# Applied Cryptography and Network Security CS 1653

Summer 2023

Sherif Khattab

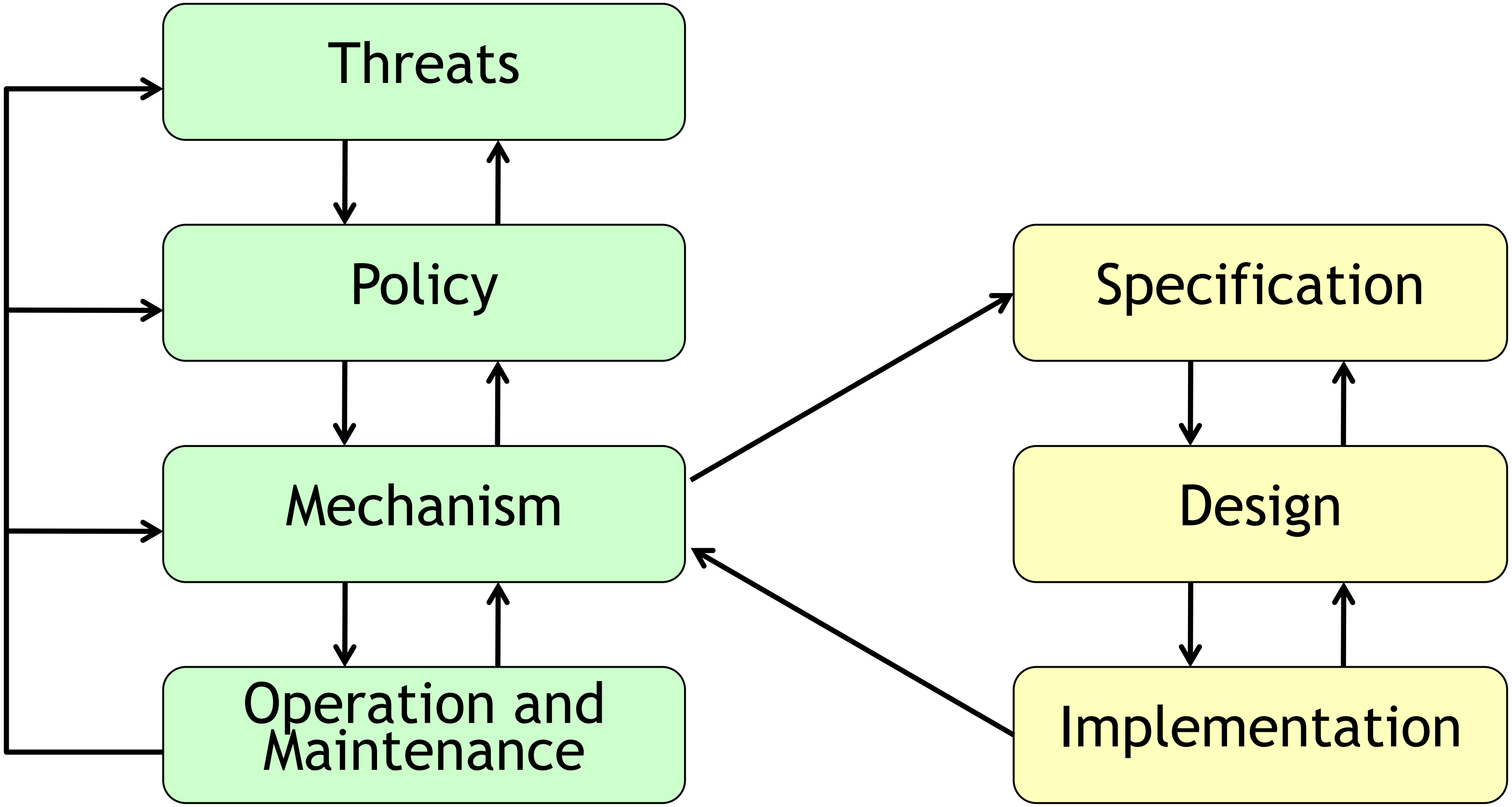
ksm73@pitt.edu

(Slides are adapted from Prof. Adam Lee's CS1653 slides.)

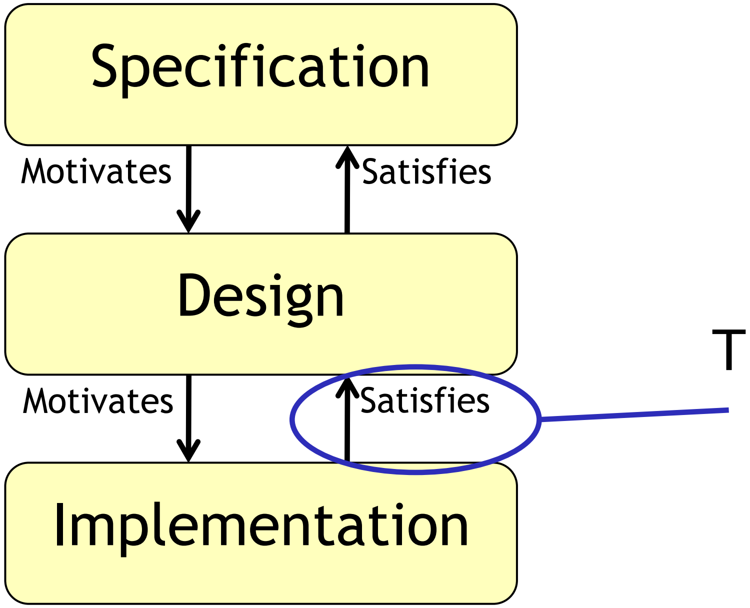
# Announcements

- Please skim over assigned reading
- Homework 1 due this Friday @ 11:59 pm
  - 3 attempts
- Based on the polls last lecture:
  - add 15 minutes to each lecture
    - updated lecture time: **1:30 pm – 3:30 pm**
    - **feel free to leave at 3:15 if you have to**
  - Makeup lectures
    - Friday 6/16 11:00-12:45
    - Friday 6/23 15:00-16:15
  - **lectures are recorded**
  - No Tophat questions during makeup time

# Security as a Process!



# Implementation



An implementation creates a functional system based on the design.

Ideally, we want to verify two things

- The design satisfies the specification
- The implementation satisfies the design and (by transitivity) the specification

This is harder than it sounds!

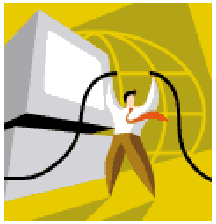
- Analyze correctness of each line of code
  - Preconditions
  - Postcondtions
- Proof of correctness depends on whether global preconditions to whole program hold
  - How do we specify these?
  - Are they correct?
  - ***Example:*** Bad compiler

# Testing-based approach as alternative to formal verification

Intuition: If a bunch of smart people can’t break into my system, I have at least some assurance that it is operating correctly

Many flavors of testing

- Regression testing/unit testing
- Red teams/penetration testing
- Fuzz testing
- …



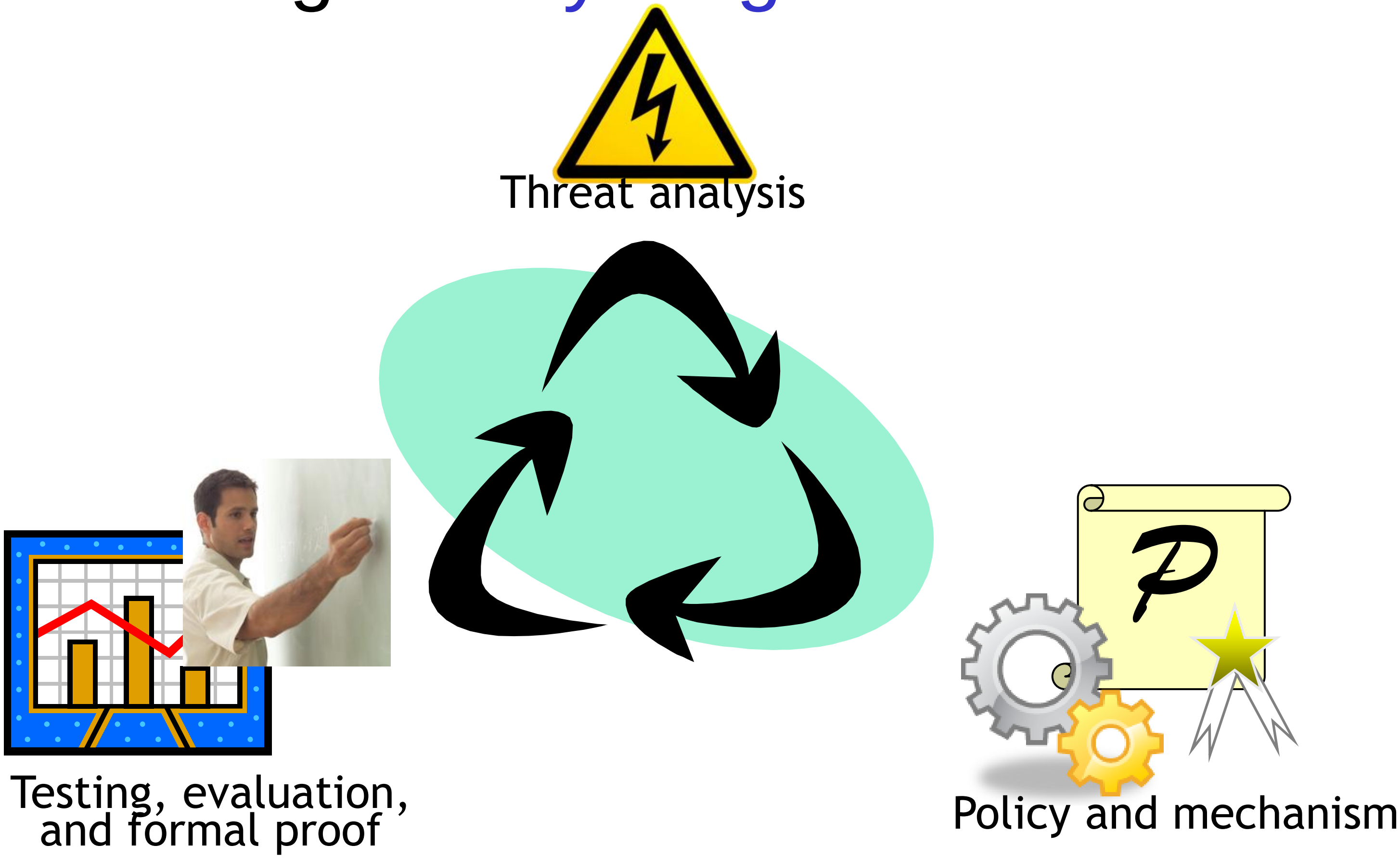
Testing does not provide you with a proven guarantee, but can uncover weaknesses or errors in a system

In the end, successfully resisting a rigorous attempt at intrusion is a good sign that things are on the right track

# Systems are dynamic, so are threat models and security policies

Changing environment means changing assumptions

This can change everything else…



Threat analysis

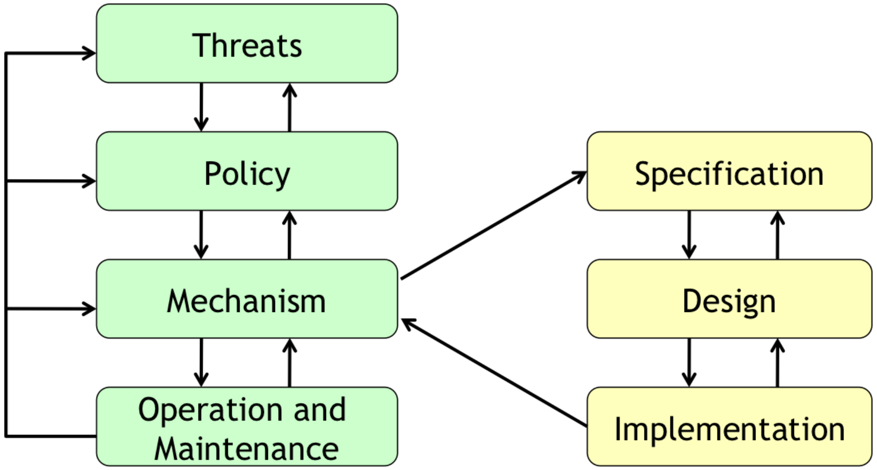
Testing, evaluation, and formal proof

Policy and mechanism

# Conclusions

Computer security does not happen by accident!

Careful attention must be given to security considerations at all stages of the software development lifecycle

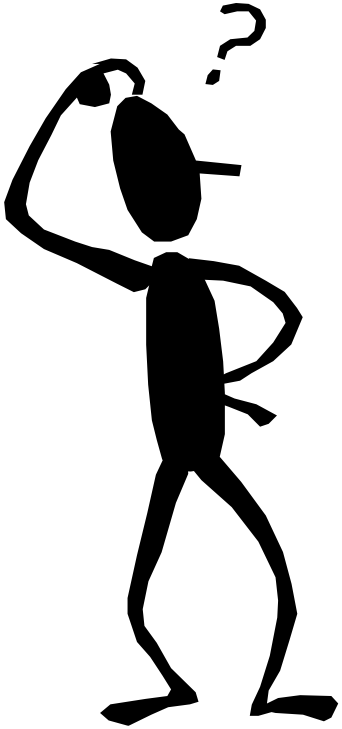


Best case scenario: Integrated process

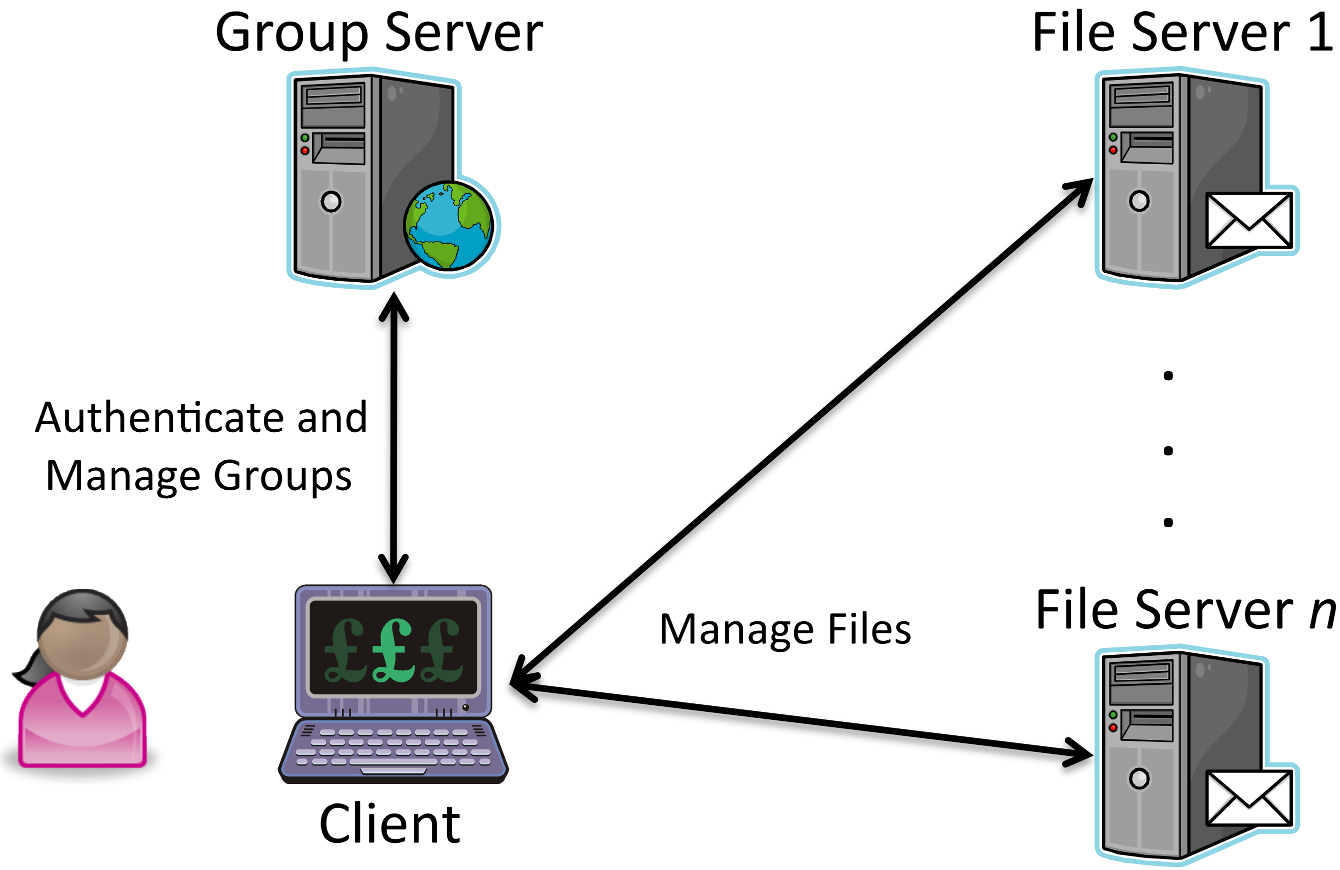
- Organizational risk analysis and cost/benefit analysis
- Saltzer and Schroeder’s design principles
- Formal verification and/or systematic testing

# Our introduction is now over

***Next:*** Introduction to classical cryptography



# Project Phase 1 will be out today!



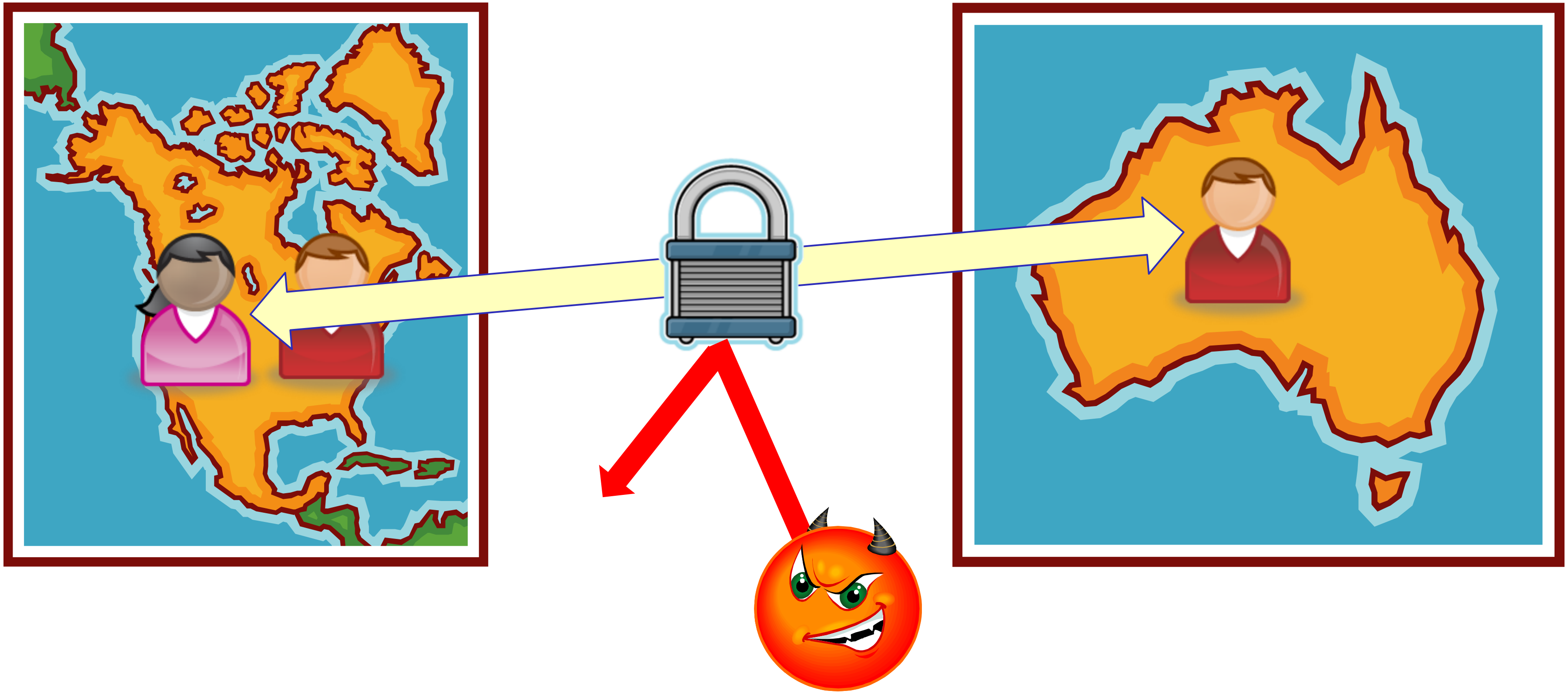
# Cryptography

# A Motivating Scenario

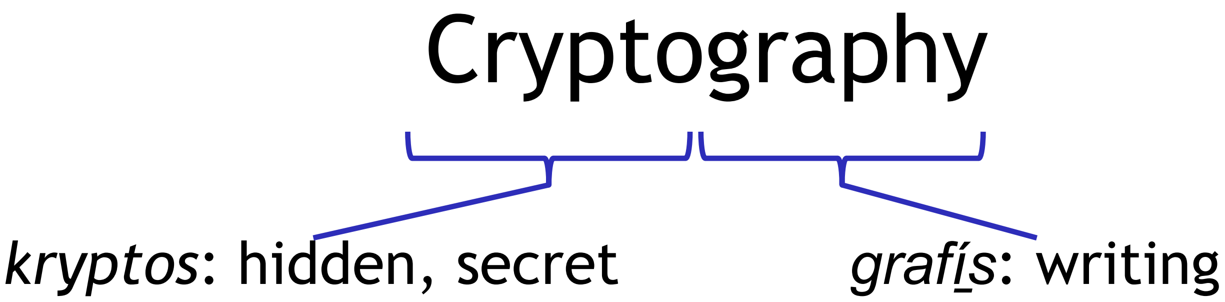
How can Alice and Bob communicate over an untrustworthy channel?

Need to ensure that:

1. Their conversations remain secret (confidentiality)
2. Modifications to any data sent can be detected (integrity)



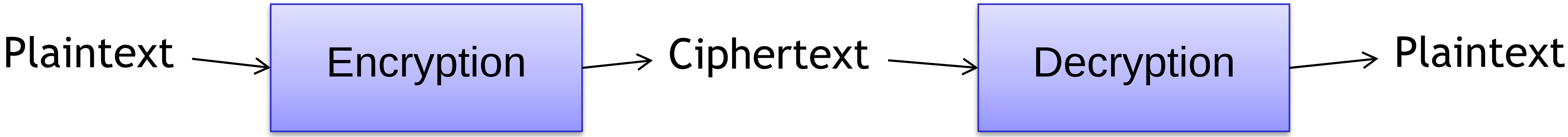
# What is cryptography?


Cryptography

*kryptos*: hidden, secret

*grafís*: writing

*Informally, cryptography is the study of methods for encoding and decoding secret messages*


Plaintext

Encryption

Ciphertext

Decryption

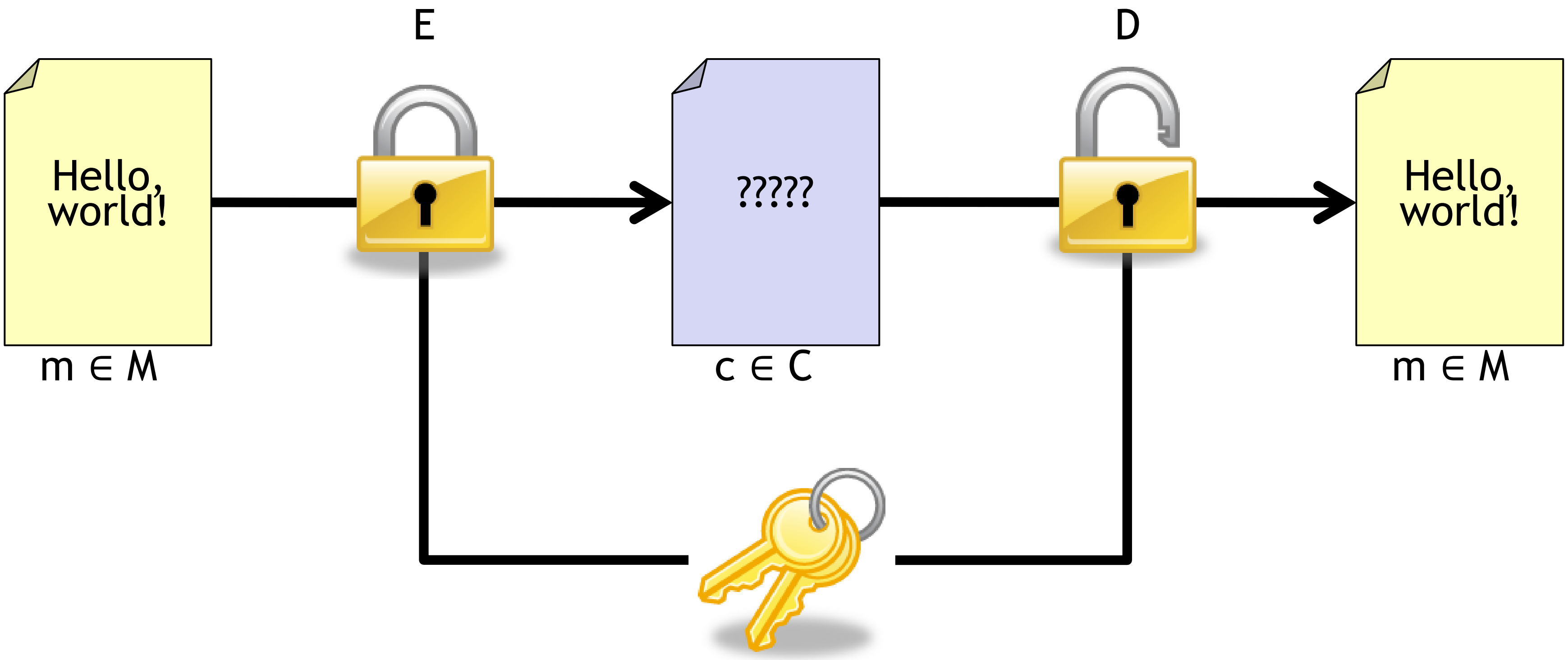
Plaintext

# More formally…

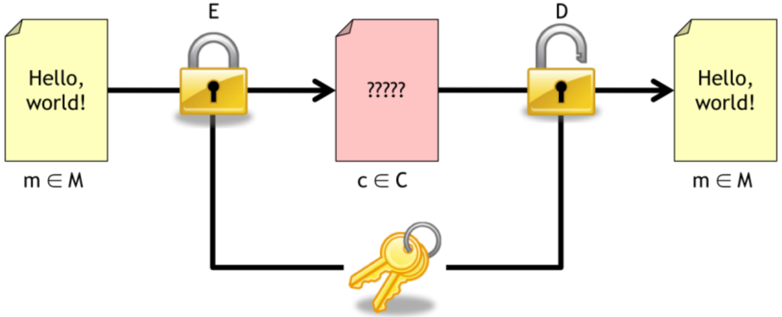
A cryptosystem can be represented as the 5-tuple

(E, D, M, C, K)

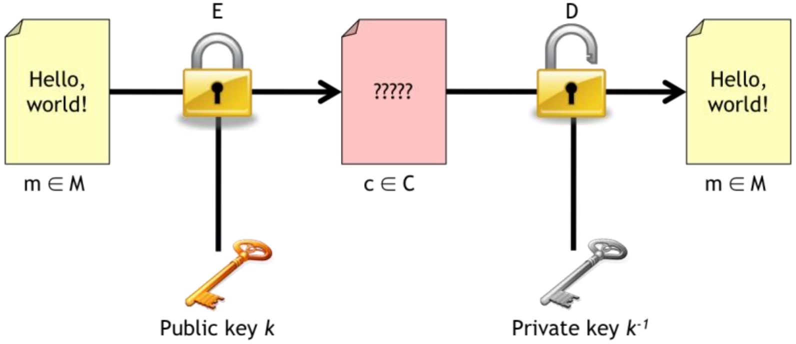
- M is a message space
- K is a key space
- C is a ciphertext space
- E is an encryption function; E: M × K → C
- D : C × K → M is a decryption function



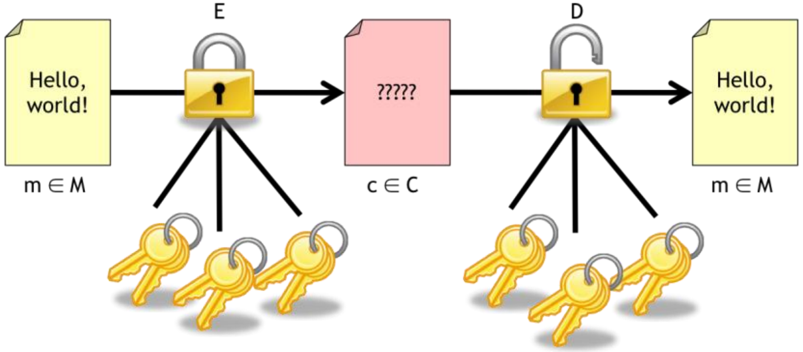
# This semester, we will learn about several classes of cryptosystems



Symmetric/secret key cryptography



Asymmetric/public key cryptography



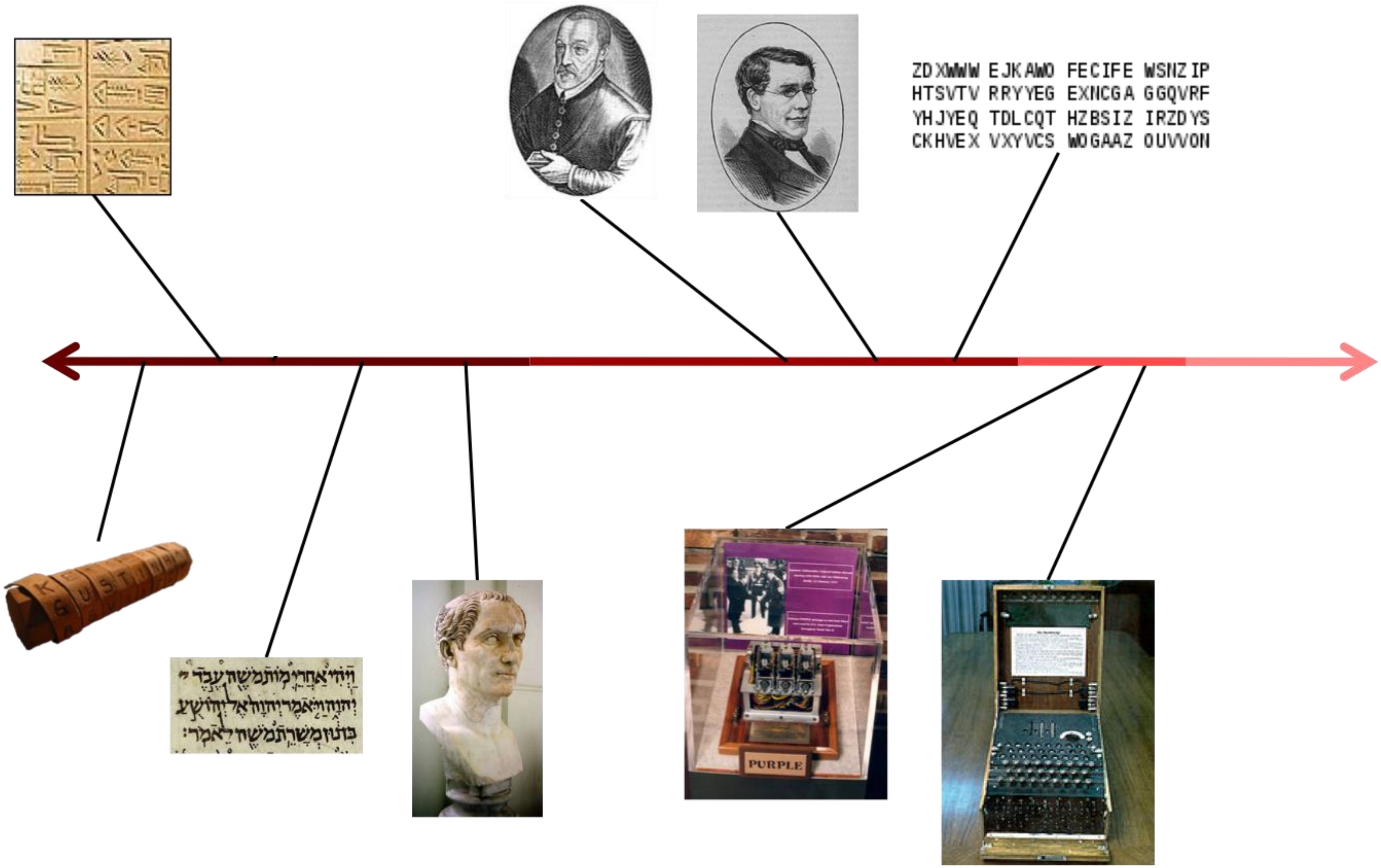
Threshold cryptography

*Cryptography is simply a tool. Different jobs require different tools.*

# Roadmap

Cryptography is far from being a new field

- This will be a history lesson of sorts
- Then we will focus on more recent developments


ZDXWWW EJKAWO FECIFE WSNZIP
HTSVTV RRYYEG EXNCGA GGQVRF
YHJYEQ TDLCQT HZBSIZ IRZDYS
CKHVEX VXYVCS WOGAAZ OUVVON
PURPLE

# Cryptography Timeline I

Steganography deals with more literally hiding secrets

Examples from antiquity include:

- Hidden writing underneath wax tablets
- Tattoos on scalps then grow hair

We won’t study steganography in this course, but it has been used frequently throughout history

Other examples:

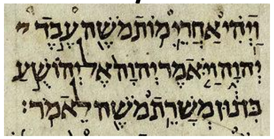
- Knots in knitting yarn, or patterns in clothes of couriers
- Invisible inks and other means of hiding text embedded in innocuous messages
- Messages embedded in JPEGs or packet delays
- …

Ancient times (BCE)

# Cryptography Timeline II

In addition to exploring steganography, ancient civilizations also developed a good number of cryptographic ciphers

While simple to break by today’s standards, these ciphers nicely illustrate the basics of many modern cryptosystems



ויהי אחרי מות משה עבד
יהוה ויאמר יהוה אל יהושע
בן נון משרת משה לאמר:

Examples include:

- Transposition ciphers
  - Scytale and rail-fence ciphers (7th century BCE)
- Substitution ciphers
  - Atbash cipher (500-600 BCE)
  - Caesar cipher (~100 BCE?)

# A scytale is an ancient cryptographic tool

A transposition cipher reorders (i.e., transposes) the characters in the plaintext message that is to be encrypted

A scytale is a tool that was used in antiquity to create a primitive transposition cipher

To encrypt:

- Wrap a strip of leather or paper around the scytale
- Write the message across the strip lengthwise
- Unwrap strip and write down the characters on a sheet of paper (Why?)

To decrypt:

- Write the characters onto a strip of leather or paper
- Wrap the message strip around an identical scytale
- Read across the length of the scytale

*Question: What is the key for this cipher?*

# The rail fence cipher is essentially a generalization of the cipher generated by using a scytale

To encrypt:

- Choose a number of "fence posts" (*f*)
- Write the message horizontally across the fence posts
- Transcribe the message by reading down each post

To decrypt:

- Set up *f* columns
- Write the first $\lceil$<message length> / *f*$\rceil$ down post 1
- …
- Write last block of characters down post *f*
- Read across posts

***Example:*** *f* = 3

$$\text{ATTACK} \rightarrow \begin{matrix}\text{ATT}\\ \text{ACK}\end{matrix} \rightarrow \text{AATCTK} \rightarrow \begin{matrix}\text{ATT}\\ \text{ACK}\end{matrix} \rightarrow \text{ATTACK}$$

# The Atbash cipher is an early substitution cipher

Substitution ciphers substitute one or more characters in the ciphertext alphabet for one or more characters in the plaintext alphabet

The Atbash cipher is an extremely simple substitution cipher that was used by ancient Hebrew scholars

Table:

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

***Example:*** ATTACK → ZGGZXP

*Question: What is the key for this cipher?*

# The Caesar cipher is another classical substitution cipher

The Caesar cipher is named after Julius Caesar, who used it to communicate with his generals

To encrypt:

Choose a shift index *s*

Apply the function e(x) = x + *s* mod 26

To decrypt:

Apply the function d(x) = x – *s* mod 26

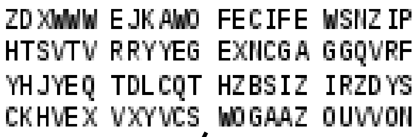
***Example:*** *s* = 3

Encryption: ATTACK → 0 19 19 0 2 10 → 3 22 22 3 5 13 → DWWDFN

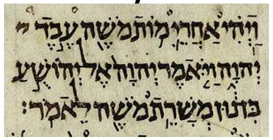
Decryption: DWWDFN→ 3 22 22 3 5 13 → 0 19 19 0 2 10 → ATTACK

# Cryptography Timeline III

“Medieval” cryptography further extended and improved upon the ideas developed in ancient times



```
ZDXWWW EJKAWO FECIFE WSNZIP
HTSVTV RRYYEG EXNCGA GGQVRF
YHJYEQ TDLCQT HZBSIZ IRZDYS
CKHVEX VXYVCS WOGAAZ OUVVON
```



Examples include:

- Vigenere cipher (1500s)
- Playfair cipher (1854)
- One-time pad (1917)

# The Vigenère cipher is an improved version of Caesar cipher

The Vigenère cipher was invented in 1553 by Giovan Battista Bellaso. It is misattributed to Blaise de Vigenère, who invented a stronger version of this cipher in the 19th century.

The Vigenere cipher basically uses a repeated key phrase to apply a different variant of the Caesar cipher to each plaintext letter.

Encryption and decryption can be aided by using a *tabula recta*

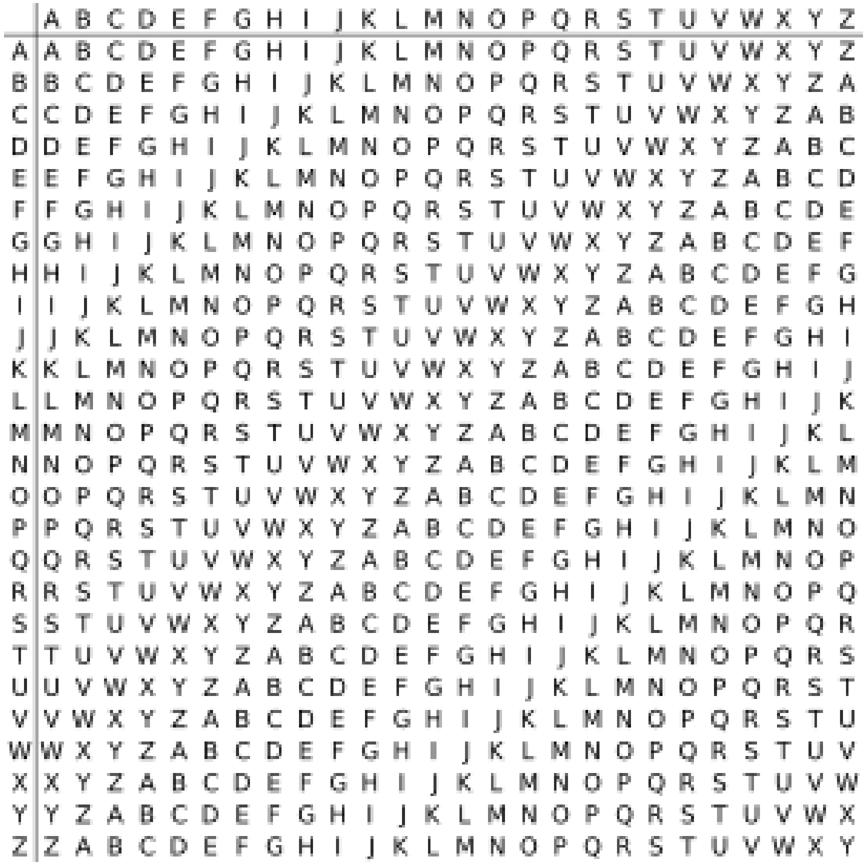
***Example:***

Plaintext: ATTACKATDAWN

Key: LEMONLEMONLE

Ciphertext: LXFOPVEFRNHR

All possible Caesar ciphers



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

# Similarity to Caesar cipher is more readily apparent when viewed mathematically

Encryption: $e(p_i, k_i) = p_i + k_i \bmod 26$

Decryption: $d(c_i, k_i) = c_i - k_i \bmod 26$

***Example:***

| | |
|---|---|
| Plaintext: | `ATTACKATDAWN → 00 19 19 00 02 10 …` |
| Key string: | `LEMONLEMONLE → 11 04 12 14 13 11 …` |
| Ciphertext: | `11 23 05 14 15 21 …` |

`LXFOPW…`

# One-time pad is a further generalization of Vigenère

The one-time pad was invented in the early 20th century as a telegraph cipher at AT&T, and saw many military and diplomatic uses

How does it work?

Choose a key that is as long as the plaintext that you wish to encrypt

$E(p) = p \oplus k$

$D(c) = c \oplus k$

***Example:*** Using modular addition/subtraction instead of XOR

| | |
|---|---|
| Plaintext: | `BUYTENSHARES` |
| Key: | `HIENVWKNUQCF` |
| Ciphertext: | `ICCGZJCUUHGX` |

Note: A single ciphertext can recover any plaintext string of the same length!

For example, with key `LGAEMVEKRKAH`, the text `TICKLEGEORGE` can be recovered from the above ciphertext!

# The Playfair cipher is a digraph substitution cipher

Digraph ciphers substitute pairs of letters—rather than single letters—for one another

The Playfair cipher was invented by Charles Wheatstone in 1854, but is named after Lord Lyon Playfair, who strongly promoted its use

*Wheatstone*

*Playfair*

To use this cipher, first build a 5x5 table using a secret keyphrase

- Write keyphrase left to right, top to bottom
- Skip any repeated letters
- Fill in any remaining letters (usually combining I/J, or skipping Q)

***Example:***

IHAVEADREAM →

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

# Encryption is carried out by following a few simple rules

1. Insert an X between any pair of repeated letters

```
MONEY IN TREE → MONEY IN TREXE
```

2. Break string to encrypt into pairs of letters

```
MONEY IN TREXE → MO NE YI NT RE XE
```

3. Encrypt digraphs using the table

   If letters are on the same row, use letters to the right

   If letters are in the same column, use letters below them

   If letters form the corners of a box, use the other corners

This is perhaps best illustrated with an example…

# Encrypting “MONEY IN TREE”

Apply rules 1 and 2: `MONEY IN TREE → MO NE YI NT RE XE`

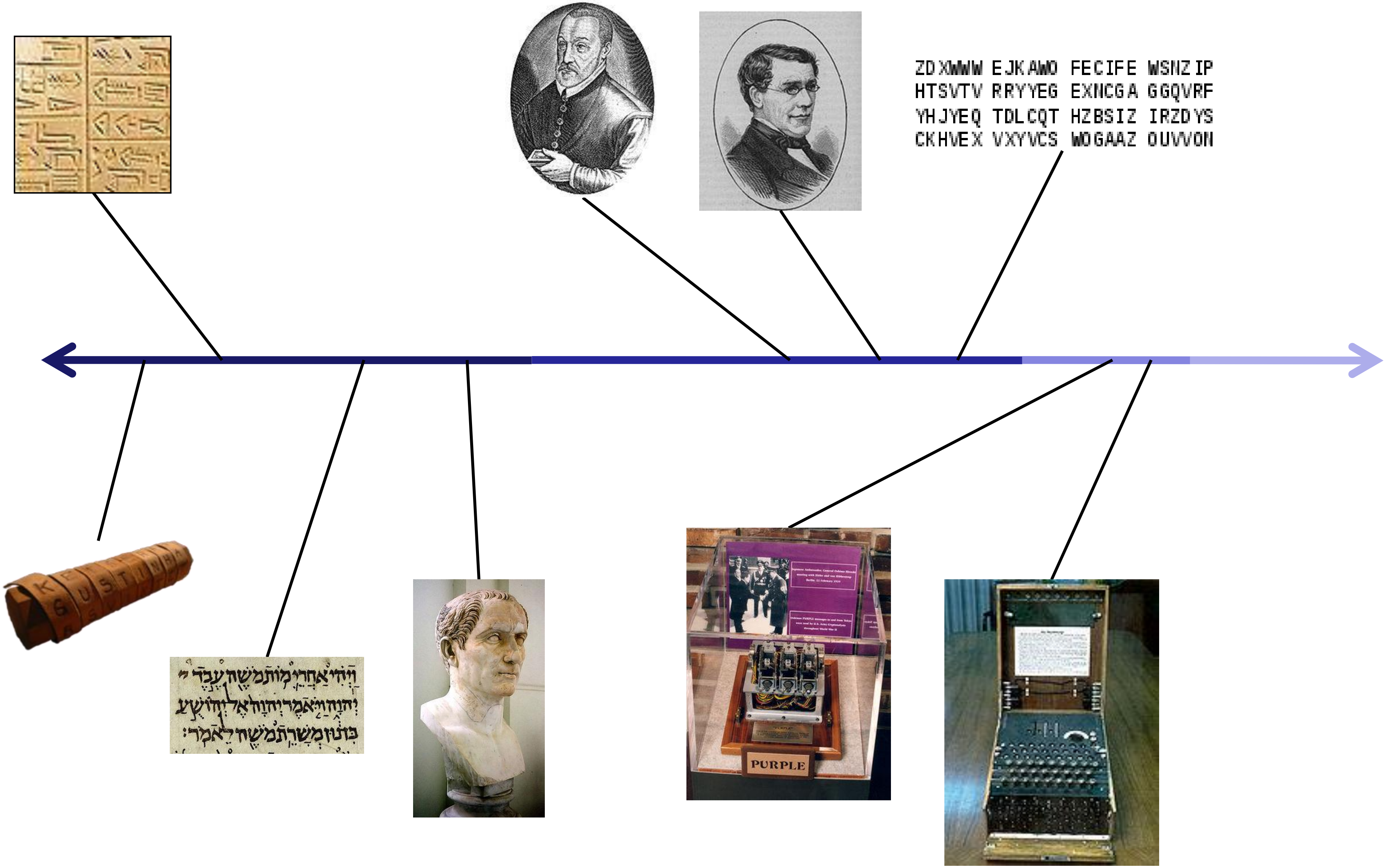
| | | | | |
|---|---|---|---|---|
| J | H | A | V | E |
| D | R | M | B | C |
| F | G | K | L | N |
| O | P | Q | S | T |
| U | W | X | Y | Z |

Now, we can use the 5x5 code box to encrypt each digraph:

`MO NE YI NT RE XE →` [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

To decrypt, simply apply the rules in reverse

# Cryptography Timeline IV



```
ZDXWWW EJKAWO FECIFE WSNZIP
HTSVTV RRYYEG EXNCGA GGQVRF
YHJYEQ TDLCQT HZBSIZ IRZDYS
CKHVEX VXYVCS WOGAAZ OUVVON
```

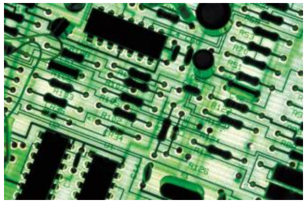
# Unfortunately, classical cryptosystems are easy to break

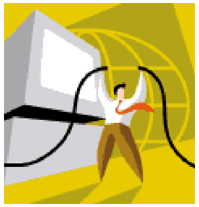
These (and other) algorithms were designed in a time when people instead of computers were trying to break ciphers, and few people used cryptography

What does this mean?

- Relatively small key spaces were OK
- Relying on an algorithm remaining secret was a less dubious assumption

Fast forward: 20th century





Widespread mechanical and digital computing device and pervasive communication invalidate the above assumptions

- Machines can systematically try all possible keys
- Security by obscurity breaks down

*How can we define and measure the security of a cryptosystem?*

# When studying cryptosystems, we are generally concerned with three primary classes of attack

In a ciphertext only attack, the adversary is assumed to have stored some amount of ciphertext that can be analyzed offline to attempt to break the cipher.

Hardest for the attacker

In a known plaintext attack, the adversary is assumed to have collected some number of (plaintext, ciphertext) pairs that can be used to guide his attempt at breaking the cipher.

In a chosen plaintext attack, the adversary has access to the cryptographic algorithm and may encrypt anything that he chooses. The resulting (plaintext, ciphertext) pairs are then used to guide attempts at breaking the cipher.

**Hardest to defend against**

# most classical cryptosystems can be broken by ciphertext-only attacks

Breaking the Caesar cipher is trivial

Only 26 possible keys to try

Alternately, frequency analysis can be used

The rail fence cipher is also easy to break

- We know that there are most likely less than <message length>/2 rails.  For most messages, a computer can break this very quickly.
- *n*-gram frequency analysis can also be used

# The Vigenère cipher also falls to this type of attack

Observation: The Vigenère cipher is really just *k* different Caesar ciphers, where *k* is the length of the key used

Therefore, breaking the Vigenère cipher is a two-step process:

1. Figure out *k*
2. Break each of the cipher's *k* Caesar ciphers

*Breaking a Caesar cipher is easy, so the "hard" part is figuring out the key length, k… How do we do this?*

# Kasiski examination is one method of finding the key length of a Vigenère cipher

Intuition: Given enough ciphertext, short words like “the” will appear encrypted the same way multiple times.

The distance between these repeated blocks of ciphertext is typically a multiple of the keyword length!

***Example:***

| | |
|---|---|
| Plaintext: | `CRYPTO IS SHORT FOR CRYPTOGRAPHY` |
| Key: | `ABCDAB CD ABCDA BCD ABCDABCDABCD` |
| Ciphertext: | `CSASTP KV SIQUT GQU CSASTPIUAQJB` |

Note that finding multiple repeated blocks of ciphertext will make this attack even easier (Why?)

# Known plaintext and chosen plaintext attacks simplify things even further…

Think about it:

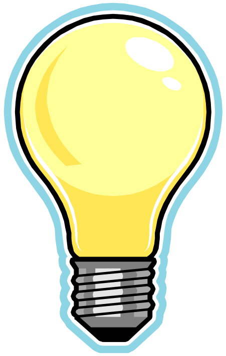
- **Caesar cipher:** A match between a single pair of plaintext and ciphertext characters reveals the shift index *s*
- **Rail fence cipher:** Given a long enough string of ciphertext corresponding to a known piece of plaintext, *f* can be learned
- **Vigenère cipher:** If the length of the matching plaintext and ciphertext strings exceeds the key length *k*, the key can be easily recovered

As a result, evaluating a cryptosystem typically involves making sure that it resists chosen plaintext attacks. As we will see later, even stronger attacker models are often assumed.

In short: Assume that your attacker knows as much as possible!

# Although the one-time pad offers protection against these attacks, it is not a panacea

## *Potential Pitfalls:*

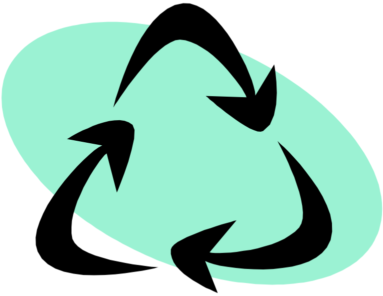


### *Predictable Key Pads*

If the key can be guessed with high probability, it is no good!

Note: The Allies broke a German version of the OTP this way during WWII

Can break implementation even if the algorithm has perfect security!



### *Key Reuse*

- Given $p_1 \oplus k$ and $p_2 \oplus k$, we can recover $p_1 \oplus p_2$
- If $p_1$ and $p_2$ have some predicable structure, this can leak information
- If we can choose one of the $p_i$'s, we can recover $k$

### *Key Distribution*

- We need as much key material as we have material to transmit!
- Inefficient if much communication is to occur…

# Moving Forward…

In the short term:

How do we build secure cryptosystems with short, manageable keys?

How can we develop an assurance that these cryptosystems are reasonably strong?

In the medium term:

Different tools for different jobs

Choosing the right tool

In the long term:

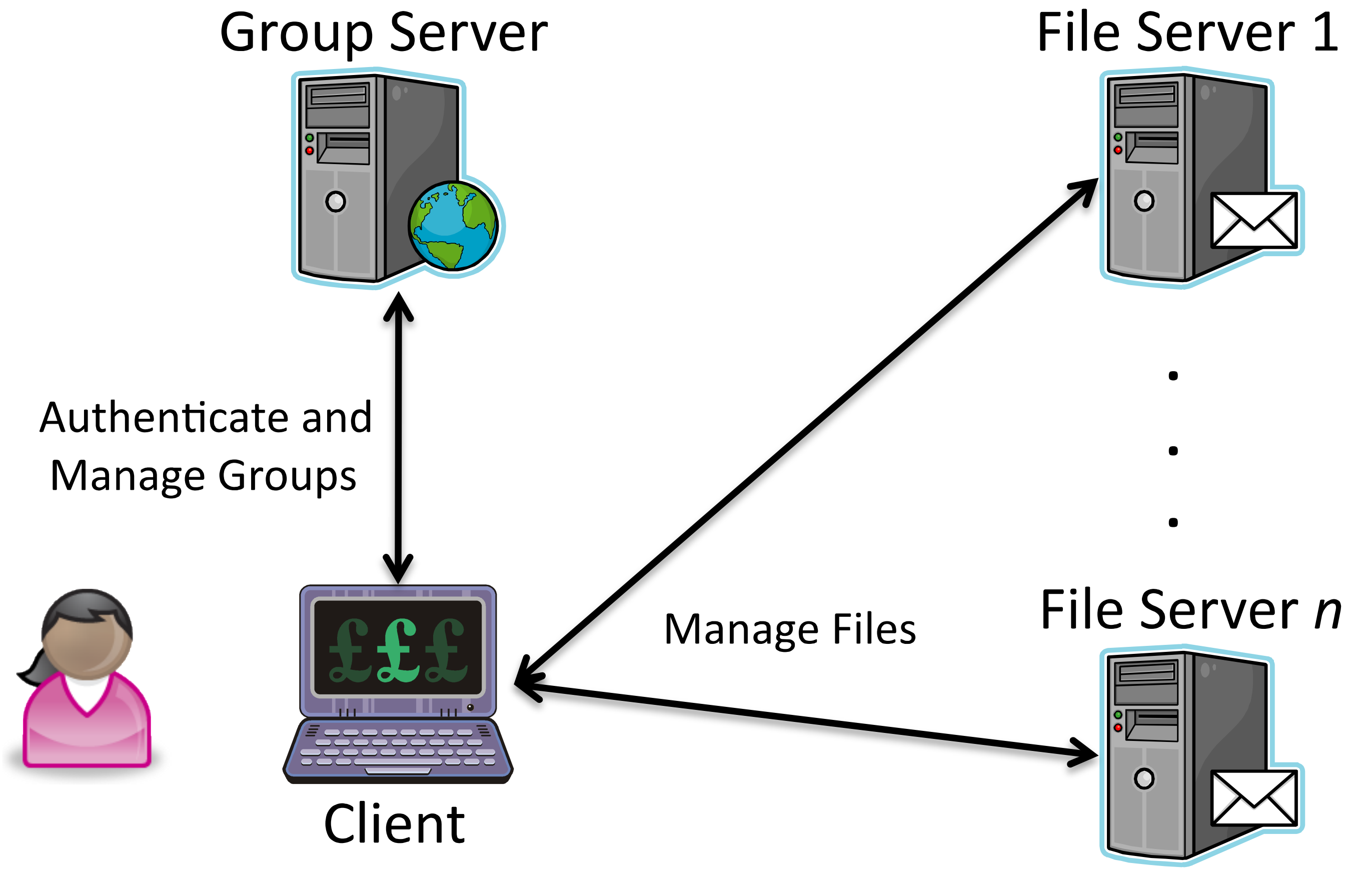
Applications of cryptography

Next: The basics of modern symmetric key cryptography

Right now: Project!

# Groups!

# File Sharing System



# Threat Models?

Example: Intra-office Protected Subnet

The system will be deployed within a small organization to facilitate file sharing between members of the technical staff. All servers will be operated on a subnet that can only be accessed from a wired connection inside of the office building, and only machines whose MAC addresses have been explicitly authorized can connect to these wired ports.

As such, it is assumed that only members of the technical staff can listen to communications on this subnet, and that servers on this subnet cannot communicate with the broader Internet.

At least two threat models!