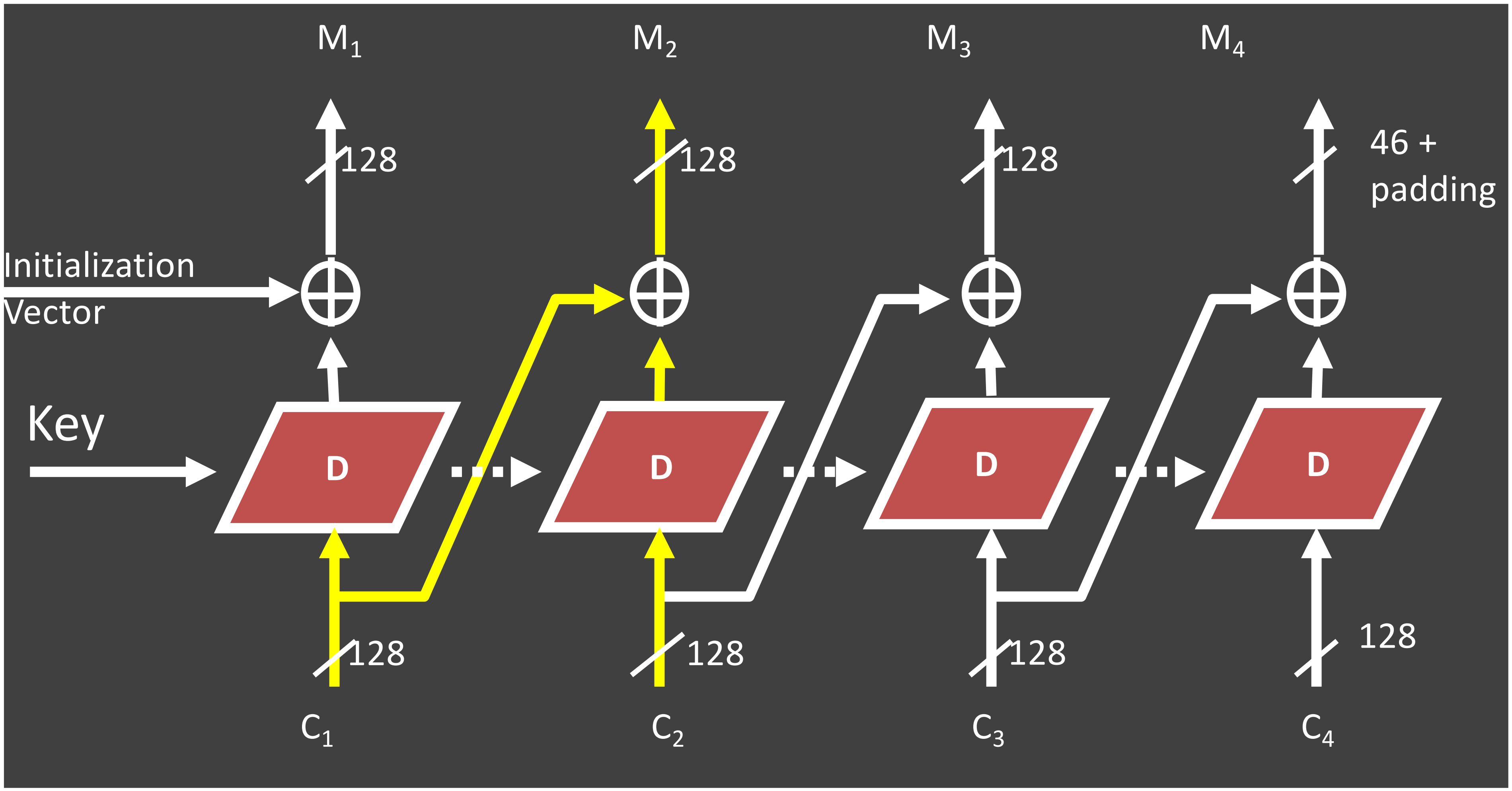


- Chaining dependency: each ciphertext block depends on all preceding plaintext blocks

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

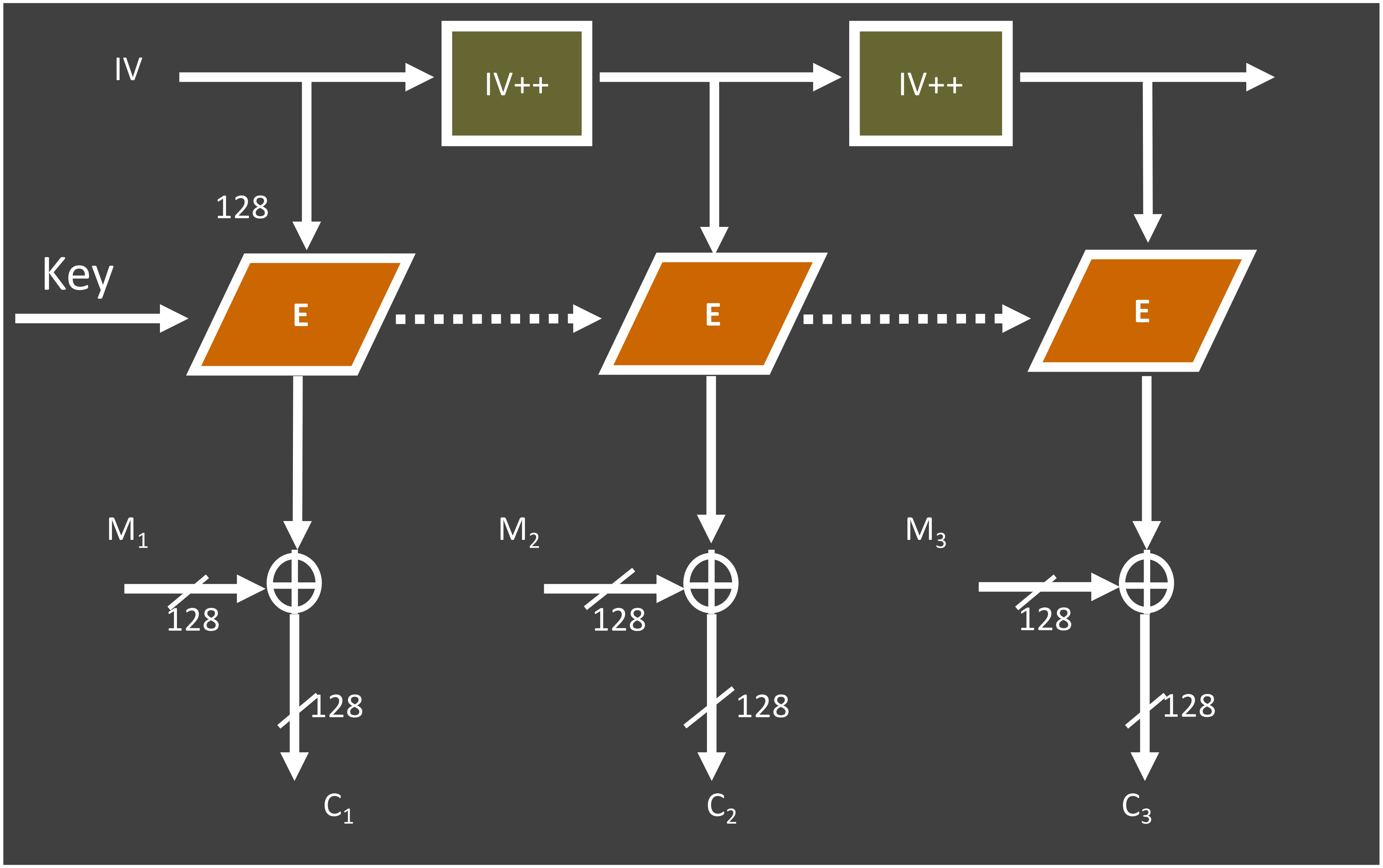


- How many ciphertext blocks does each plaintext block depend on?

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)



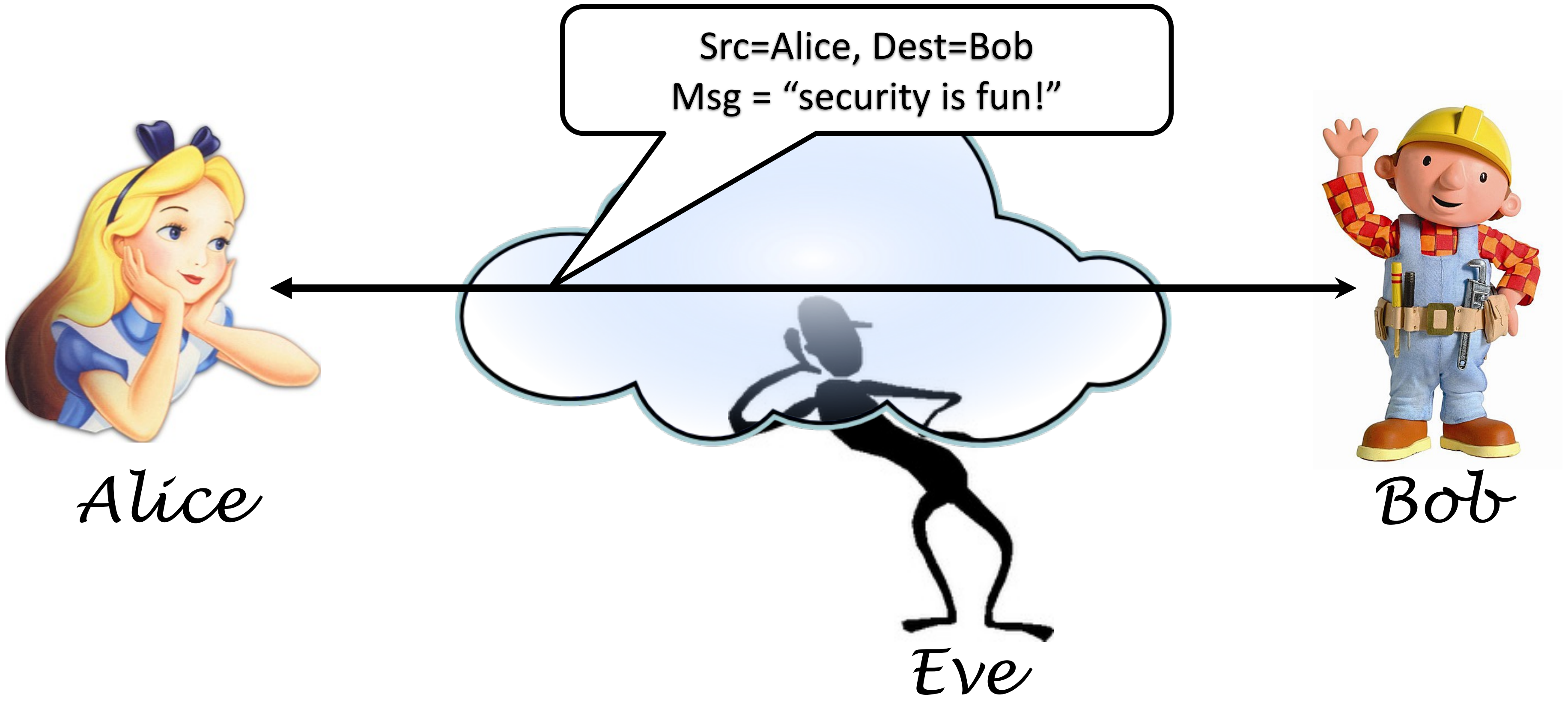
CTR Mode Properties

- 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

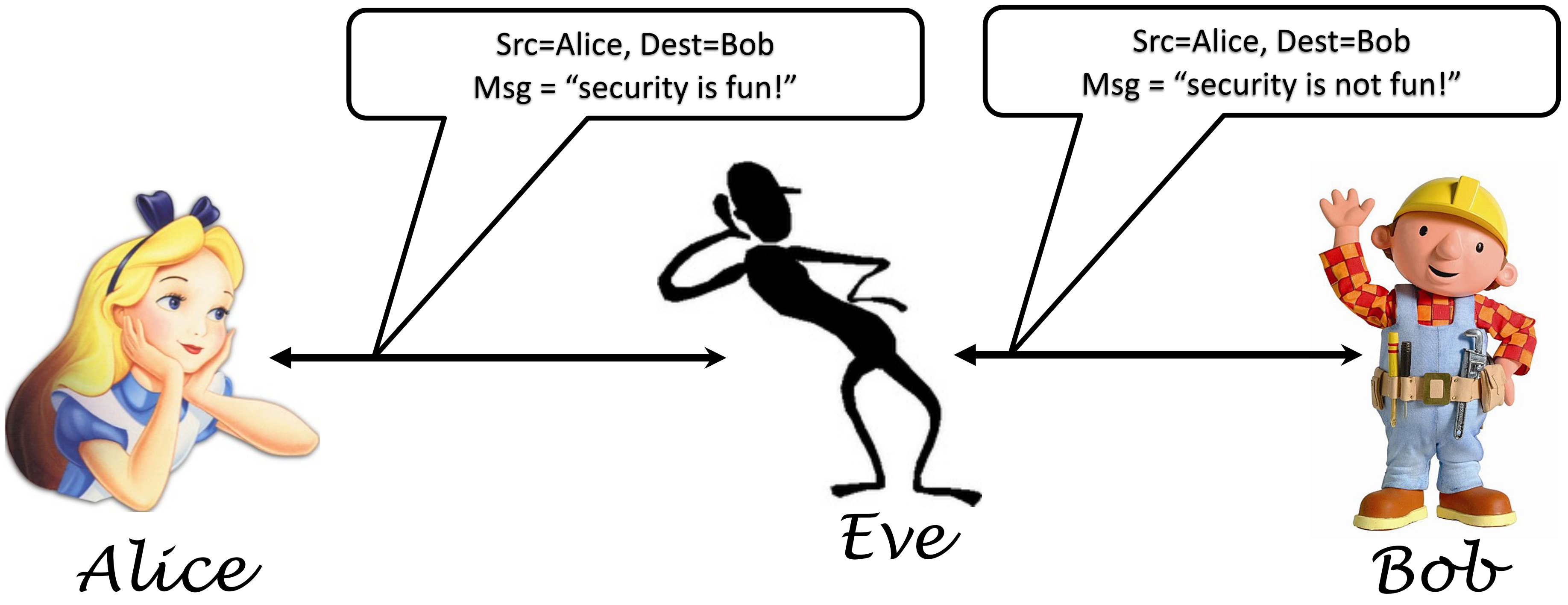
What encryption does and does not

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

Principals



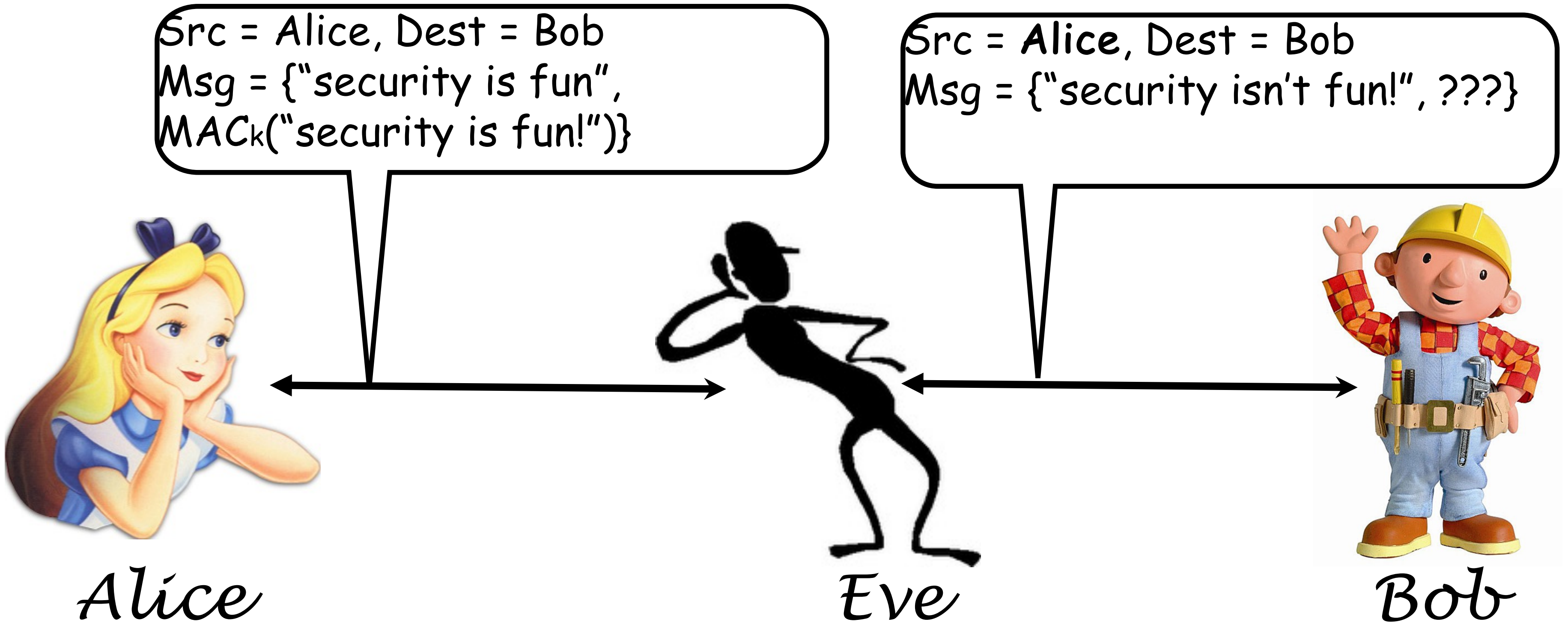
Man-in-the-Middle (MitM) attack



Message Authentication Codes (MACs)

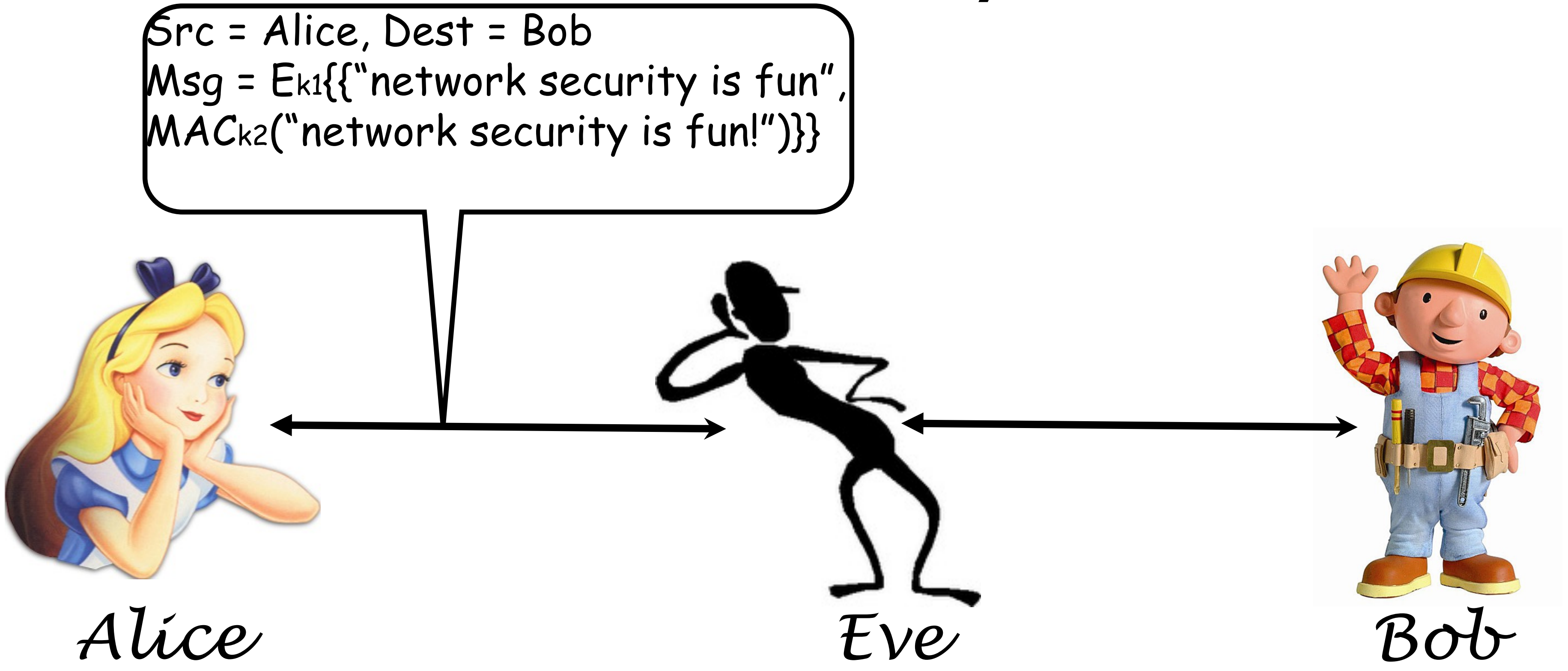
- MACs provide message **integrity** and **authenticity**
- $MAC_K(M)$ – use symmetric encryption to produce **short sequence of bits** that depends on both the message (M) and the key (K)
- MACs should be resistant to **existential forgery**: Eve should not be able to produce a valid MAC for a message M' without knowing K
- To provide confidentiality, authenticity, and integrity of a message, Alice sends
  - $E_K(M, MAC_K(M))$ where $E_K(X)$ is the encryption of X using key K
- Proves that M was encrypted (confidentiality and integrity) by someone who knew K (authenticity)

Message Authenticity


Src = Alice, Dest = Bob
Msg = {"security is fun",
MACk("security is fun!")}
Src = **Alice**, Dest = Bob
Msg = {"security isn't fun!", ???}
Alice
Eve
Bob

Without knowledge of *k*, Eve can't compute a valid MAC for her forged message!

Encryption and Message Authenticity


Src = Alice, Dest = Bob
Msg = Ek1{{"network security is fun",
MACk2("network security is fun!")}}
Alice
Eve
Bob

Without knowing *k1*,
Eve can't read Alice's message.

Without knowing *k2*, Eve can't compute a valid MAC for her forged message!

Cryptographic Hash Functions

- **Hash function** `h`: deterministic one-way function that takes as input an arbitrary message `M` (sometimes called a *preimage*) and returns as output `h(M)`, a small fixed length *hash* (sometimes called a *digest*)
- Hash functions should have the following two properties:
  - *compression*: reduces arbitrary length string to fixed length hash
  - *ease of computation*: given message `M`, `h(M)` is easy to compute

Hash functions are usually fairly inexpensive

(i.e., compared with public key cryptography)

```
adwait$ openssl speed sha
To get the most accurate results, try to run this
program when this computer is idle.
Doing sha1 for 3s on 16 size blocks: 9255072 sha1's in 2.97s
Doing sha1 for 3s on 64 size blocks: 6687775 sha1's in 2.97s
Doing sha1 for 3s on 256 size blocks: 3570692 sha1's in 2.98s
Doing sha1 for 3s on 1024 size blocks: 1234275 sha1's in 2.97s
Doing sha1 for 3s on 8192 size blocks: 174704 sha1's in 2.97s
Doing sha256 for 3s on 16 size blocks: 6374888 sha256's in 2.98s
Doing sha256 for 3s on 64 size blocks: 3926000 sha256's in 2.98s
Doing sha256 for 3s on 256 size blocks: 1697500 sha256's in 2.98s
Doing sha256 for 3s on 1024 size blocks: 532592 sha256's in 2.97s
Doing sha256 for 3s on 8192 size blocks: 72132 sha256's in 2.97s
Doing sha512 for 3s on 16 size blocks: 4913872 sha512's in 2.97s
Doing sha512 for 3s on 64 size blocks: 4915170 sha512's in 2.97s
Doing sha512 for 3s on 256 size blocks: 2160195 sha512's in 2.97s
Doing sha512 for 3s on 1024 size blocks: 795869 sha512's in 2.97s
Doing sha512 for 3s on 8192 size blocks: 113596 sha512's in 2.97s
OpenSSL 0.9.8zh 14 Jan 2016
built on: Jan 23 2017
options:bn(64,64) md2(int) rc4(ptr,char) des(idx,cisc,16,int) aes(partial) blowfish(idx)
compiler: -arch x86_64 -fmessage-length=0 -pipe -Wno-trigraphs -fpascal-strings -fasm-blocks -O3
D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DL_ENDIAN -DMD32_REG_T=int -DOPENSSL_NO_IDEA -DOPENSSL_PI
DOPENSSL_THREADS -DZLIB -mmacosx-version-min=10.6
available timing options: TIMEB USE_TOD HZ=100 [sysconf value]
timing function used: getrusage
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
sha1             49891.95k   144024.59k   307178.70k   425012.39k   482007.81k
sha256           34281.92k    84424.15k   146042.36k   183727.34k   198842.41k
sha512           26445.57k   105956.90k   186126.06k   274305.03k   313698.39k
```

Why might hashes be useful?

- **Message authentication codes (MACs):**
  - e.g.: `MAC`$_K$`(M) = h(K|M)`
    (but don't do this, use HMAC instead)
- **Modification detection codes:**
  - detect modification of data
  - any change in data will cause change in hash

Prof. Pedantic proposes the following hash function, arguing that it offers both compression and ease of computation.

- h(M) = 0 if the number of 0s in M is divisible by 3
- h(M) = 1 otherwise

Why is this a lousy crypto hash function?

Cryptographic Hash Functions

- Properties of good <u>cryptographic</u> hash functions:
  - **preimage resistance:** given digest y, computationally infeasible to find preimage x' such that h(x')=y
    (also called "one-way property")
  - **2nd-preimage resistance:** given preimage x, computationally infeasible to find preimage x' such that h(x)=h(x')
    (also called "weak collision resistance")
  - **collision resistance:** computationally infeasible to find preimages i,j such that h(i)=h(j)
    (also called "strong collision resistance")

Birthday Attack

- **Birthday Paradox:** chances that 2+ people share birthday in group of 23 is > 50%.
- General formulation
  - function f() whose output is uniformly distributed over H possible outputs
  - Number of experiments Q(H) until we find a collision is approximately:

$$Q(H) \approx \sqrt{\frac{\pi}{2} H}$$

  - E.g.,

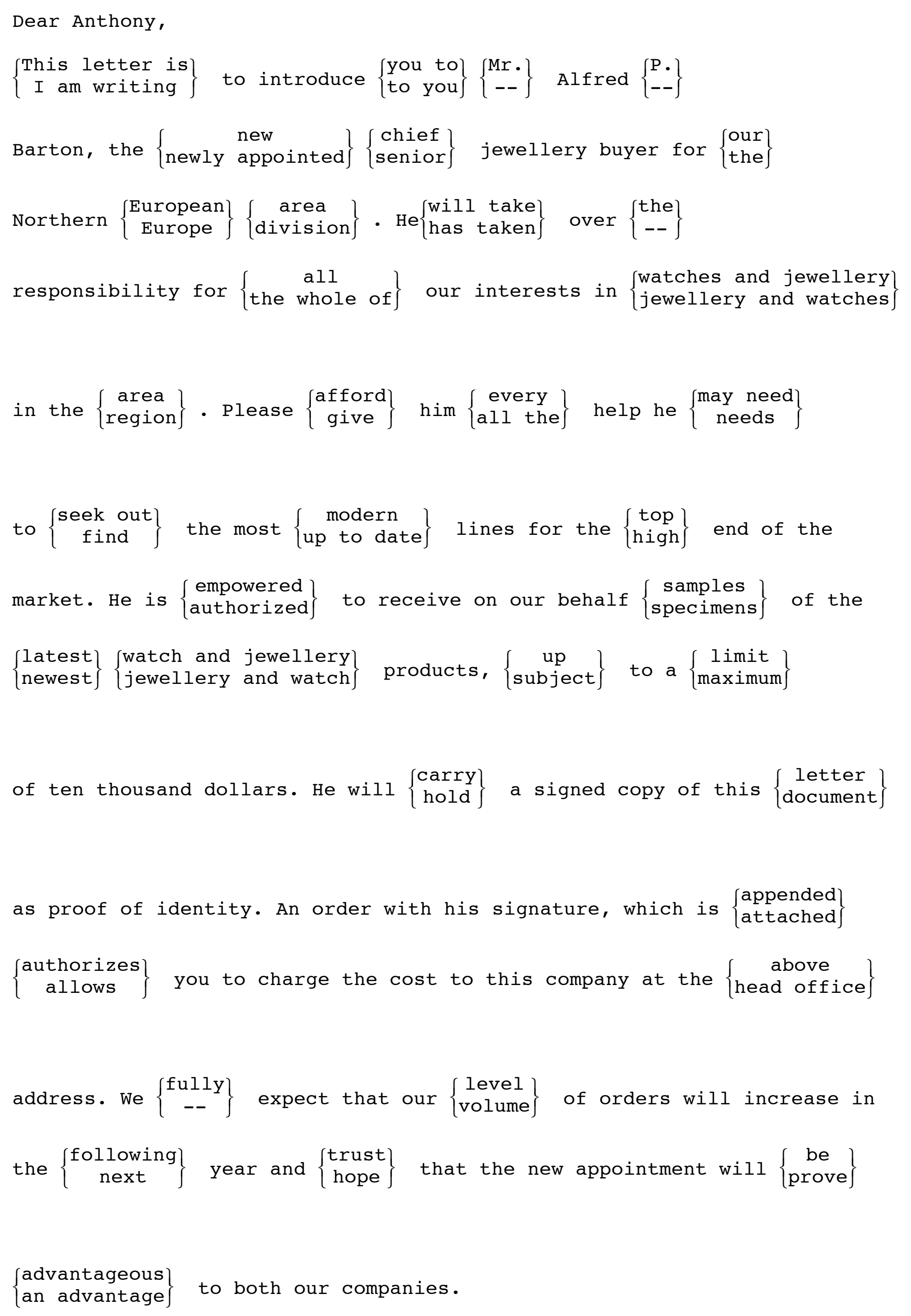
$$Q(365) \approx \sqrt{\frac{\pi}{2} 365} = 23.94$$

- Why is this relevant to hash sizes?

See: https://betterexplained.com/articles/understanding-the-birthday-paradox/

Practical Implications

- Choosing two messages that have the same hash h(x) = h(x') is more practical than you might think.
- Example attack: secretary is asked to write a "bad" letter, but wants to replace with a "good" letter.
  - Boss signs the letter after reading



Dear Anthony,

$\begin{Bmatrix}\text{This letter is}\\\text{I am writing}\end{Bmatrix}$ to introduce $\begin{Bmatrix}\text{you to}\\\text{to you}\end{Bmatrix}$ $\begin{Bmatrix}\text{Mr.}\\\text{--}\end{Bmatrix}$ Alfred $\begin{Bmatrix}\text{P.}\\\text{--}\end{Bmatrix}$

Barton, the $\begin{Bmatrix}\text{new}\\\text{newly appointed}\end{Bmatrix}$ $\begin{Bmatrix}\text{chief}\\\text{senior}\end{Bmatrix}$ jewellery buyer for $\begin{Bmatrix}\text{our}\\\text{the}\end{Bmatrix}$

Northern $\begin{Bmatrix}\text{European}\\\text{Europe}\end{Bmatrix}$ $\begin{Bmatrix}\text{area}\\\text{division}\end{Bmatrix}$. He $\begin{Bmatrix}\text{will take}\\\text{has taken}\end{Bmatrix}$ over $\begin{Bmatrix}\text{the}\\\text{--}\end{Bmatrix}$

responsibility for $\begin{Bmatrix}\text{all}\\\text{the whole of}\end{Bmatrix}$ our interests in $\begin{Bmatrix}\text{watches and jewellery}\\\text{jewellery and watches}\end{Bmatrix}$

in the $\begin{Bmatrix}\text{area}\\\text{region}\end{Bmatrix}$. Please $\begin{Bmatrix}\text{afford}\\\text{give}\end{Bmatrix}$ him $\begin{Bmatrix}\text{every}\\\text{all the}\end{Bmatrix}$ help he $\begin{Bmatrix}\text{may need}\\\text{needs}\end{Bmatrix}$

to $\begin{Bmatrix}\text{seek out}\\\text{find}\end{Bmatrix}$ the most $\begin{Bmatrix}\text{modern}\\\text{up to date}\end{Bmatrix}$ lines for the $\begin{Bmatrix}\text{top}\\\text{high}\end{Bmatrix}$ end of the

market. He is $\begin{Bmatrix}\text{empowered}\\\text{authorized}\end{Bmatrix}$ to receive on our behalf $\begin{Bmatrix}\text{samples}\\\text{specimens}\end{Bmatrix}$ of the

$\begin{Bmatrix}\text{latest}\\\text{newest}\end{Bmatrix}$ $\begin{Bmatrix}\text{watch and jewellery}\\\text{jewellery and watch}\end{Bmatrix}$ products, $\begin{Bmatrix}\text{up}\\\text{subject}\end{Bmatrix}$ to a $\begin{Bmatrix}\text{limit}\\\text{maximum}\end{Bmatrix}$

of ten thousand dollars. He will $\begin{Bmatrix}\text{carry}\\\text{hold}\end{Bmatrix}$ a signed copy of this $\begin{Bmatrix}\text{letter}\\\text{document}\end{Bmatrix}$

as proof of identity. An order with his signature, which is $\begin{Bmatrix}\text{appended}\\\text{attached}\end{Bmatrix}$

$\begin{Bmatrix}\text{authorizes}\\\text{allows}\end{Bmatrix}$ you to charge the cost to this company at the $\begin{Bmatrix}\text{above}\\\text{head office}\end{Bmatrix}$

address. We $\begin{Bmatrix}\text{fully}\\\text{--}\end{Bmatrix}$ expect that our $\begin{Bmatrix}\text{level}\\\text{volume}\end{Bmatrix}$ of orders will increase in

the $\begin{Bmatrix}\text{following}\\\text{next}\end{Bmatrix}$ year and $\begin{Bmatrix}\text{trust}\\\text{hope}\end{Bmatrix}$ that the new appointment will $\begin{Bmatrix}\text{be}\\\text{prove}\end{Bmatrix}$

$\begin{Bmatrix}\text{advantageous}\\\text{an advantage}\end{Bmatrix}$ to both our companies.

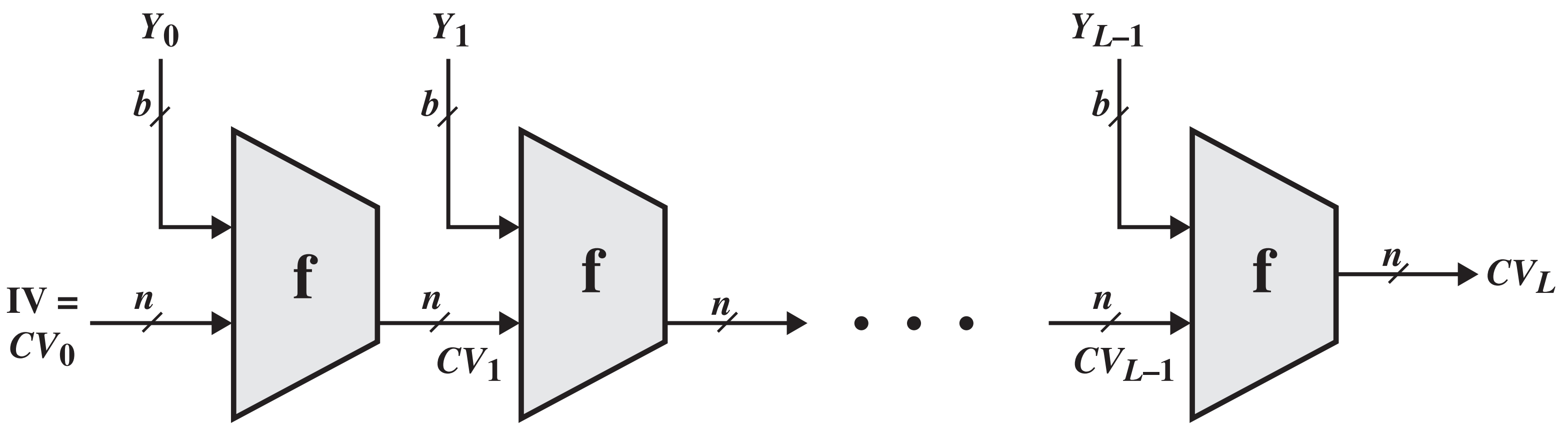
Figure 11.7 A Letter in 2^{37} Variations

(from Stallings, Crypto and Net Security)

Some common cryptographic hash functions

- MD5 (128-bit digest) [don't use this]
- SHA-1 (160-bit digest) [stop using this*]
- SHA-256 (256-bit digest)
- SHA-512 (512-bit digest)
- SHA-3 [recent competition winner]

General Structure of Hash



IV	=	Initial value	L	=	number of input blocks
CV_i	=	chaining variable	n	=	length of hash code
Y_i	=	*i*th input block	b	=	length of input block
f	=	compression algorithm			

(from Stallings, Crypto and Net Security)

Message Extension Attack

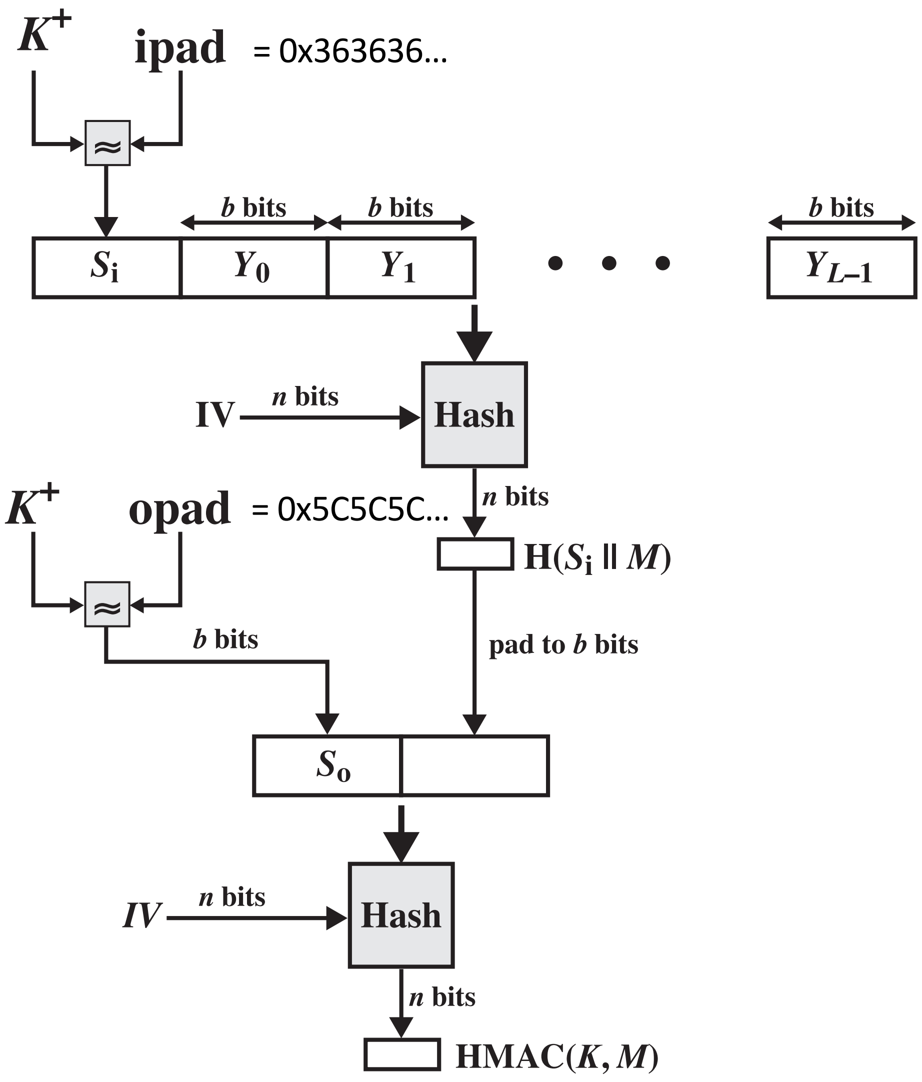
- Why is $MAC_k(M)$ = H(k|M) bad?
- How can Eve append M’ to M?
  - Goal: compute H(k|M|M’) without knowing k
- Solution: Use H(k|M) as IV for next f iteration in H()

A Better MAC

- Objectives
  - – Use available hash functions without modification
  - – Easily replace embedded hash function as more secure ones are found
  - – Preserve original performance of hash function
  - – Easy to use

HMAC

- HMAC(k,M) = H(k⊕opad || H(k⊕ipad || M))
  - Attacker cannot extend MAC as before
  - Prove it to yourself



(from Stallings, Crypto and Net Security)

The End