# Introduction to Modular Arithmetic Using Vigenère's Cipher

Lecture notes of Alexander Wood
CSCI 360 Cryptography and Cryptanalysis
awood@jjay.cuny.edu

John Jay College of Criminal Justice

# Monoalphabetic Ciphers

The previous ciphers we have seen, the shift cipher and the substitution cipher, are examples of **monoalphabetic ciphers** which use a *fixed* substitution over the entire message.

# Polyalphabetic Ciphers

The next big “leap forward" in cryptography came in the form of *polyalphabetic ciphers*.

# Polyalphabetic Ciphers

Fun fact: The **Enigma Machine** from WWII is an example of a (very complex) polyalphabetic cipher.

# Vigenère's Cipher

A popular and straightforward polyalphabetic cipher is the **Vigenère Cipher**. Now, a message is encrypted using a predefined **keyword**.

# Intro to Modular Arithmetic

In order to use Vigenère's Cipher we must first learn some basics of **modular arithmetic**. A common example of modular arithmetic that we are all familiar with is time!

# Modular Arithmetic: The Clock

Consider a clock – it starts at the top with a 12. Let's replace this 12 with a zero. Starting at noon, we are at hour zero and proceed

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 0, 1, 2, 3, 4, 5, \ldots$$

This is a method of counting **modulo 12**.

# Modulo n

To count **modulo 7**, we would count

$$0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, \ldots$$

To count **modulo n** we count

$$0, 1, 2, \ldots, n-2, n-1, 0, 1, 2, \ldots$$

# Residue Classes

Each integer from 1 up to $n-1$ can be expressed modulo $n$. We call these the **residue classes** modulo $n$, sometimes denoted $\mod n$.

# Residue Classes

Any integer can be described as its residue class modulo $n$.

- 14 is 4 modulo 5, denoted $14 \equiv 4 \pmod 5$.
- 6 is 1 modulo 5, denoted $6 \equiv 1 \pmod 5$.
- 17 is 3 modulo 14, denoted $14 \equiv 3 \pmod{17}$.

# Residue

We say that $a$ is the **modulo-$n$ residue of** $b$ when $b \equiv a \pmod{n}$, and $0 \leq a < n$.

Note that this residue relates each integer to its remainder after division by the **modulus** $n$.

# Congruence

**Congruence** is the mathematical term for equivalence modulo $n$. For instance, 11 and 1 are **congruent** modulo 5.

In general, $a - b$ are congruent modulo $n$ if $a - b$ is a multiple of $n$.

# Examples

- $31 \equiv 1 \pmod 5$ because $31 - 1 = 30$ is a multiple of 5.
- $20 \equiv 13 \equiv 6 \pmod 7$ because $20 - 13 = 7$, $20 - 6 = 14$, and $13 - 6 = 7$.
- $7 \not\equiv -6 \pmod 3$ because $7 - (-6) = 13$ is not a multiple of 3.

# Exercise 1: Residues

Compute the following residues.

- $15 \equiv ? \pmod 5$
- $21 \equiv ? \pmod 5$
- $4 \equiv ? \pmod 3$
- $57 \equiv ? \pmod{26}$

# Encoding the Alphabet with Modular Arithmetic

We can think of each letter as a number. In this way, $A = 0$, $B = 1$, $C = 2$, et cetera.

# Addition over Encoded Alphabet

This is useful because we can now add letters as if they were numbers! This provides a quick shortcut from the method we used in our previous codes, where we listed out the letters and performed computations over their indexes.

# Addition over Encoded Alphabet

For example, say we want to encrypt the word `HELLO` by “adding" the world `LUCKY` to it, ie, by using the **keyword** `LUCKY`.

# Addition over Encoded Alphabet

We would do this letter-by-letter, as before, but using modular arithmetic. We would first compute

$$A + L = 7 + 11 = 18 = S$$

because the letter in the alphabet at index 18 is S.

# Addition over Encoded Alphabet

The last computation we carry out is

$$O + Y = 14 + 25 = 38$$

However, there is no index 38 in the alphabet! Thus we must employ computation **modulo 26**.

$$38 \equiv 12 \pmod{26}$$

and hence is encrypted as M.

# Modular Arithmetic in Python

The modulus operator in Python is denoted `%`. For instance, $14 \pmod 8$ is computed as `14 % 8`, which returns `6`.

# Using ASCII

We can use ASCII as a convenient way of encoding the alphabet.

`http://sticksandstones.kstrom.com/appen.html`

However note that instead of having index 0, instead *A* has ASCII index 65 so our encoding will require an extra step.

# ASCII in Python

- Convert from letters to ASCII in Python using `ord()`:
  - *Example:* `ord('A') = 65`
- Convert from ASCII to letters using `chr()`:
  - *Example:* `chr(90) = 'Z'`

# Coding Exercise 1: Encoding The Alphabet with ASCII

Let's use ASCII and modular arithmetic to write a function, `alph_to_num`, which takes a capital letter as input and converts it to a number between 0 and 25 using ASCII codes and modular arithmetic.

# Coding Exercise 2: Converting from Numbers Modulo 26 to The Alphabet

Let's use ASCII and modular arithmetic to write a function, `num_to_alph`, which takes a number modulo 26 as input and converts it to the corresponding letter in the alphabet using ASCII.

# Vigenère's Cipher

First, Alice (A) and Bob (B) decide upon a keyword. This is done offline, privately. Encryption follows the following steps:

1) Write out the plaintext message
2) Below it, write out the keyword, aligning each letter of the keyword below each letter of the plaintext. Repeat the keyword over and over until you reach the end of the plaintext.
3) "Add" these letters together using their values modulo 26, as computed in the coding exercises in the previous slides. (*Note that this corresponds to a different Caeser shift for each letter in the keyword!*)

# A Helpful Chart

For hand computations we can use the chart provided at `http://www.counton.org/explorer/codebreaking/vigenere-cipher.php`.

| | 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 | D |
| **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 | C |
| **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 | B |
| **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 |

# Execise 2: Viginere's Cipher

Encrypt the following message by hand using Vigenère's cipher.

```
Plaintext:     GOODBYE
Keyword:       LUCKYLU
```

# Execise 2: Viginere's Cipher

Encrypt the following message by hand using Vigenère's cipher.

```
Plaintext:     GOODBYE
Keyword:       LUCKYLU
Encryption:    RIQNZJY
```

# Concept Check

Can we use a straightforward frequency analysis to hack Vigenère's cipher, as we could with the substitution cipher? Why or why not?

Brainstorm a new hacking method. (Don't worry about implementing this for now.)

# Annoucements

- We will have a quiz next class where you will need to:
  - Encrypt a short word using Vigenère's cipher, given the keyword and the Vigenère table.
  - Perform some basic modular arithmetic.
- Remember that Project 1 is due a week from today!

# References

- ASCII chart: `http://sticksandstones.kstrom.com/appen.html`
- Introduction to Modular Arithmetic:
  - `https://www.artofproblemsolving.com/wiki/index.php/Modular_arithmetic/Introduction`
  - `https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/what-is-modular-arithmetic`
  - A video: `https://www.youtube.com/watch?v=Eg6CTCu8iio`
- Viginere Cipher: `http://www.counton.org/explorer/codebreaking/vigenere-cipher.php`