Cryptography
PVO Chapter 2

Imagine that Alice (A) is talking to Bob (B).

E.g., over the phone line.

E.g., Alice is a web browser, Bob a server.

Eve (E) is an eavesdropper trying to

read data

modify data

inject data

delete data

Eve is called the **adversary** (theory)
or **attacker** (systems)

Eve is called the **adversary** (theory)
or **attacker** (systems)
If Eve only reads data,
it is called **passive adversary**

Eve is called the **adversary** (theory)
or **attacker** (systems)
If Eve only reads data,
it is called **passive adversary**
If Eve can modify, inject, and delete data,
it is called **active adversary**

Eve is called the **adversary** (theory)
or **attacker** (systems)
If Eve only reads data,
it is called **passive adversary**
If Eve can modify, inject, and delete data,
it is called **active adversary**
If Eve can inject but cannot read,
it is called **blind adversary**

Eve is called the **adversary** (theory)
or **attacker** (systems)
If Eve only reads data,
it is called **passive adversary**
If Eve can modify, inject, and delete data,
it is called **active adversary**
If Eve can inject but cannot read,
it is called **blind adversary**
If Eve has infinite time and memory,
it is called **information theoretic** adversary

Eve is called the **adversary** (theory)
or **attacker** (systems)
If Eve only reads data,
it is called **passive adversary**
If Eve can modify, inject, and delete data,
it is called **active adversary**
If Eve can inject but cannot read,
it is called **blind adversary**
If Eve has infinite time and memory,
it is called **information theoretic** adversary
If Eve does not have infinite time and memory,
it is called **computationally bounded** adversary

Communication threat modelling

- Alice and Bob are communicating
- Eve has some access to the communication channel
  - all using Internet
  - is on same local network
  - taps the lines
  - is in same broadcast range
    - e.g., WiFi
  - or the lines run through Eve
    - Eve is router, ISP, AS
    - “on-path” adversary

Adversary can be not on-path and not blind

Adversary can be not on-path and not blind
e.g., Alice, Bob, and Eve are all in same Wifi range

Adversary can be not on-path and not blind
e.g., Alice, Bob, and Eve are all in same Wifi range
Eve can inject and read traffic...

Adversary can be not on-path and not blind
e.g., Alice, Bob, and Eve are all in same Wifi range
Eve can inject and read traffic...
...but not easily modify or delete

Example protocol

- file system protocol
  - mount
  - open
  - read
  - write
  - close
  - unlink
  - umount

Alice (user) connects to Bob (remote FS)

Alice (user) connects to Bob (remote FS)
Alice issues commands:

Alice (user) connects to Bob (remote FS)
Alice issues commands:
mount

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

write SOMETHING to SOMEFILE

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

write SOMETHING to SOMEFILE

close SOMEFILE

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

write SOMETHING to SOMEFILE

close SOMEFILE

open OTHERFILE

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

write SOMETHING to SOMEFILE

close SOMEFILE

open OTHERFILE

read OTHERFILE

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

write SOMETHING to SOMEFILE

close SOMEFILE

open OTHERFILE

read OTHERFILE

close OTHERFILE

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

write SOMETHING to SOMEFILE

close SOMEFILE

open OTHERFILE

read OTHERFILE

close OTHERFILE

unlink OTHERFILE

Alice (user) connects to Bob (remote FS)

Alice issues commands:

mount

open SOMEFILE

write SOMETHING to SOMEFILE

close SOMEFILE

open OTHERFILE

read OTHERFILE

close OTHERFILE

unlink OTHERFILE

umount

What are some things Eve could do?

What are some things Eve could do?
Assume there is no security
and Bob is reachable on the Internet

Suppose Eve is on-path

- man-in-the-middle attack
  - read Alice's data
  - read Alice's FS usage (metadata)
    - e.g., file names
- impersonation attack
  - impersonate Bob to Alice
    - may not "look" like Bob
- denial of service attack
  - delete traffic so Bob can't be used

Suppose Eve is blind

- impersonate Alice to Bob
  - can read files
  - can unlink (delete) all files
    - denial of service
  - how could Eve know file names?

Can Eve impersonate Bob to Alice when Eve is off-path?

Amend protocol

- assume Alice and Bob have a secret number
  - every command must include this magic number
  - Bob ignores messages that don't have this magic number
  - like a password is sent with each command
- what can Eve do?

Attacks

- non-blind
  - eavesdrop on any message to learn magic number
    - can impersonate Alice after
- blind
  - try every possible number to see if its the right one
    - brute-force guessing attack

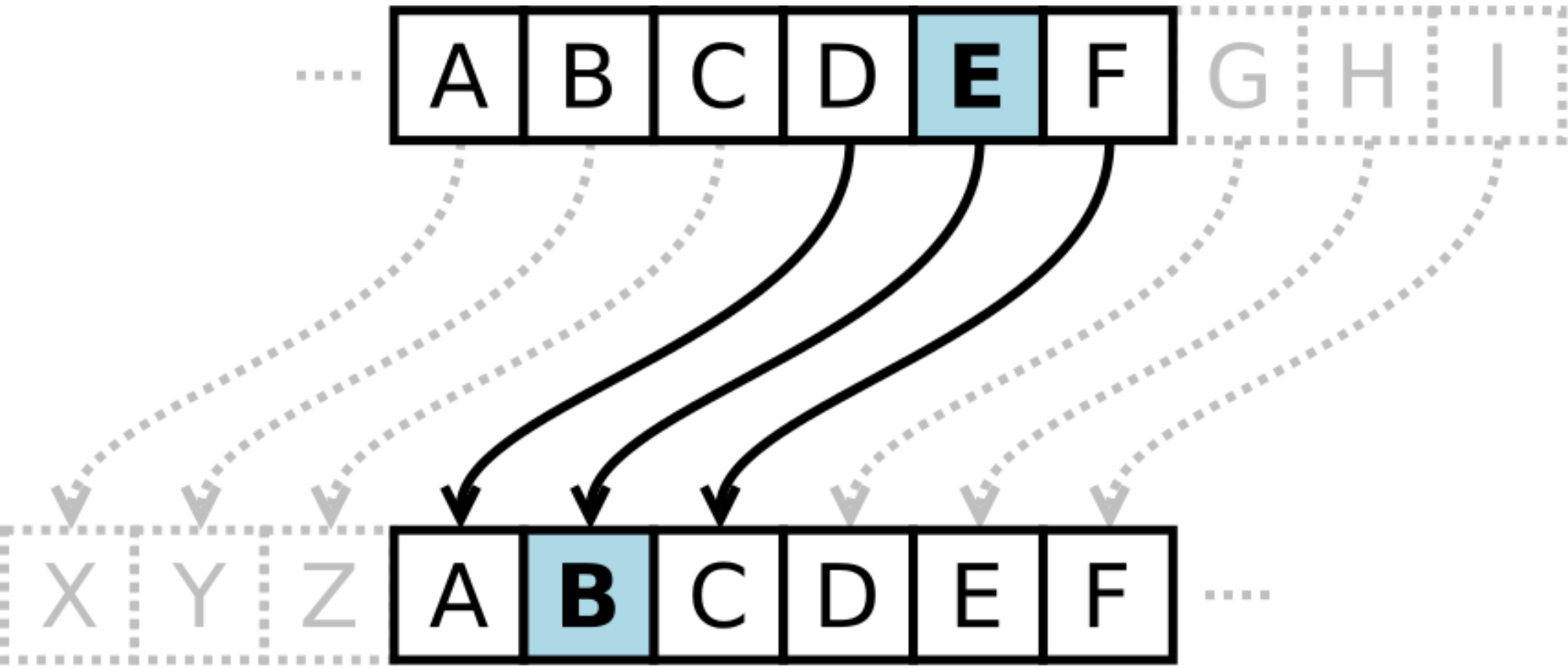
Cryptography – lit. secret writing

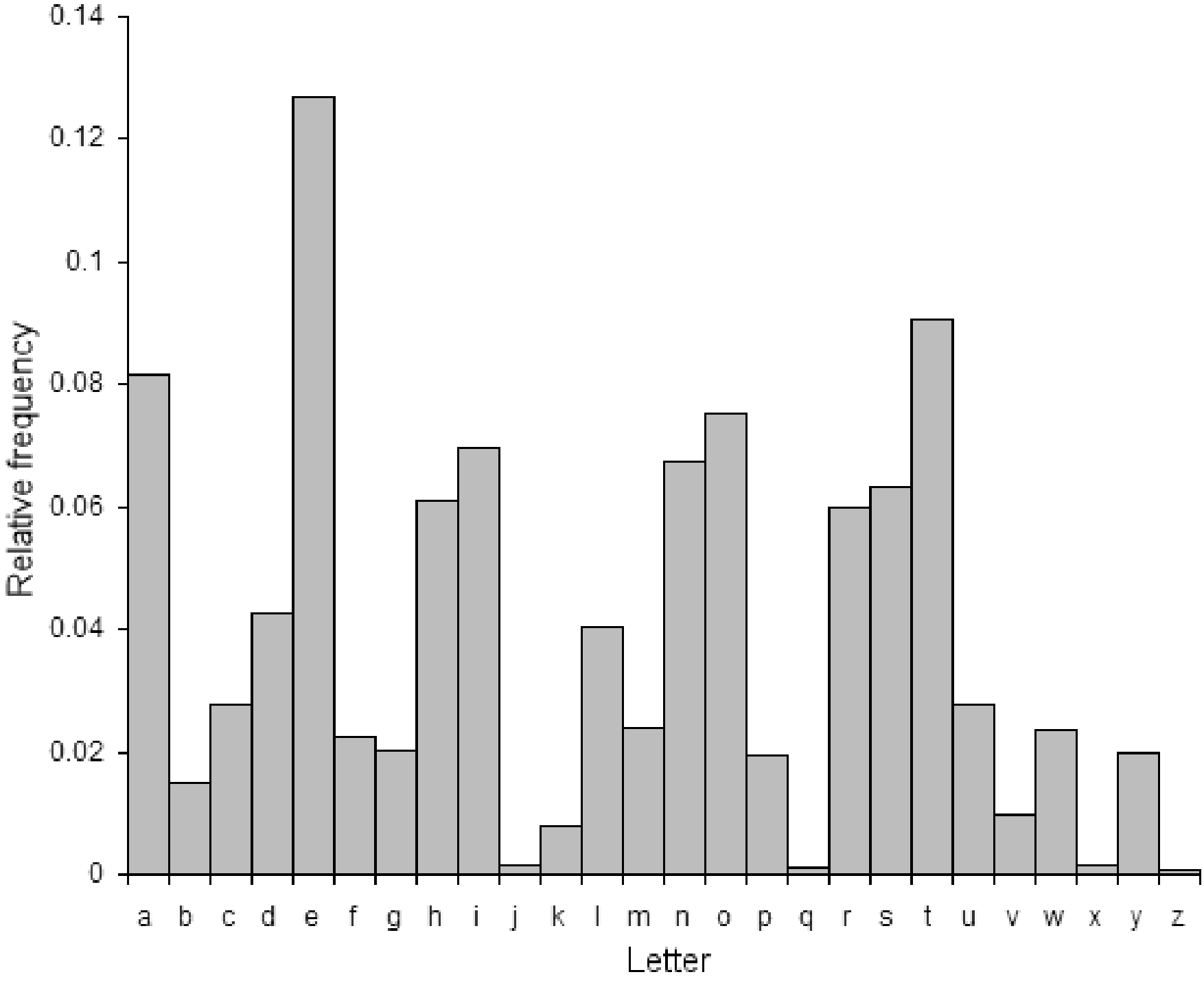
Cryptography is the art of sending messages so that no one else can read them.

The actual message is called the **plain-text**.
The encrypted message is called the **cipher-text**.

Caesar Cipher

- simple alphabet shift
- $E_k(x) = x + k (\text{mod } 26)$
- $D_k(x) = x - k (\text{mod } 26)$


A B C D E F G H I
X Y Z A B C D E F


0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
Relative frequency
a b c d e f g h i j k l m n o p q r s t u v w x y z
Letter

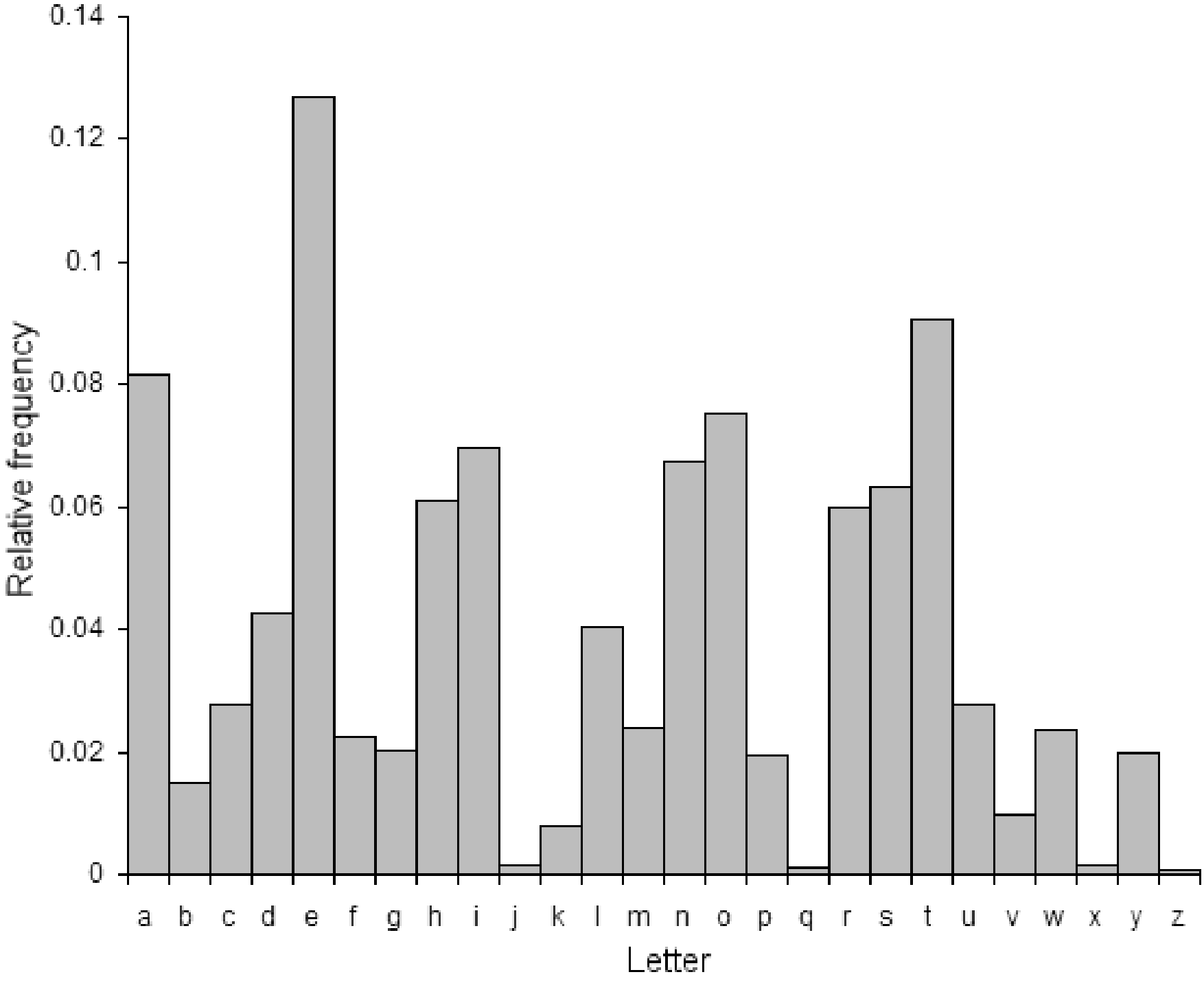
Even without frequency analysis,
there's not much work to do:
only 25 keys to try.

Even without frequency analysis,
there's not much work to do:
only 25 keys to try.
brute-force search

Consider “vuhzzcmuqymigy”.
What is the key?

Letter Substitution Cipher

- better than a simple shift
- each letter has its own replacement
- ‘a’ = ‘r’, ‘b’ = ’d’, etc.
- still vulnerable to frequency analysis


0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
Relative frequency
a b c d e f g h i j k l m n o p q r s t u v w x y z
Letter

Cracking Substitution

- most common is ‘e’
- guess for the next most common ones
- look to see if you get words or nonsense
- keep repeating until you work out the letters
- how many “keys”?

XOR on binary strings

- operates bit-by-bit
  - $x_1 x_2 \ldots x_n \oplus y_1 y_2 \ldots y_n = (x_1 \oplus y_1)(x_2 \oplus y_2) \ldots (x_n \oplus y_n)$
- ASCII/UTF-8 for 'b' is 98, 'h' is 104
  - b: $0b01100010$
  - h: $0b01101000$
  - $b \oplus h : 0b00001010 = 8 + 2 = 10$ or "newline" (\n)
- similar to addition modulo 26 but suitable for bytes
  - benefit that E and D are the same function
  - $x = x \oplus y \oplus y$

Vigenère Cipher

- also known as “le chiffre indéchiffrable
- basically just multiple Caesar ciphers at different offsets
- the key is a sequence of shifts values that loop
  - e.g., Caesar cipher with key C is add CCCCCCCCCCC to the message
  - Vigenère cipher with key HIDDEN is to add HIDDENHIDDENHIDDEN
- this worked from 1500s until 1800s
- so how do you break it?

Vernam Cipher

- like Vigenère, but the secret key has same length as message
  - called one-time pad (OTP)
- encrypt M by XORing it with K ($E_K(M) = M \oplus K$)
- decrypt C by XORing it with K ($D_K(C) = C \oplus K$)
- $D_K(E_K(M)) = D_K(M \oplus K) = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 00\ldots0 = M$
- $|K| = |M| = |C|$
  - given a ciphertext c, choose any possible message m. Then $k_m = c \oplus m$ is a key that decrypts c to m
  - another m' has a corresponding $k_{m'} = c \oplus m'$.
  - no way to "prove" that any particular k is the "correct" one if all are equally possible.

One-Time Pad

- an example of “perfect” security
- no frequency analysis can help decode it
- “information theoretically” secure
  - there is a key for all plaintext-ciphertext combinations
  - you can’t ever prove that it’s the “correct” deciphering

One-Time Pad

- securing the delivery of all the one-time pads
  - you don't gain anything by sending a new one-time pad using old one-time pad (why?)
- ensuring there is sufficient padding material ahead of time
  - i.e., before it is needed
  - why does this matter?
- making sure the pad is never **used twice**
- making sure the pad is **generated randomly**
- encryption is trivially "malleable"
- no integrity or authenticity

Why is it called a one-time pad?

Two-time pad

- $c_1 = m_1 \oplus p$
- $c_2 = m_2 \oplus p$
- $\Rightarrow c_1 \oplus c_2 = m_1 \oplus m_2$ (why?)
- how can we break this?

Stream Ciphers

- one-time pads in practice
  - XORs a long random string to the plaintext
- instead of a huge random string as the key it is just a small seed
  - used for a cryptographically secure pseudo-random number generator (PRNG)
    - generates unpredictable numbers
    - provided adversary does not know the key
  - it creates the one-time pad
- this key cannot be reused (why?)

Stream Ciphers are not secure against an information theoretic adversary. (Why?)

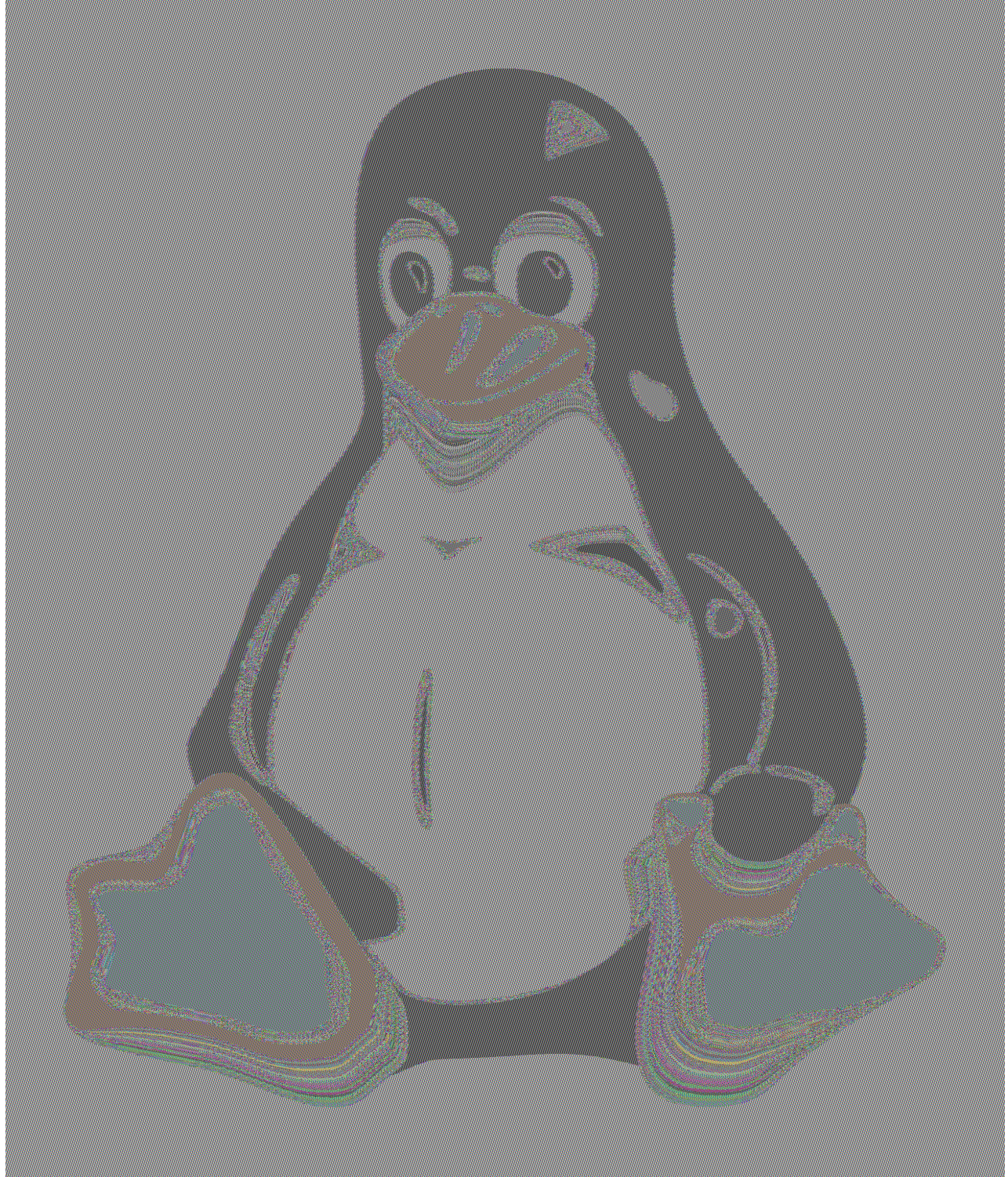
How do you get around not reusing the key?
That is, how do you send two messages?

Are stream ciphers malleable?

Block Ciphers

- standard for modern crypto
  - was DES (data encryption standard)
  - now AES (advanced encryption standard)
- block ciphers divide the message into blocks
  - block size 128 bit, 192 bit, 256 bit
- encrypts the blocks one by one
- decrypts the blocks one by one
- encryption is a function that maps some block value to another
  - a key defines a permutation from one block value to another

Block cipher naively used to encrypt a picture:



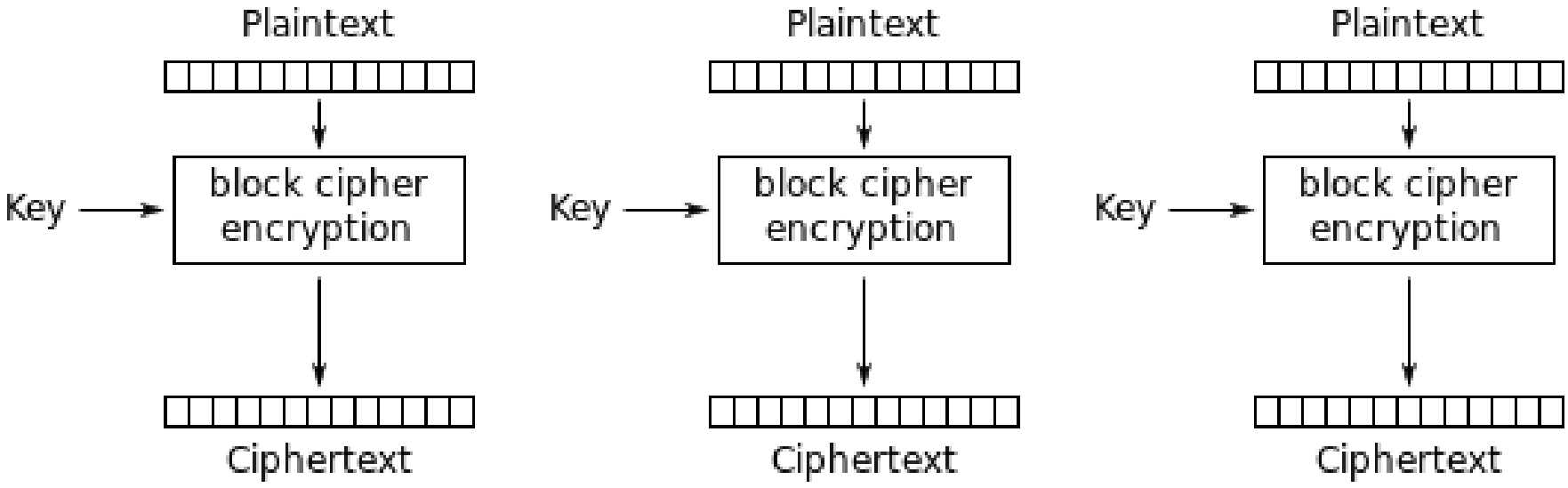
Clearly you can learn something about the plaintext

What happened?

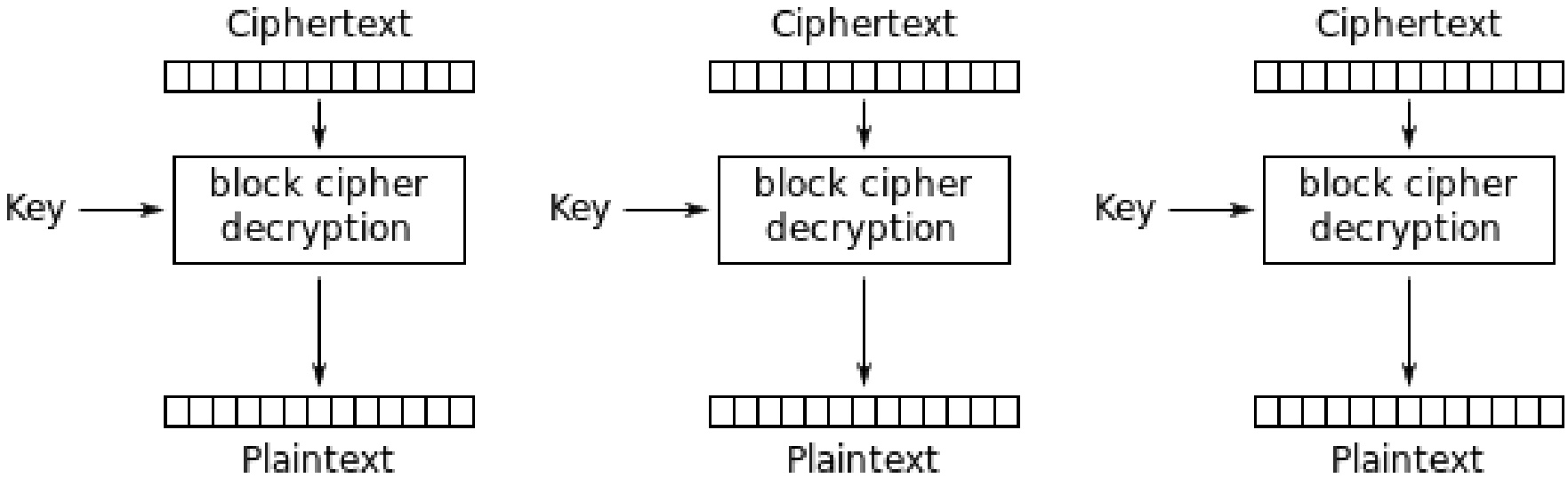
- each block is encrypted separately
- there's only so many different values that can appear as plaintext blocks
- if block value X encrypts to Y, and X appears alot, then Y appears alot in the ciphertext
- this is **frequency analysis**!
  - instead of at the level of letters it is at the level of blocks
- like encrypting a phone number digit by digit
  - if you know the area code is E(4) E(0) E(3), you learn where those appear in the rest

ECB Mode

- electronic codebook mode
- each block of data has an encrypted version and it's looked up
- shows when the same original data is encrypted again
  - this is the encrypted penguin!



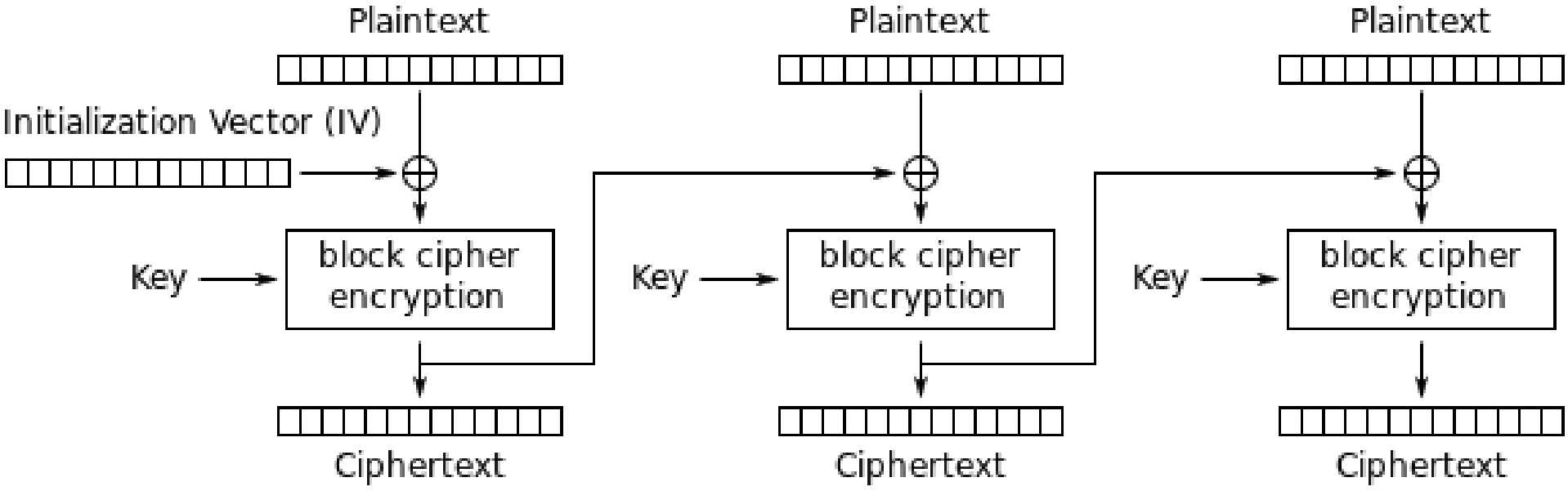
Electronic Codebook (ECB) mode encryption



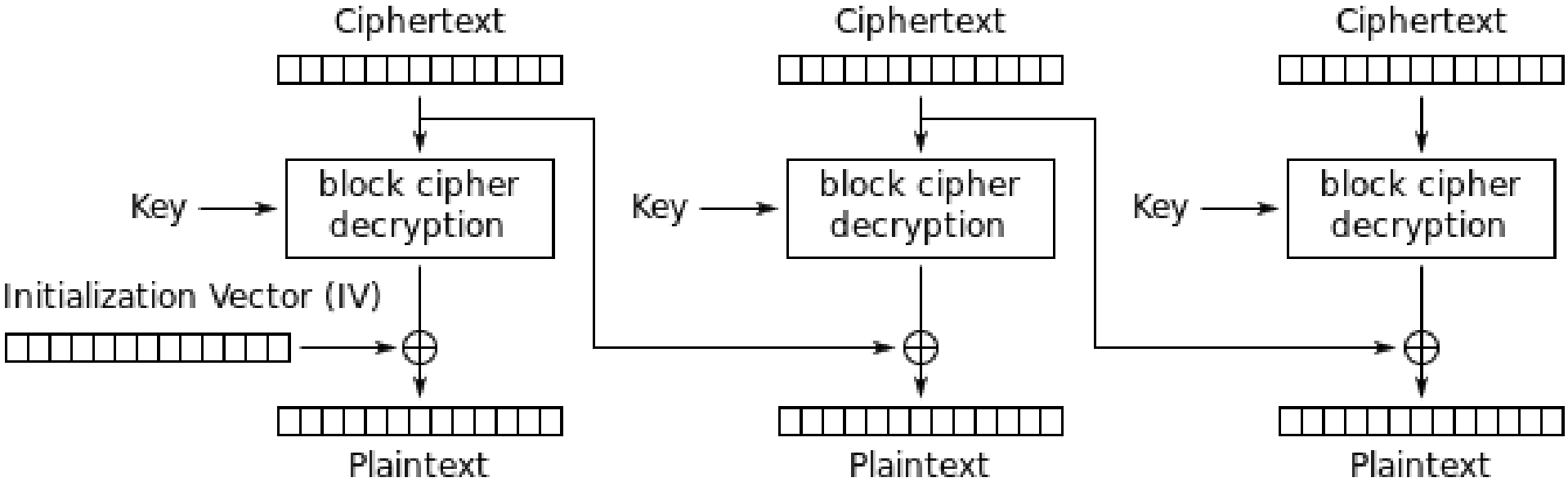
Electronic Codebook (ECB) mode decryption

CBC Mode

- cipherblock chaining mode
- an **initialization vector** (IV) is used alongside the key
  - IV is XORed to the block of plaintext
  - ciphertext changes with the IV
  - ciphertext of previous block is used as IV for next block



Cipher Block Chaining (CBC) mode encryption



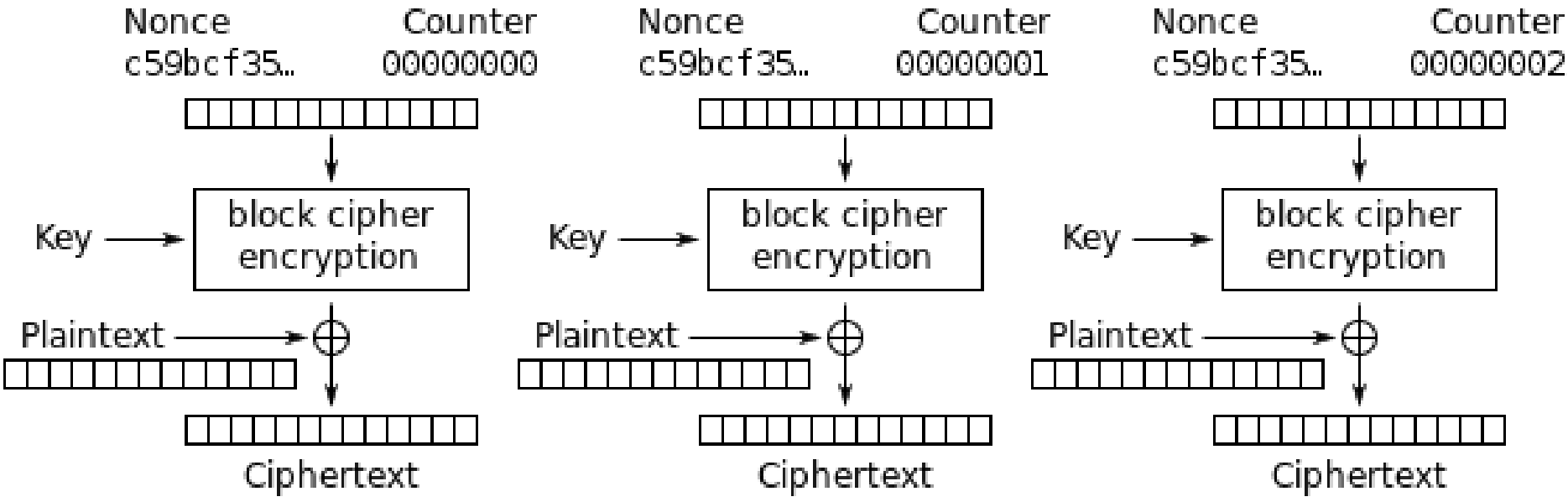
Cipher Block Chaining (CBC) mode decryption

CBC Mode

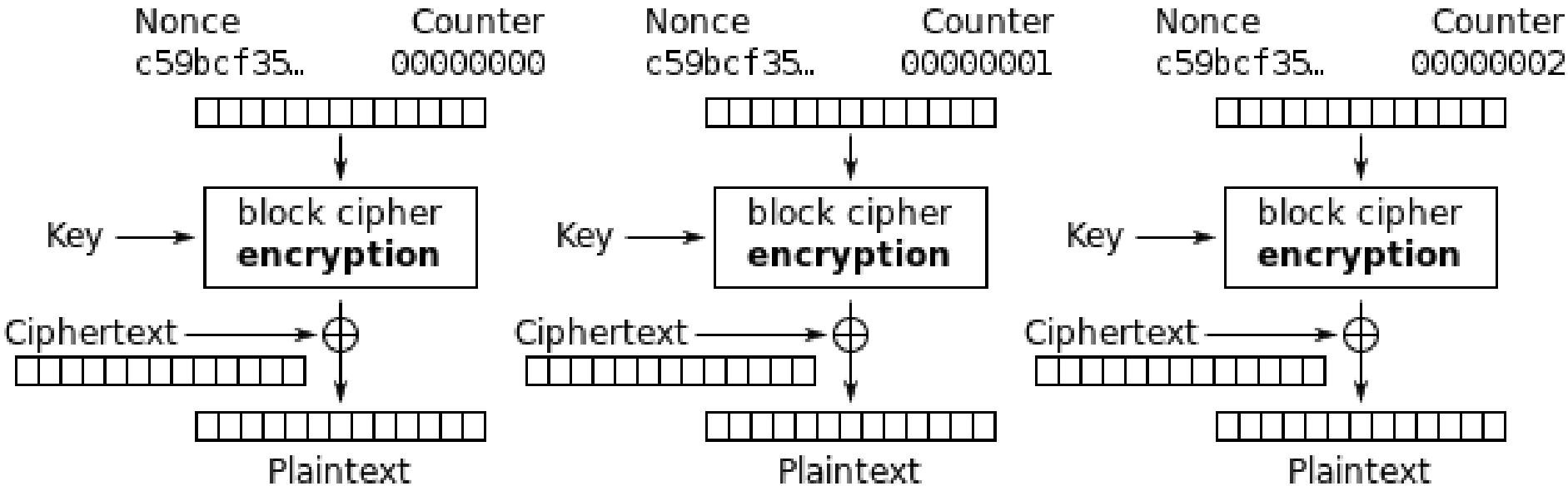
- what happens to recovery if some ciphertext is damage?
- is CBC parallelizable?
- does CBC support random access?
- what happens if the IV is reused?
  - does it behave like ECB?

CTR Mode

- counter mode
- turns block cipher into a stream cipher
- encrypts a sequence of numbers and XORs it to data
  - notion of IV is called a nonce (number used once)
  - nonce is the initial number
    - its encryption is for octet 0 of plaintext
    - why is it needed?
  - encryption does not need plaintext to operate
- supports missing data in the stream



Counter (CTR) mode encryption



Counter (CTR) mode decryption

GCM Mode

- Galois/Counter mode
- CTR mode with added authentication
- considered best practices

Amend protocol

- assume Alice and Bob share a secret key
  - use encryption to encrypt the message
  - only Alice and Bob can read it
- instead of sending "mount", sends E_k(mount)
  - E_k is an encryption function for a key k
  - D_k is corresponding decryption function
  - symmetric key crypto, implemented as AES

Attacks?

Encryption provides **confidentiality**
to the communication channel

Encryption provides **confidentiality**
to the communication channel
Eve can be a MITM but can't understand

Encryption provides **confidentiality**
to the communication channel
Eve can be a MITM but can't understand
Alice and Bob are only ones with key
and computing E_k or D_k requires
knowledge of the secret key k.

Does encryption prevent impersonation?

Replay attack

- suppose Eve is a MITM
- Alice connects to Bob through Eve
  - $A \rightarrow B : E_k(\text{mount})$
  - $A \rightarrow B : E_k(\text{read filename})$
  - $A \rightarrow B : E_k(\text{umount})$
- if Eve captured the messages, Eve can replay them later
  - no evidence that it is actually Alice saying it now

Replaying a READ may not be that interesting

Replaying a READ may not be that interesting
but replaying an UNLINK might be.

How could Eve figure out
what the commands are?

How could Eve figure out
what the commands are?
Defenders have protocols
(mount / commands / unmount)

How could Eve figure out
what the commands are?
Defenders have protocols
(mount / commands / unmount)
Or the command might trigger
or may be in response to
something externally observable

How could Eve figure out
what the commands are?
Defenders have protocols
(mount / commands / unmount)
Or the command might trigger
or may be in response to
something externally observable
e.g., a door becomes unlocked so
what was just said may be unlock

Or the crypto isn't **semantically** secure:
if $E_K($ read $)$ always appears
the same way when it is encrypted,
its possible to start seeing patterns.

E.g., what if E_K(read filename)
was the same as E_K(read)E_K(filename)?

E.g., what if E_K(read filename)
was the same as E_K(read)E_K(filename)?
if you already knew E_K(unlink) you could
replay E_K(unlink filename) by
sending E_K(unlink)E_K(filename).

A **secure** channel is not the
same as an **authentic** channel.

Amend protocol

- Alice connects to Bob
- Bob sends Alice a **challenge**
  - e.g., current time, random number
- Alice sends E_k(challenge message)
- does this prove to Bob it is Alice?

Brute-force attack

- suppose the challenge is 8 bits (1 octet)
  - 256 different challenges
- will be reused quickly
- will be easy to build a collection of all possible values

Suppose it is 128-bit challenge

Suppose it is 128-bit challenge
128 is minimum to consider un-bruteforcable

Suppose it is 128-bit challenge
128 is minimum to consider un-bruteforcable
does it work now?

Depends on how E works

Depends on how E works

suppose $E_k($ challenge message $)$

is the same as $E_k($ challenge $)E_k($ message $)$

then what could Eve do?

Depends on how E works

suppose $E_k($ challenge message $)$

is the same as $E_k($ challenge $)E_k($ message $)$

then what could Eve do?

Encryption of challenge is now basically

a password sent in plaintext

Some ways of encrypting are **malleable**
so if Eve flips the 1000th bit of the encrypted message, the 1000th bit of the output is flipped.

Some ways of encrypting are **malleable**
so if Eve flips the 1000th bit of the encrypted message, the 1000th bit of the output is flipped.
encrypting is no guarantee that what is received is what was sent.

In general, decrypting a ciphertext
and having the 500th bit correct
is not a guarantee that any other bit
was also correctly decrypted.

Communication Channels

- **insecure channel**: standard, like speaking over the telephone
- **secure channel**: data is encrypted and you decrypt it to see
- **authentic channel**: data sent was not tampered with and you know the source of it
- **secure authentic channel**: data is secure and authentic

A known person speaking in a room is
an authentic but not secure channel.

A known person speaking in a room is
an authentic but not secure channel.
Encryption makes the channel secure
but does nothing against replay,
modification, deletion, or proving
the identity of the source.