ECS655U: Security Engineering

Week 1: Setting the Scene

Dr Arman Khouzani

January 11, 2019

EECS, QMUL

Table of contents

1. Basic Principles

2. Historical Cryptosystems (and why they are all useless!)

Learning Outcomes

- Motivate information security and security engineering;
- Identify basic security services that cryptography provides;
- Describe the basic model of a cryptosystem and familiarize with basic cryptographic terminologies;
- Recognise the differences between symmetric and public-key (asymmetric) cryptosystems;
- Relate a number of historical cryptosystems to the basic model of a cryptosystem;
- Understand weaknesses of these historical cryptosystems that make them unsuitable to use.

Basic Principles

Motivation: Information Security

Information Security (a.k.a *Cyber Security*)
Protection of information and information systems

Security Engineering
The knowledge and skills to achieve information security: how to build dependable systems in the face of *malicious* adversarial agents

- The scope of system can be as narrow as a single e-ID card reader, or as wide as an entire organization (involving PCs, smart phones, servers, wired and wireless networks, OS, middleware, applications, external services, staff, customers, etc.).

Motivation: Information Security

Related (but different) disciplines in systems engineering (including software engineering):

Reliability engineering: Building systems that continue to function as expected (for its life-cycle) in the face of (*non-malicious*) errors and/or mishaps, e.g. a nuclear plant to withstand an earthquake.

Safety engineering: Building systems that "fail-safe", i.e., if a component fails, the system should behave in an expected way, e.g. critical functionalities remain and catastrophic events prevented, e.g., a nuclear plant to shut down if cooling pumps fail.

Motivation: Information Security

- There is nothing new about information security, but as more applications and services conducted electronically, its profile is rising.
- Business automation, a trend threatening other jobs, is often done to increase productivity or availability, and in fact, can exacerbate security.
- It may also be what is preventing very exciting technological innovations from releasing their true potentials: Internet-of-Things, smart appliances, smart cars, smart homes, smart cities, smart transportation, smart government,. . .

Motivation: Cryptography

Cryptography provides the tools and techniques for a lot of security technologies.

- People use it everyday: *Internet , Email, Messaging, Wifi, Molibe telephony, e-banking (EMV cards), digital rights (e.g. in video broadcasting), eID cards, . . .*
- Or they may be using increasingly: *Tor, Bitcoin, . . .*

Motivation: Cryptography

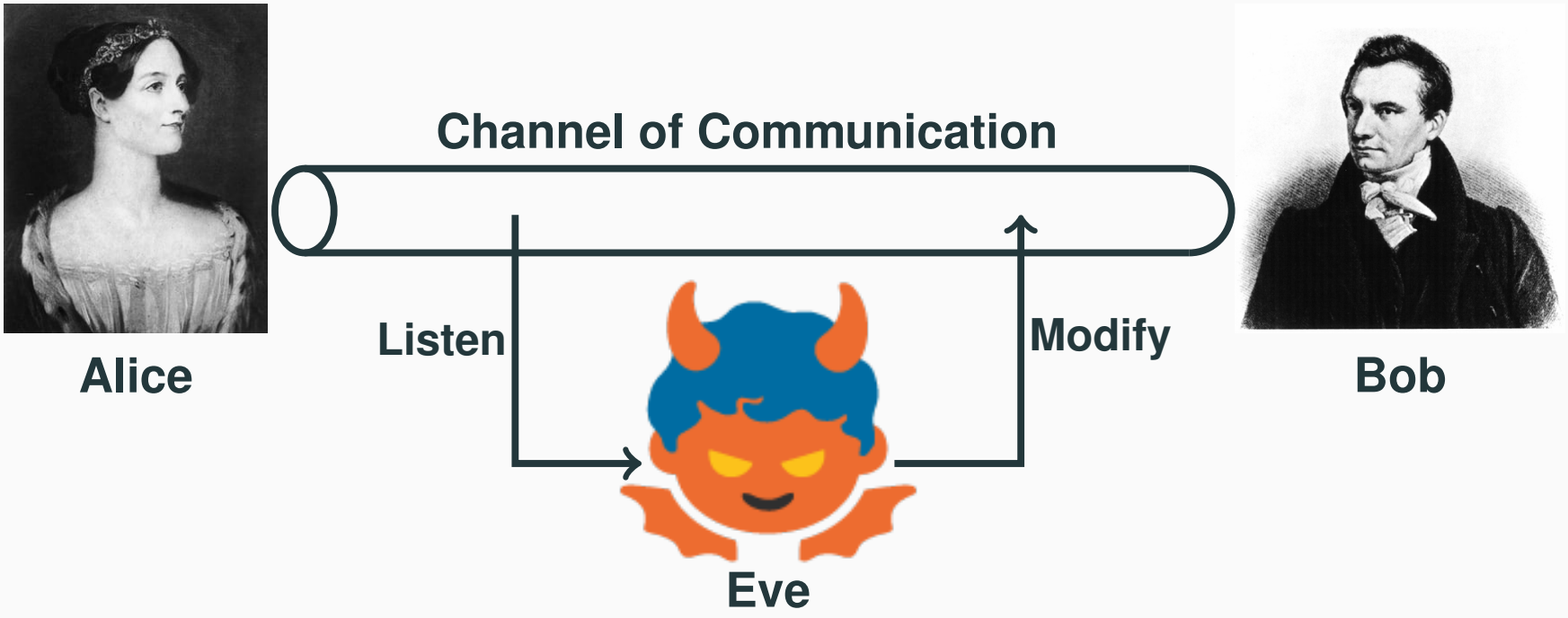
Cryptography is not everything: security needs a holistic approach.

- All components of the system needs to be *designed*, *configured*, and *used* securely:
  - Physical security
  - Network infrastructure
  - Hardware
  - Operating System
  - Middleware
  - Applications
  - Business logic
  - And last but not least, humans!

We discuss some of these in the 2nd half of the module.

Basic Setting

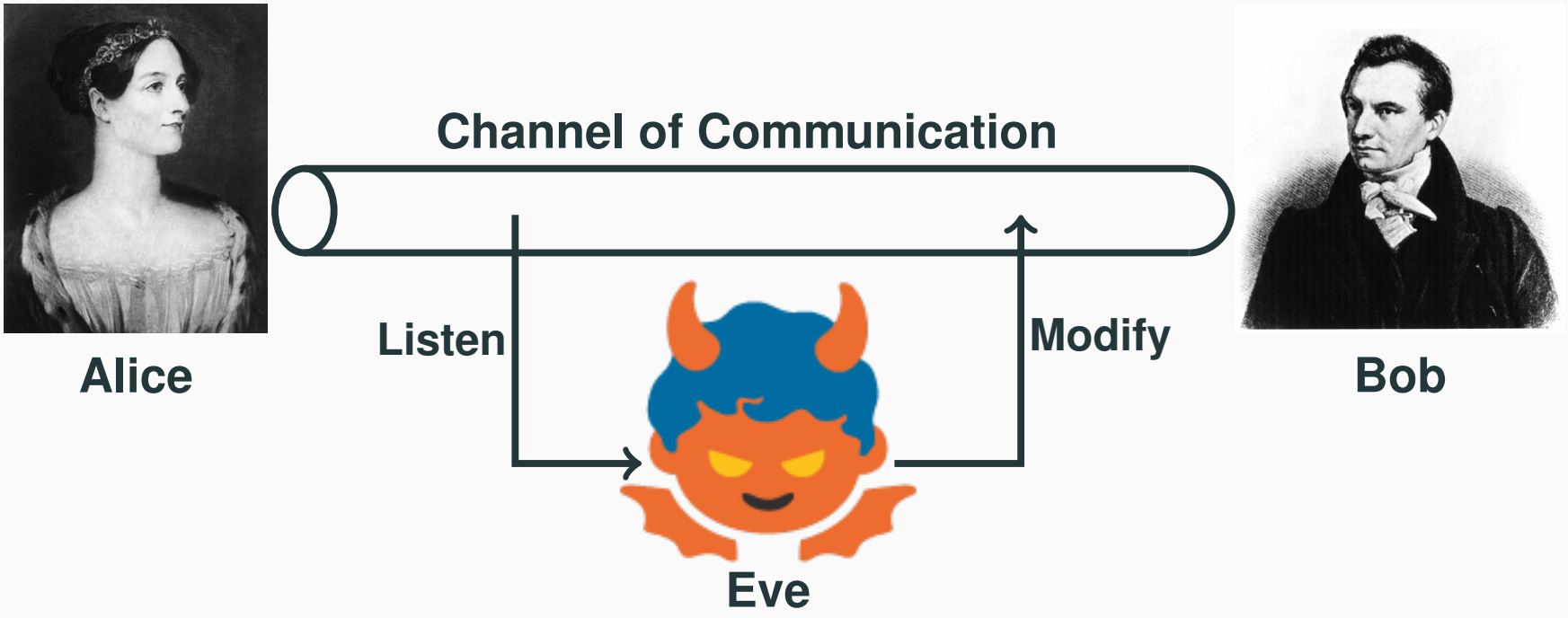
Basic setting of a cryptosystem:



Other possible settings: *broadcast* (multiple recipients), *file storage* (sender and recipients the same entity), . . .

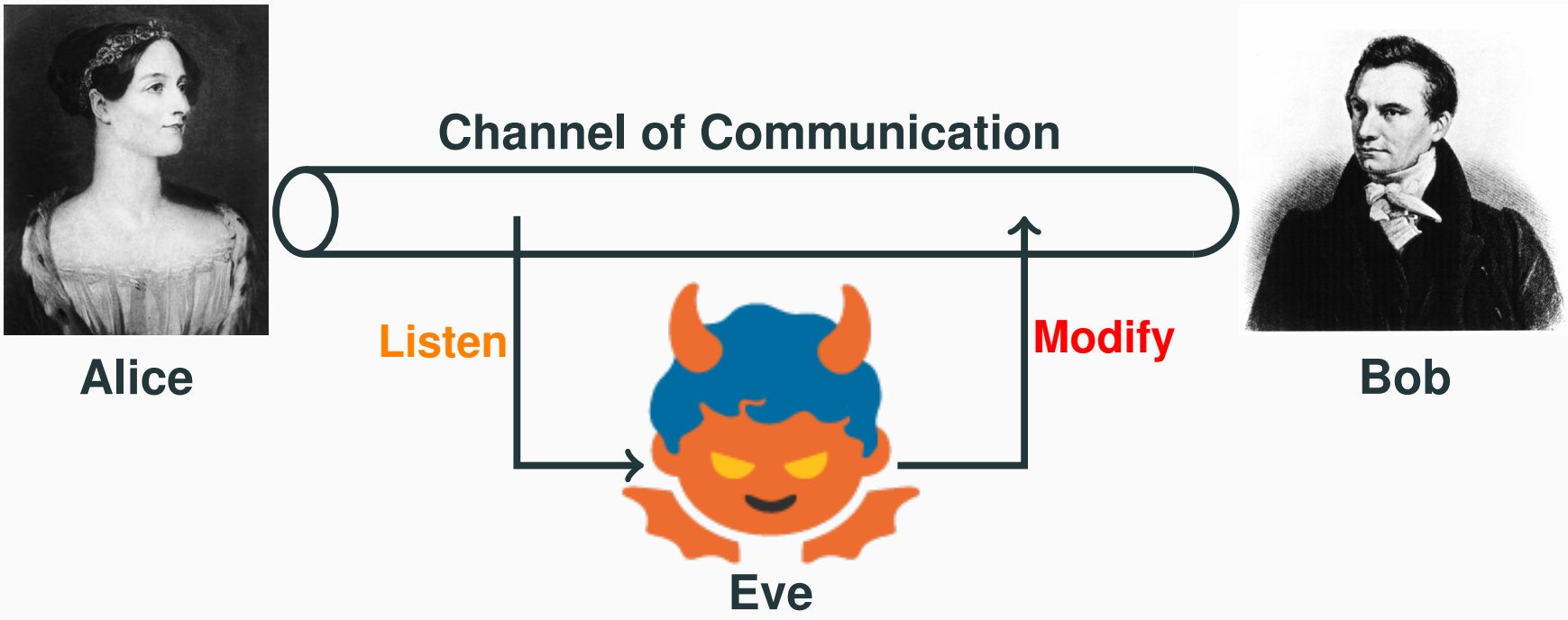
Basic Setting

Basic setting of a cryptosystem:



- Note that *entities* (or *users*) can be humans, computers, devices, etc.; whoever or whatever is taking part in the processing of data.

Basic Setting



Passive Setting: unauthorized access to data (eavesdropping, copying restricted files, etc.).

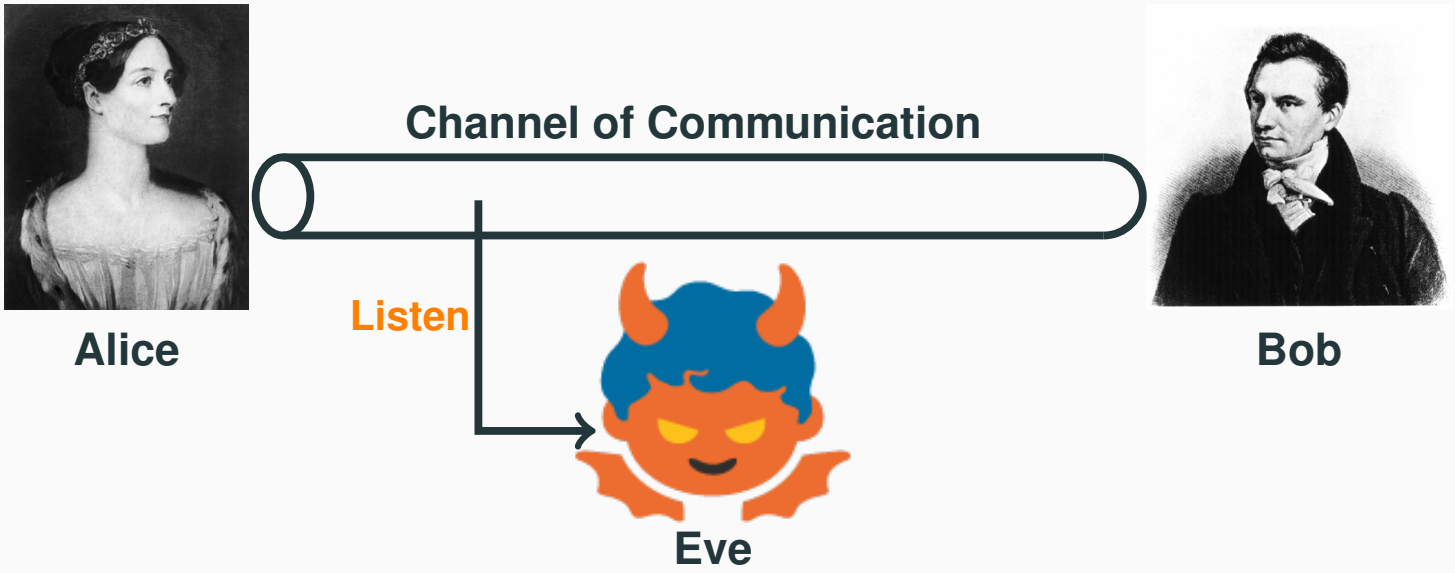
Active Attack: unauthorised alteration, deletion, transmission or access prevention to data.

Security Services

- A *security service* is a specific security goal.

Confidentiality (aka. “secrecy”) the assurance that data cannot be viewed by an unauthorised user.

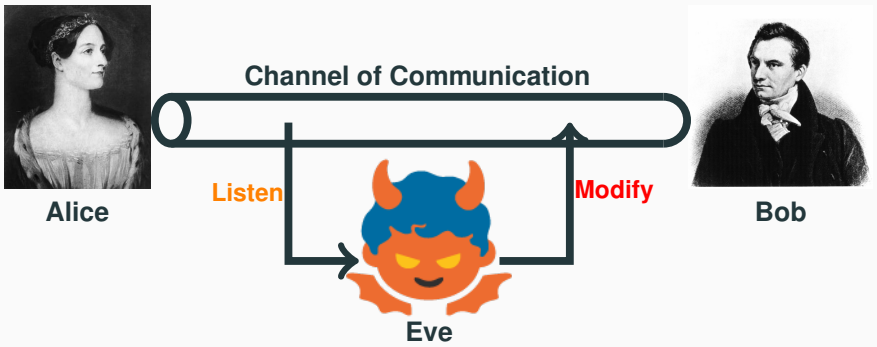
- this is the classic security service, which we also consider first.



Security Services

Data integrity the assurance that data has not been altered in an unauthorised manner (including accidental errors).

- ▷ This assurance applies from the time the data was last created, transmitted, or stored by an authorised user.
- ▷ Data integrity is not concerned with "preventing" alteration of data, but provides a means for "detection" of such.



Security Services

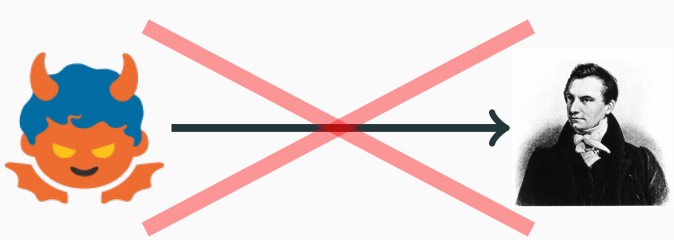
Data origin authentication assurance that a given entity was the original source of received data.

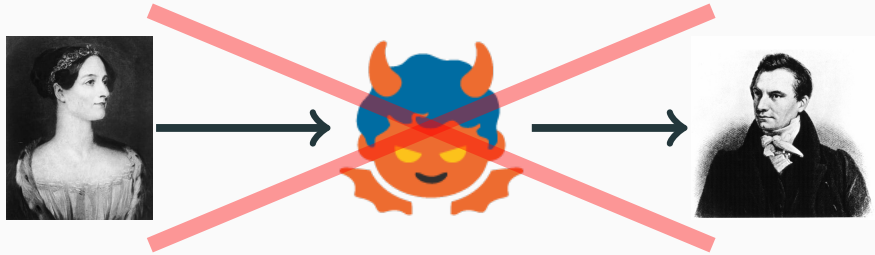
- data origin authentication is sometimes referred to as “message authentication” since it is concerned with the authentication of the data (message) and not who we are communicating with at the time the data is received.
- the time of creation or the immediate source of data is not of concern.

Security Services

Entity authentication the assurance that a given entity is involved and currently active in a communication session.

- ▷ entity authentication can also be referred to as “identification” because it is concerned with determining: who am I communicating with now, in real time?
- ▷ It is, e.g., detection (and not prevention!) of the following:

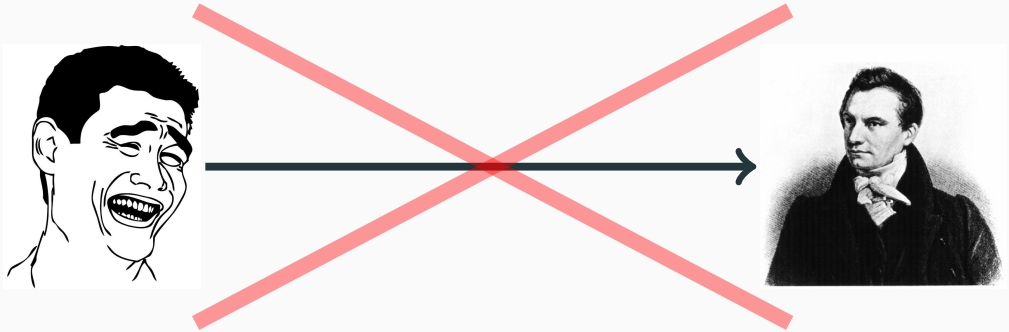




Security Services

Non-repudiation the assurance that an entity cannot deny a previous commitment or action (to a "third party")

- ▷ desirable in situations where there is the potential for a dispute over the exchange of data.



Relationships between security services

- *Data Origin Authentication is a Stronger Notion than Data Integrity*
  - data origin authentication requires data integrity
  - this is just a logical statement: if we cannot be sure if data has been tampered with or not, how can we ascertain the identity of its creator?
  - data origin authentication is data integrity with the extra property of assurance of the identity of the original source of the data.

Relationships between security services

- *Non-Repudiation of a Source is a Stronger Notion than Data Origin Authentication*
  - we can only bind the source to the data, in a manner that cannot be later denied, if we have assurance that the data itself is from that source.
  - hence, non-repudiation requires data origin authentication (and therefore, in turn, data integrity)

Relationships between security services

- *Data Origin Authentication and Entity Authentication are Different*
  - for instance, regarding an email, we need data origin authentication and entity authentication is rather meaningless;
  - on the other hand, before “authorizing” an entity to use a resource, we need entity authentication.
- *Data Origin Authentication Plus a Freshness Check can Provide Entity Authentication*
  - this is because we then are certain who the data originated from *and* that the originator is involved in the current communication session.

Relationships between security services

- *Confidentiality Does Not Imply Data Origin Authentication*
  - because for instance an adversary (Eve) can also re-arrange the encrypted messages of (Alice), or slightly modify them, so that they decrypt to a legitimate message, which is obviously not originated (created) by the sender (Alice);
  - we usually need both, but they should be explicitly and separately shown to be provided.

Basic cryptographic terminology

cryptography design and analysis of mechanisms based on mathematical techniques that provide fundamental security services.

- ▷ it comes from the Greek word `kryptos` meaning “hidden”, “concealed”, “secret”.

cryptographic primitive a cryptographic process that provides a number of specified security services.

- ▷ examples: block ciphers, stream ciphers, message authentication codes, hash functions, and digital signature schemes

Basic cryptographic terminology

cryptographic algorithm the particular specification of a cryptographic primitive (sufficiently detailed such that a programmer could implement it).

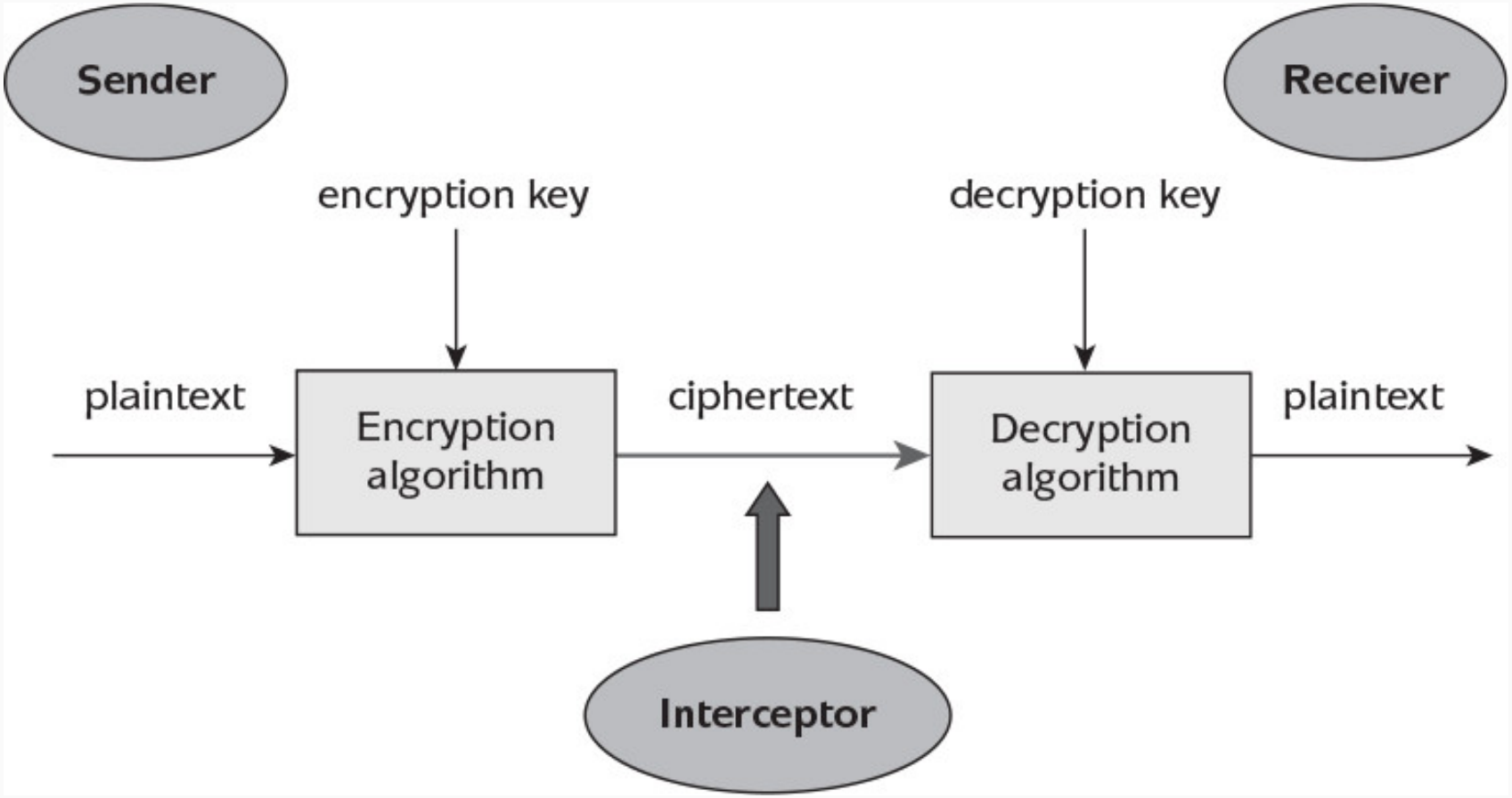
- ▷ e.g., AES is a cryptographic algorithm that specifies a block cipher

cryptographic protocol a sequence of message exchanges and operations between one or more parties, at the end of which a series of security goals should have been achieved

- ▷ e.g., SSL/TLS

cryptosystem (aka. “cipher system”) the implementation of some cryptographic primitives and their accompanying infrastructure

Basic model of a cryptosystem for confidentiality



Basic model of a cryptosystem for confidentiality

plaintext the raw data to be protected during transmission from sender to receiver.

- ▷ also said to be in the “clear”

ciphertext the scrambled version of the plaintext that results from applying the encryption algorithm (and the encryption key) to the plaintext.

- ▷ ciphertext is not a secret and can be obtained by anyone who “eavesdrops” the channel.
- ▷ no-one should be able to extract (compute) the plaintext from cipher-text without having the *decryption key*.

Basic model of a cryptosystem for confidentiality

encryption algorithm the set of rules that determines, for any given plaintext and encryption key, a ciphertext.

decryption algorithm the set of rules that determines, for any given ciphertext and decryption key, a unique plaintext.

encryption key a value that the sender inputs into the encryption algorithm along with the plaintext in order to compute the ciphertext.

decryption key is related to (but not always identical to) the encryption key. The receiver inputs the decryption key into the decryption algorithm along with the ciphertext to get the plaintext.

Basic model of a cryptosystem for confidentiality

keyspace the collection of all possible decryption keys

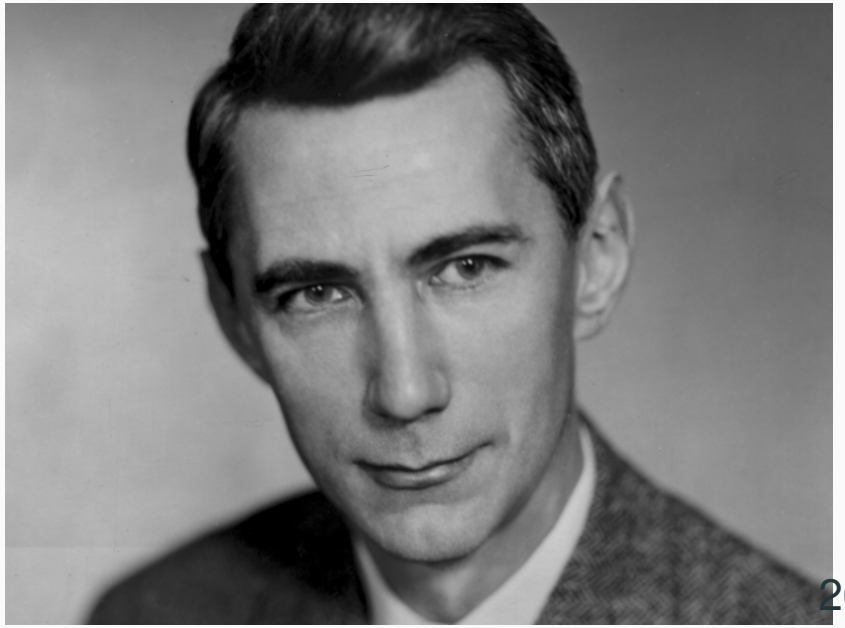
interceptor (aka “adversary” or “attacker”) an entity other than the sender or receiver who attempts to determine the plaintext.

- ▷ The interceptor will be able to see the ciphertext. The interceptor may know the decryption algorithm. The one piece of information the interceptor must not know is the decryption key.

(side note)

- encryption/decryption is an example of coding/decoding, esp. called cryptographic coding.
- other types of coding (each for a specific purpose):
  - “source-coding”: compression
  - “channel-coding”: error-correction
  - “line-coding”: modulation

Astonishingly, most of the fundamental ideas of all of the above (and much more) was developed by Claude C. Shannon (the father of “information theory”) during 40s (starting from his Masters thesis!).



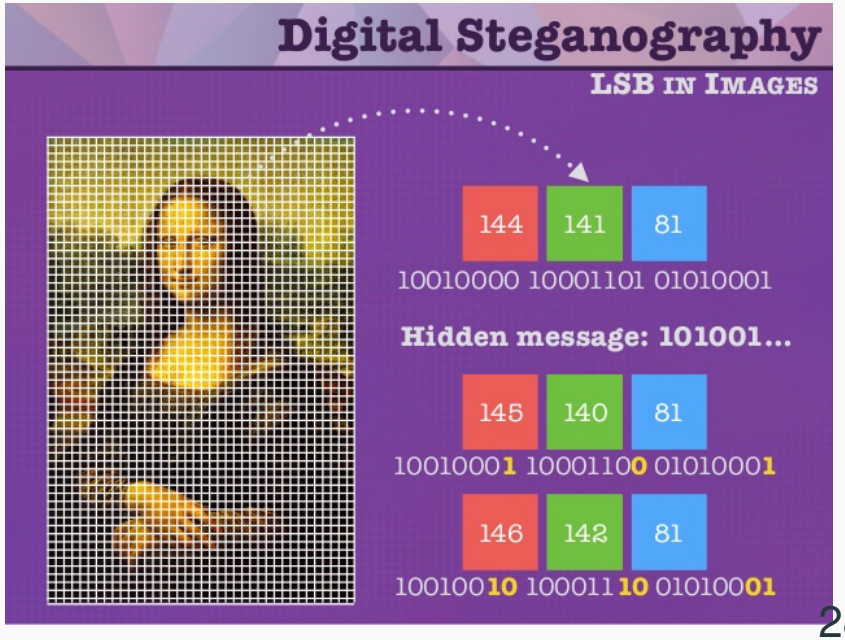
comparison to other mechanisms

- Encryption vs. Access Control (AC): hardware, OS and middleware mechanisms to prevent unauthorised users from accessing data. However, it is tied to the location the data resides, difficult to achieve when data is in “communication” (e.g. in cloud-computing).
  - AC’s confidentiality guarantee is solely based on enforcing lack of access to entire data, in contrast, in encryption, the adversary can see the entire ciphertext, and only does not have the decryption key.
  - the majority of our data protection is through AC, so we will return to this important topic in week 8.

comparison to other mechanisms

- Encryption vs. Steganography: steganography is about “information hiding”: plaintext is embedded in another non-secret data and the combined is sent; only the receiver knows that a hidden plaintext “exists” in the first place, and how to look for it.

  - in steganography, the adversary should not even notice that there is a plaintext at all, in contrast to encryption, where he knows there is a plaintext, but cannot compute it.



comparison to other mechanisms

(side-note)

- These mechanisms can of course be combined to provide “multiple layers” of security:
  - ▷ an adversary who gets around the access control to a computer/channel, if notices hidden information/traffic, will only find it in encrypted format.
  - ▷ however, steganography seems to be rarely used in practice seriously, because it does not provide any “guarantees”.
  - ▷ that said, for different security services, related techniques such as “digital watermarking” (to assert ownership of a data) and “honeypots” (to detect unauthorised access) are in frequent use.

Symmetric vs. Public-Key cryptosystem

symmetric (key) cryptosystems the encryption key and the decryption key are "essentially" the same.

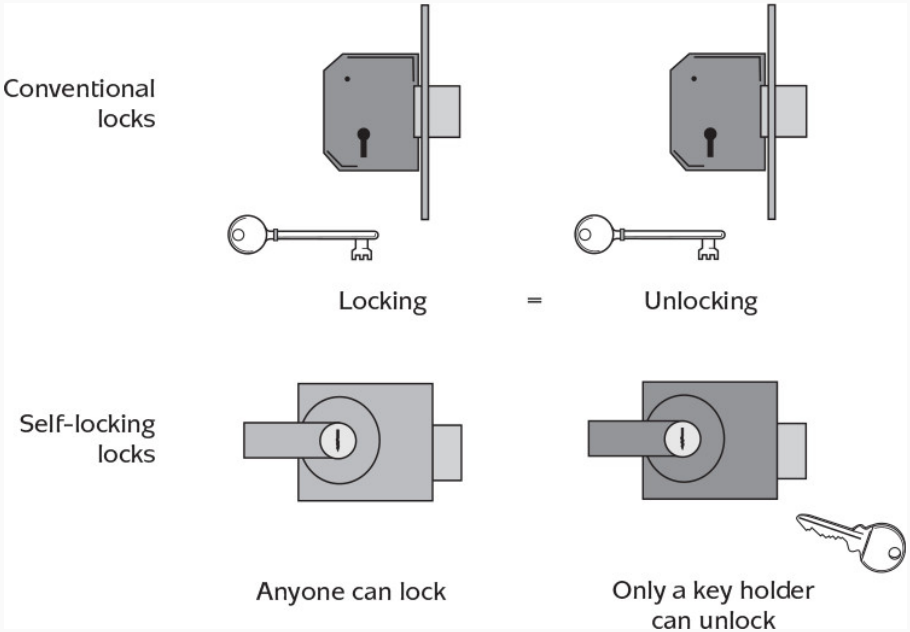
- ▷ even if they are not the same, they can be readily derived from each other.

public-key cryptosystems the encryption key and the decryption key are fundamentally different.

- ▷ $\Rightarrow$ aka. "*asymmetric* cryptosystems"
- ▷ it is computationally infeasible to determine the decryption key from the encryption key (this is not an intuitive property, in fact, quite amazing!).

Symmetric vs. Public-Key cryptosystem

- in both symmetric and public-key cryptosystems, the *decryption* key should be kept secret (from adversary)
- in symmetric-key crypto, the sender also knows the decryption key (but not adversary), whereas in public-key crypto, *everyone* can know the encryption key (hence the name: public key), incl. the adversary!



Symmetric vs. Public-Key cryptosystem

- in both symmetric and public-key crypto, the encryption and decryption are closely tied (so that only each can “neutralize” the effect of the other).
- in symmetric crypto, the decryption and encryption keys are essentially the same, while in public-key, it is (computationally) impossible to derive the decryption key from the encryption key!
  - this is thanks to the magic of mathematics! we will see example of this (weirdness) in subsequent weeks.

Making the encryption algorithm known

- It may be tempting to keep the detail of the encryption/decryption “algorithm” secret (on top of the decryption key). However:
  - a device implementing the algorithm can be “reverse engineered” to extract the algorithm;
  - the details of the algorithm may be accidentally or deliberately leaked to the public domain.
- **Kerckhoff’s principle**: the cryptographic algorithm should not be required to be secret (it should stay secure even if the detail of the algorithm is revealed).
  - Essentially, all the “hidden” stuff should be concentrated in the decryption key.

Making the encryption algorithm known

The case for using publicly known algorithm (as opposed to *proprietary* algorithms):

- *Scrutiny:* public algorithms are studied by a wide range of experts and perhaps adopted by public standardisation bodies;
- *Interoperability:* much easier to adopt and implement publicly known algorithms in open networks (esp., when requiring secure communications with external clients, perhaps in the future);
- *Transparency:* easier to convince a trading partner that their systems are secure if the security techniques they employ are open to assessment by their partners.

Cryptosystem security assumptions

the attacker (eavesdropper, adversary) knows:

1. *all transmitted ciphertexts*
2. *some corresponding plaintext-ciphertext pairs*, due to
   - ▷ receiver's failure to keep decrypted ciphertexts secret;
   - ▷ some predictable plaintexts (e.g. document headers);
   - ▷ the attacker has influenced the choice of plaintexts encrypted by the sender;.
   - ▷ the attacker has (temporary) access to either the encryption or decryption device (its interface);
   - ▷ in case of a public-key cryptosystem, the encryption key is known to any attacker, so they can generate as many plaintext-ciphertext pairs as they wish.
3. *the details of the encryption algorithm*

Cryptosystem security assumptions

(side-note)
some terminology of cryptographic attacks:

ciphertext-only attacks attacker only knows the encryption algorithm and some ciphertext;

known-plaintext attacks $\sim$ and some arbitrary plaintext/ciphertext pairs;

chosen-plaintext attacks $\sim$ and some plaintext/ciphertext pairs that correspond to plaintexts chosen by the attacker.

Note that these are increasingly powerful attacks (why?)

Breaking encryption algorithms

Breaking an encryption algorithm: a method of determining the plaintext from the ciphertext that does not involve being legitimately given the decryption key.

1. determining the decryption key directly (the most powerful type of break, since it allows decryption of all other ciphertexts corresponding to that key);
2. deducing a plaintext from the corresponding ciphertext without first determining the decryption key.

Exhaustive key searches

Exhaustive key search can be used to break almost all encryption algorithms (with a notable exception to come!).

- This attack, also known as *a brute-force attack* provides a ‘benchmark’ against which the effectiveness of other attacks can be measured.
- It simply involves *decrypting the ciphertext with different decryption keys (from the “key-space”) until candidates for the correct decryption key are found.*

Exhaustive key searches

The candidate decryption keys can be identified from:

- *some known plaintext/ciphertext pairs:* if a decryption key successfully decrypts the known ciphertexts into the corresponding known plaintexts, it's a candidate!
- *knowledge of the plaintext language:* if the plaintext is in a known language, such as English, then the attacker will be able to use the *statistical properties of the language* to recognise candidate plaintexts, and hence candidate decryption keys.
- *contextual information:* e.g. specific strings/characters or format in the plaintext.

some generic cryptographic attacks

(side note)

- **Dictionary attacks** involve compiling a type of ‘dictionary’, e.g.:
  - in a cryptosystem with a fixed key, the dictionary may consist of ciphertexts corresponding to plaintexts the attacker has been able to learn by some means.
  - in a cryptosystem where keys are “derived” from passwords, the attacker compiles a dictionary of likely passwords and then derives the resulting keys from them, used for a (more intelligent than brute-force) exhaustive key search.
- **Time memory trade-off** combining a pre-computed table (dictionary) and exhaustive search.

some generic cryptographic attacks

(side note)

- **Side-channel attacks** not directed against the theoretical design of a cryptographic primitive, but rather on its implementation; e.g.:
  - *timing attacks:* different processor computations may take slightly different “times” to compute depending on the value of the key.
  - *power analysis:* different processor computations may take slightly different “electric power” to compute depending on the value of the key.
  - *fault analysis:* inducing errors in a cryptosystem and studying the resulting output for useful information.
  - *padding attacks:* manipulating the “padding” process and monitoring the resulting error messages.

Historical Cryptosystems (and why they are all useless!)

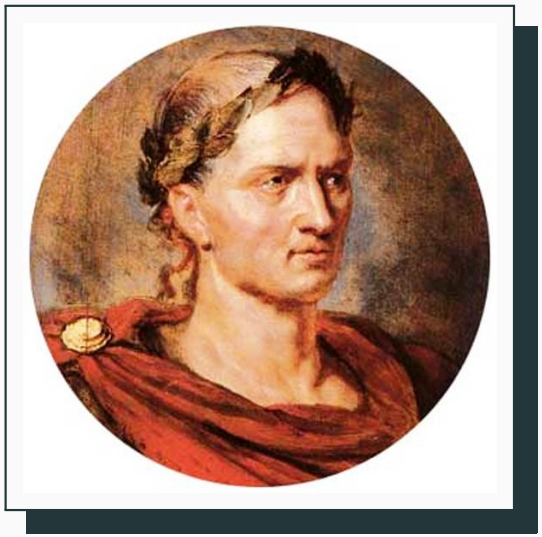
Historical Cryptosystems

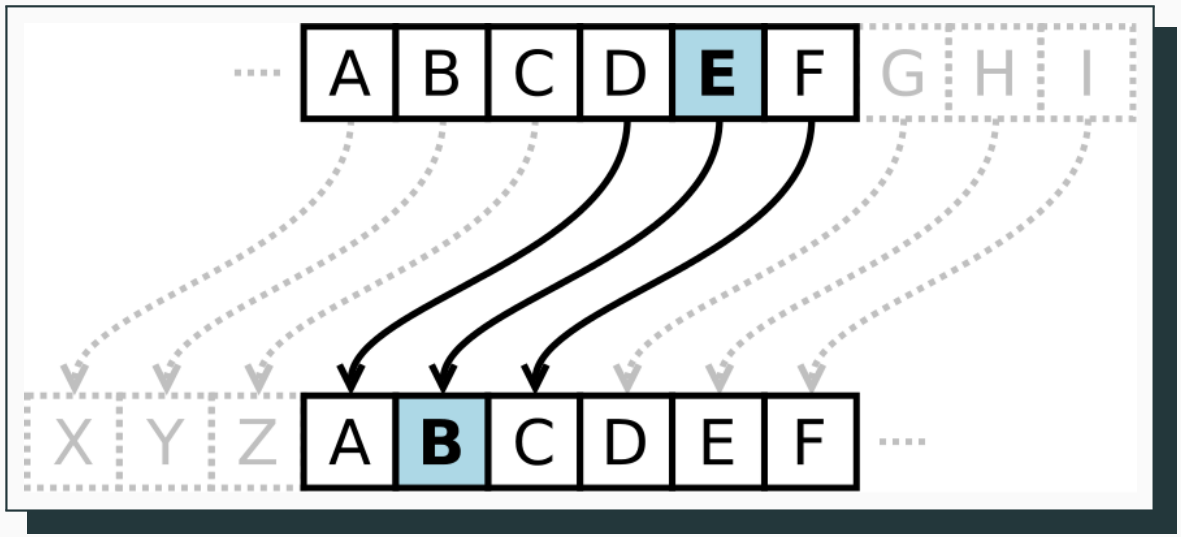
They are all:

- *symmetric* cryptosystems (predate the discovery of public-key cryptography)
- designed to provide *confidentiality* only.
- described as operating on *alphabetic* characters (in contrast to modern cryptosystems, which generally operate on binary numbers
- each is *completely unsuitable* for use in modern cryptographic applications

So, we only study them to internalize some concepts and grasp intuition.

Caesar Cipher





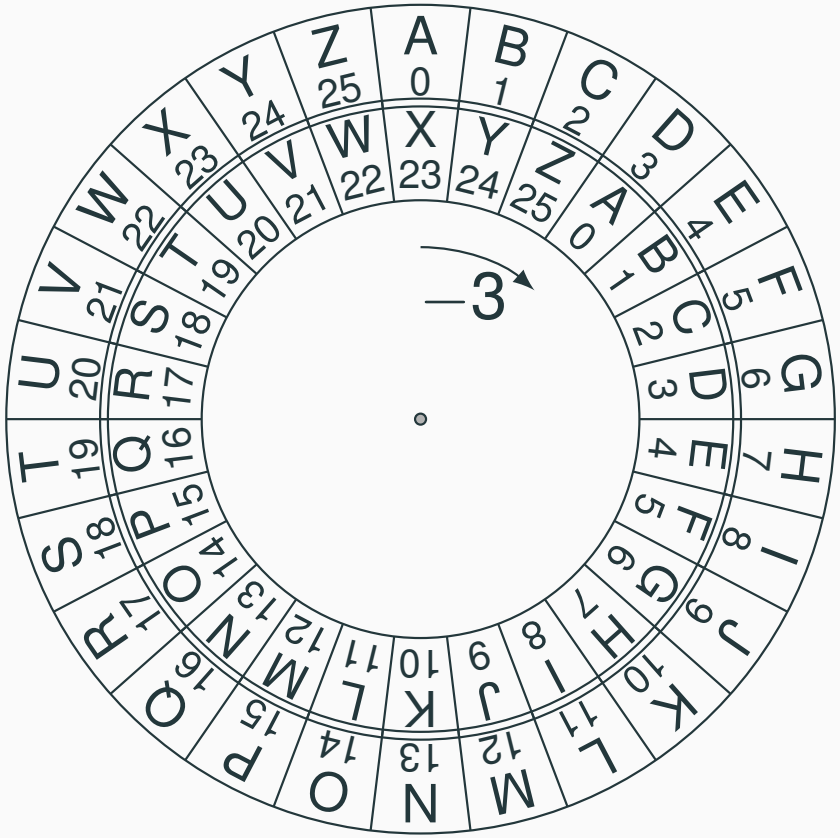
- Julius Caesar: (100 BC – 44 BC) used this method in his private correspondence $\Rightarrow$ Caesar (shift) cipher

  Plaintext : THE QUICK BROWN FOX JUMPS ...

  Ciphertext : QEB NRFZH YOLTK CLU GRJMP ...

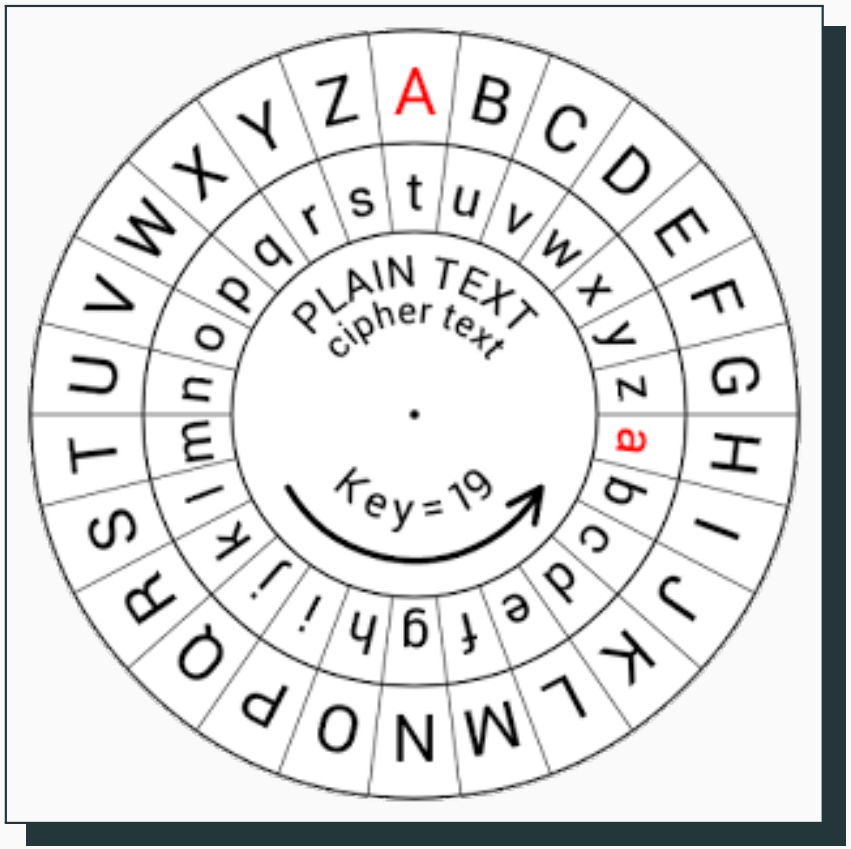
  Question: What is the "key" in the above example?

Caesar Cipher



plaintext	ciphertext
D (3)	*A* (0)
E (4)	*B* (1)
⋮	⋮
Z (25)	*W* (22)
A (0)	*X* (23)
B (1)	*Y* (24)
C (2)	*Z* (25)
D (3)	*A* (0)

Caesar Cipher



Math. representation:
Encryption:

$$C = P + K \bmod 26.$$

Decryption:

$$P = C - K \bmod 26.$$

Question: what is the biggest weakness of the Caesar Cipher?

Simple Substitution Cipher

- The main problem with Caesar cipher is its miniscule key-space size (of only 26). The *simple substitution* tries to resolve that:

A	B	C	D	E	F	G	H	I	J	K	L	M
D	I	Q	M	T	B	Z	S	Y	K	V	O	F
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
E	R	J	A	U	W	P	X	H	L	C	N	G

Simple Substitution Cipher

- **Question:** what is the key-space size of “simple substitution” cipher?

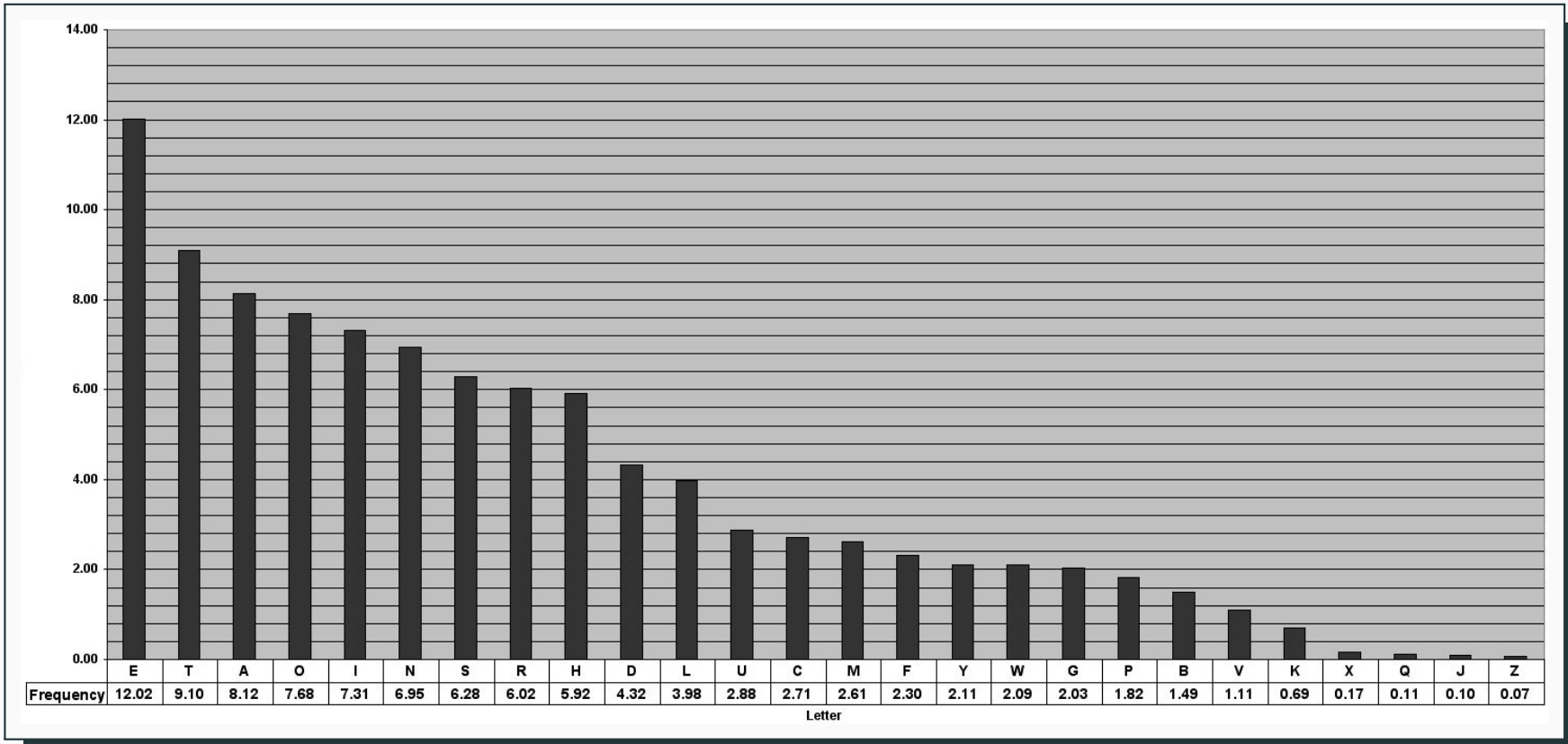
A	B	C	D	E	F	G	H	I	J	K	L	M
D	I	Q	M	T	B	Z	S	Y	K	V	O	F
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
E	R	J	A	U	W	P	X	H	L	C	N	G

Simple Substitution Cipher

- ▷ **Answer:** the key used in the Simple Substitution Cipher is a permutation of the letters of the alphabet.
- ▷ Hence, the key-space size is the number of possible permutations of 26 items, i.e., “26!” (26 factorial)
- ▷ but how big is “26!”? Answer: it is about 40000 the number of stars in our universe!
- ► **Question** But does this mean Simple Substitution Cipher is secure (i.e., provides confidentiality)?

Simple Substitution Cipher

- **Answer:** no! because of simple single-letter *frequency analysis*:



Simple Substitution Cipher

Insecurity of simple substitution cipher has an important general lesson:

- a large keyspace is necessary to make an exhaustive key search impractical to conduct, but it is not sufficient to guarantee the security of a cryptosystem.

Simple Substitution Cipher

Specifically for Simple Substitution Cipher, frequency analysis could so simply break it because it is a "mono-alphabetic" cipher. Drawing on that observation, we can think of the following ways to make statistical analysis harder:

- ▷ *increase the size of the plaintext and ciphertext alphabets* e.g. consider bigrams, trigrams, etc.
- ▷ *allow the same plaintext letter to be encrypted to different ciphertext letters*.
- ▷ *introduce positional dependency,* make the choice of ciphertext letter depend on the position the plaintext letter as well.

Vigenère Cipher

The key of the Vigenère Cipher consists of a string of letters that form a *keyword*.

1. write the keyword repeatedly underneath the plaintext
2. encrypt each plaintext letter using a Caesar Cipher, whose key is the keyword letter beneath it.

A	A	R	D	V	A	R	K	S	E	A	T	A	N	T	S
D	I	G	D	I	G	D	I	G	D	I	G	D	I	G	D
D	I	X	G	D	G	U	S	Y	H	I	Z	D	V	Z	V

Vigenère Cipher

Vigenère Cipher ensures plaintext letters encrypt to a variety of different ciphertext letters, and introduces positional dependency.

- **Question:** But is Vigenère Cipher secure?
- **Answer:** no, the Vigenère Cipher is easily broken! Can you see how?

- *Hint 1:* if the length of the keyword is known, it can be broken up into groups of Ceasar Ciphers and each broken using simple frequency analysis.
- *Hint 2:* the length of the keyword can be derived using other statistical properties of the English language (esp. its auto-correlation)

Vigenère Cipher

Insecurity of Vigenère Cipher, lessons learned:

- Vigenère Cipher tried to destroy simple statistical analysis of the cipher-text by introducing location-dependency: a letter of the plain-text can be mapped into different cipher-text letters, depending on the corresponding letter in the key. But it fails to kill the statistics of the plaintext completely.
- Choosing a longer keyphrase can make the cipher stronger (although still insecure), but at the cost of efficiency.

Historical Ciphers: Lessons learned

- Having a large key-space size is necessary but not sufficient for security.
- Attackers can utilise of the statistics of the plaintext language to “break” the cipher easier than “brute-force”.
- An encryption algorithm should disguise (destroying) the statistics of the plaintext language. In particular, it should avoid mono-alphabetic substitution, ensure plaintext letters encrypt to a variety of different ciphertext letters, and introduce “positional dependency”, however, none of these can guarantee security.

Historical Ciphers: Lessons learned

Destroying the statistics of the plaintext language efficiently is not easy. Two necessary properties for this are *confusion* and *diffusion* principles:

confusion each bit of the ciphertext should depend on several parts of the key;

diffusion changing a single bit of the plaintext should uniformly affect the entire ciphertext (i.e., half of its bits on average), and vice versa.

Questions?