School of

Engineering and Computer Science

Te Kura Mātai Pūkaha, Pūrorohiko

CYBR 171 T1 2023

Ngā whakapūtanga o Te Haumaru rorohiko
Cybersecurity Fundamentals

Classical Cryptography II

Learning objectives

- Implementation of Caesar cipher.
- Understand some issues related to Caesar and other substitution ciphers.
- Samples of modern symmetric ciphers.
  - DES
  - AES
  - Blowfish

PART I:
Caesar’s Cipher implementation and related issues

How Is That Helpful?

- When using a Caesar cipher, you assign each letter to an index starting from 0.

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- You would then compute the following.
  *(plain letter index + **key**) mod (total number of letters)*
- This will give you the index of the encrypted letter!
- As you can see, the modulus is the total number of letters in the alphabet. For English, this modulus is 26.

A Simple Example

- Let's say we have a 5 letter alphabet with only the letters A-E
- First, we assign each letter an index, starting from 0.

A	B	C	D	E
0	1	2	3	4

- We then have to choose a key. For this example, we'll use 2.
- Let's try encoding the word BEAD using the formula for the previous slide.
- The index of the letter B is 1. The key is 2. The modulus is 5, since the alphabet is 5 letters.
- Let's use the algorithm: (1+2) = 3. 3 mod 5 = 3. The index of D is 3, so B would become the letter D.
- Using algorithm on each letter, can you encode the full word?

DBCA

Why Does Modding Work?

- The mod accounts for wrapping back around once you reach the end of the alphabet.

A	B	C	D	E
0	1	2	3	4

- When converting the letter E using the key 2, you first add 4 + 2, which gives you 6. As you can see, the index 6 is beyond the scope of this alphabet. When you do 6 mod 5, you get the remainder of 1, which is how much you need to go back and count from the beginning of the alphabet.

What About Decoding?

- To decode, you do the following:
  (cipher letter index – key + **total number of letters**) mod (total number of letters)

- Does anyone know why we can’t just do:
  (cipher letter index – key) mod (total number of letters)?

- By adding the total number of letters, you don’t have to end up taking the mod of a negative number.
- See if you can decode DBCA to get back to the word BEAD.

How Would You Program This?

- It’s unbelievably simple!
- The mod function is one of the most basic components of a lot of programming languages. You’ll learn the mod function for python early on. That means all we have to deal with is indexing the letters.
- Believe it or not, this also is no problem. The solution lies in UTF-8.

Wait, What's ASCII?

- Computers have to represent everything as numbers, including things as basic as text.
- ASCII was an early standard character encoding scheme, where each symbol is assigned a number.
- These symbols range from upper and lower case letters to numbers to things like punctuation and arrows.
- Problem was that the letters were related to the Latin alphabet used by European languages.

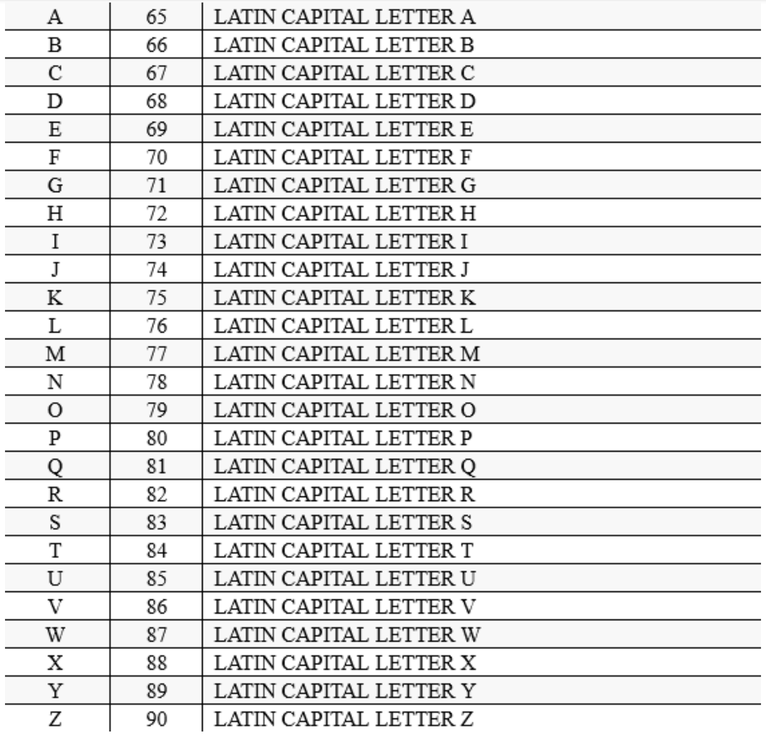
What’s UTF-8?

- Unicode is the universal standard for encoding all human languages. It supports millions of characters. Even includes emojis. Unicode allows characters specific to a particular language to be identified. It doesn’t map to a number though.

- UTF-8 is a way of expressing Unicode as numbers in one-byte numbers and includes all of the ASCII characters. One byte is eight bits and can represent an decimal value between 0 and 255.

- There are many tables online that allow you to look up the number associated with each symbol. We’ll work with the integer values to start with but when working in Linux you might see them represented as hexadecimal or even binary.
  https://blog.hubspot.com/website/what-is-utf-8

The UTF-8 Table (Latin)



A	65	LATIN CAPITAL LETTER A
B	66	LATIN CAPITAL LETTER B
C	67	LATIN CAPITAL LETTER C
D	68	LATIN CAPITAL LETTER D
E	69	LATIN CAPITAL LETTER E
F	70	LATIN CAPITAL LETTER F
G	71	LATIN CAPITAL LETTER G
H	72	LATIN CAPITAL LETTER H
I	73	LATIN CAPITAL LETTER I
J	74	LATIN CAPITAL LETTER J
K	75	LATIN CAPITAL LETTER K
L	76	LATIN CAPITAL LETTER L
M	77	LATIN CAPITAL LETTER M
N	78	LATIN CAPITAL LETTER N
O	79	LATIN CAPITAL LETTER O
P	80	LATIN CAPITAL LETTER P
Q	81	LATIN CAPITAL LETTER Q
R	82	LATIN CAPITAL LETTER R
S	83	LATIN CAPITAL LETTER S
T	84	LATIN CAPITAL LETTER T
U	85	LATIN CAPITAL LETTER U
V	86	LATIN CAPITAL LETTER V
W	87	LATIN CAPITAL LETTER W
X	88	LATIN CAPITAL LETTER X
Y	89	LATIN CAPITAL LETTER Y
Z	90	LATIN CAPITAL LETTER Z

https://utf8-chartable.de/unicode-utf8-table.pl?utf8=dec

Using UTF-8 for Caesar’s Cipher

- You probably noticed that the letter A is associated with the number 65, B with 66, C with 67, and so on.
- Most programming languages have a function that can take a letter and find out it’s ASCII value.
- What would you need to do to make the ASCII encoded letters usable for Caesar’s Cipher?
- Subtract 65 from each letter!
- Since it’s really simple to code for a computer to use mod and assign the correct index for each letter, you can probably see how programming a simple encoder/decoder would be a cinch!

PART II:

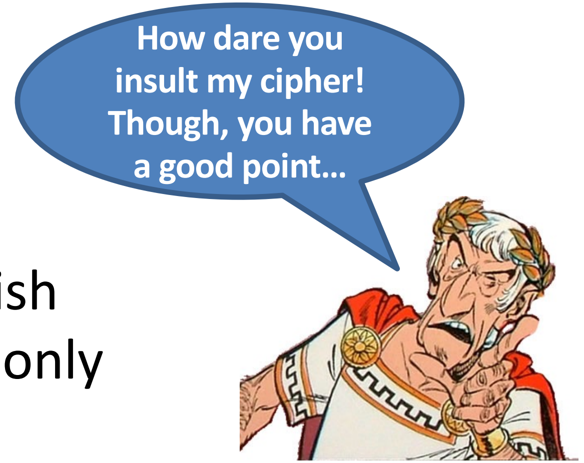
Issues of substitution ciphers and solutions

The Problem with Caesar's Cipher

- It's too easy to break! To illustrate let's consider a five letter alphabet rather than our twenty six letter alphabet, i.e. ABCDE.
- Say you had the ciphertext DBCA, but didn't know the key. It would be easy to make a list of all possible keys. If you expected the unencrypted text to be in English, you could easily figure out which word was right:

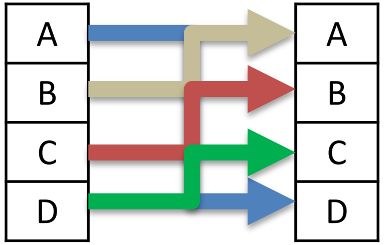
Shift of 1	**CABE**
Shift of 2	**BEAD** ← **The clear winner!**
Shift of 3	**ADEC**
Shift of 4	**ECDB**

- Can make it harder to break by using the full English alphabet and not restricting substitution to shifts only means 26! or 403,291,461,126,605 combos.


How dare you insult my cipher! Though, you have a good point…

Improving on Caesar's cipher

- You can make the job of cryptanalysis significantly hard by increasing the alphabet and not restricting substitution to a single shift.
  - Note that cryptanalysis is a process of finding weaknesses in cryptographic algorithms and using these weaknesses to decipher the ciphertext without knowing the secret key.
- What does not restricting substitution to a single shift mean?
  - Any letter can be mapped to any other letter
  - Mapping is done by having a different shift for each letter in the alphabet
- Can visualise this as a plug board connecting the plaintext alphabet with the ciphertext alphabet. For example, A maps to D, B maps to A, C maps to B, D maps to C.



- Using the full English alphabet and not restricting substitution to a single shift means that there are now potentially factorial 26! or 403,291,461,126,605 combos.

The problem with substitution ciphers

- Al-Kindi’s (800-873 CE) insight was that characteristics of a letter are not changed by fixed substitution of one letter or another letter or symbol.
- Frequency of occurrence of each letter will be unchanged between the plaintext and ciphertext.
- Cryptanalysis can use this information to make educated guesses about the substitution-rules based on looking for the most common ciphertext letter and assuming it maps to the most common plaintext letter. Repeat for other letters.
- For example, most common letters in English are EARI and O. Assume that our ciphertext most common letters are FCEUR.
  - Where a simple SHIFT is used, could assume that F is E so and try a shift of one and see if the result of the decryption makes sense in English
  - Where any substitution is allowed, could assume F->E, C->A, E->R, U->I, R->O.

Uhoh! The relative frequencies do not change!

A solution - Polyalphabetic ciphers

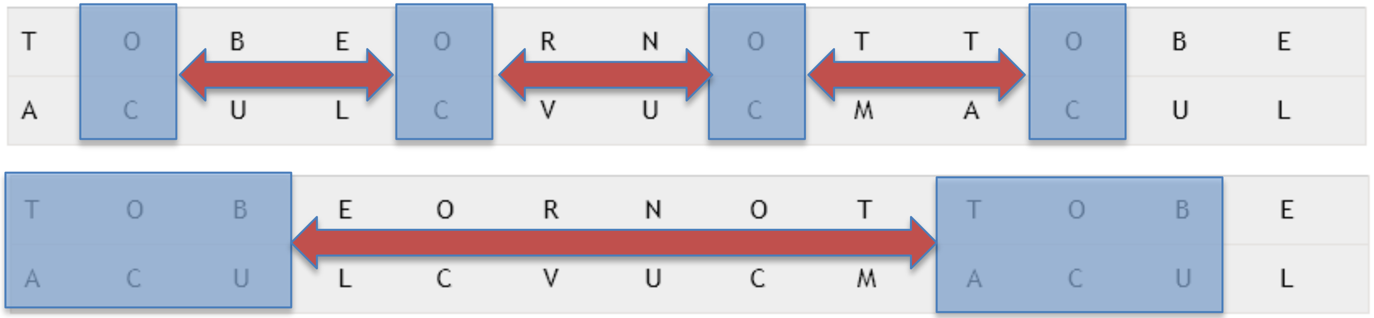
- Blaise de Vigenere (1523-1596 CE) cipher breaks relative frequency by changing the substitution rule for every character of the plaintext.
- For Vigenere, as the length of the keyword increases, the letter frequency shows less English-like characteristics and becomes more random.
- Simple scheme is to use Caesar ciphers with different shifts for each letter in the plaintext.
    - First letter of message, shifted by 7, second letter of message, shifted by 14, third letter of message, shifted by 19, fourth letter of message, shifted by 7 … repeat the shifts 7, 14, 19 throughout rest of message.
    - Key is now the number sequence 7, 14, 19.
    - Key is usually written as characters rather than numbers so hot

T	O	B	E	O	R	N	O	T	T	O	B	E
A	C	U	L	C	V	U	C	M	A	C	U	L

    - Note that T maps to A sometimes and other times T maps to M.

How to break a polyalphabetic cipher?

- Parallel invention – Prussian colonel Friedrich Wilhelm Kasiski (1805-1881) and Charles Babbage (1701-1871).
- Observation – if we use N different substitutions in a periodic fashion then every Nth character is enciphered with the same monoalphabetic cipher.
- Key is to find N, the length of the key and apply frequency analysis to sub-cryptograms composed of every Nth character of the cryptogram.
- We find N by looking for repeated sequences in the ciphertext.

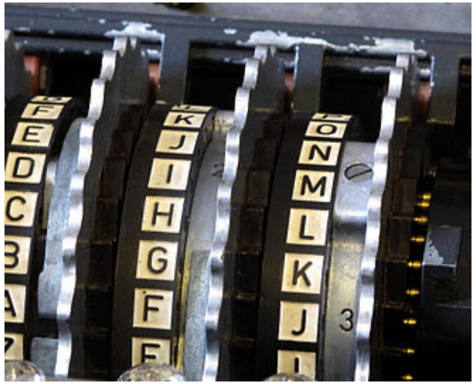
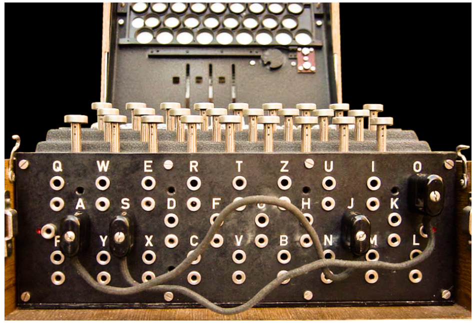


T O B E O R N O T T O B E
A C U L C V U C M A C U L
T O B E O R N O T T O B E
A C U L C V U C M A C U L

- Based upon our example from the previous slide, the possible candidates for N = 3 and N = 9. Break into multiple sequences and apply frequency analysis.

Enigma machine – automated analysis

- Electromechanical machines built to implement polyalphabetic ciphers:
    - Rotor machines that implement shifting for Caesar ciphers
    - Multiple rotors used to implement Viginere-style ciphers
    - Plug boards used to implement general substitution of the output of the rotors
    - Trillions of possible permutations possible
- Most famous of these machines was the Enigma used by Germans for secure communication to support military operations.
- Initial breakthrough determined wiring of one rotor by Marian Rejewski from Cipher Bureau of Polish Intelligence Service in Warsaw (1933) allowing decryption of German messages
- After invasion of Poland, effort passed to Bletchley Park in England to apply these ideas to further cryptanalysis. Alan Turing developed a special purpose electromechanical computer called the “bombe” that automated cryptanalysis,

Requirements and aids for success

- Requirements for successful cryptanalysis:
  - You need to know the language used for the plaintext.
  - You need to know the relative frequencies of letters in the plaintext.
  - You must have plenty of ciphertext to analysis for frequency analysis.
- Aids for success (**human error**):
  - You hope that the key is used repeatedly to increase the amount of ciphertext available or they use it for very long messages.
  - You hope that you have some **cribs** (plaintext) and encrypted version (cipher text) that allow you to quickly discard some potential keys.
  - You hope that the choice of new keys follows a pattern that you can guess.
- This was used to speed up analysis of Enigma messages:
  - Operators chose obvious keys (three successive letters)
  - Operators repeated the same key (initials of their name)
  - Operators changed the keys daily but followed certain rules reducing the possible number of keys (no rotor setting was repeated from one day to the next, and no letter of plugboard replaced by its neighbouring letters).

Concept: One time pad (OTP)

- A one-time pad is **the only cipher scheme** that is **unbreakable**, that is prefect secrecy is achieved.
- Invented by Frank Miller (1882) for telegraph messages and went through various refinements.
- Similar to a **polyalphabetic substitution** except that the secret key is **random**, **same length as the plaintext** and **used only once**.
    - **These are actually quite hard to achieve in practice**
- Proof by Shannon, the key idea being that because of random key all possible messages are equally likely, so there is no way to know which one is correct.

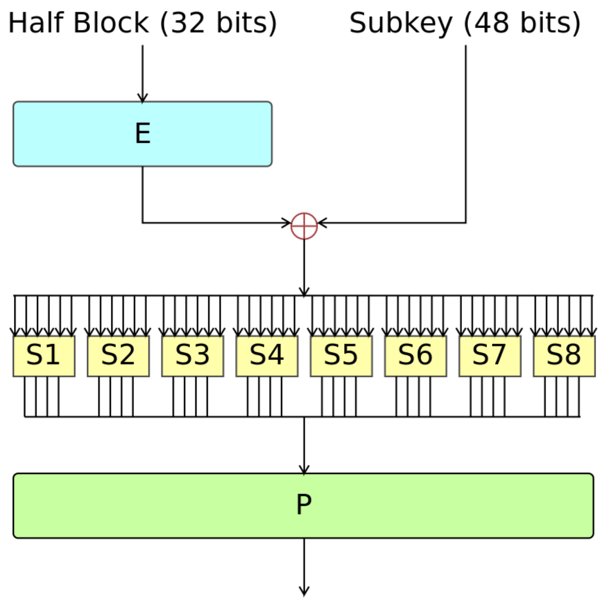
Who uses them - Spies!

- Random one time pads needed to be:
  - Very small to hide them
  - Easily destroyable
  - Used numbers to make it easy to implement, just add them to a number representing your letter (no carries)
  - Tradecraft followed and only every used once
- German Foreign Office GEE [1923-1945] used flawed implementation:
  - Pseudo-random generator for OTP
  - WW II British were able to predict output of GEE pseudo-random generator making it possible to reverse-engineer the sequence
- British Special Operations Executive in WW II
  - Small paper sheets used (mickey mouse pads)
  - Printed silk scarves
- Believed to still be used by spies today …

PART III:
Modern cryptography

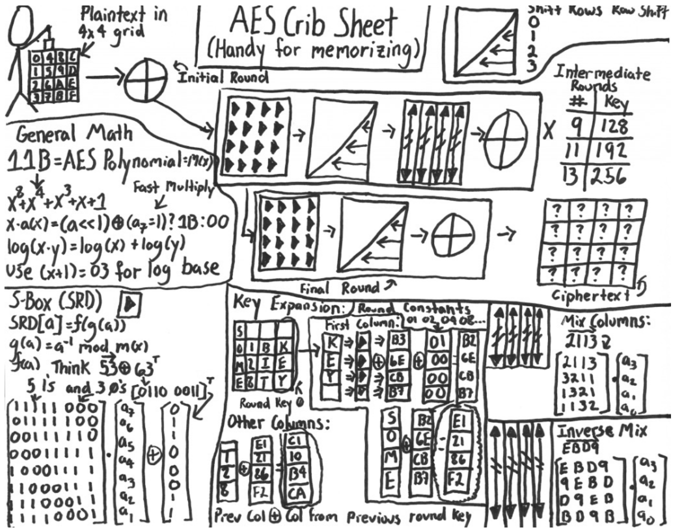
DES

- **D**ata **E**ncryption **S**tandard.
- 1970s, adopted by US forerunner of NIST
- Small key size (56 bits), easy to crack with brute force (<1 day).
- Led to competition for new standard, and use of triple DES (three rounds, three 56 bit keys)
- Triple DES is predicted to be safe until 2030.



AES

- **Advanced** **Encryption** **Standard**.
- Adopted in 2001 as US Government standard.
- Competition between fifteen designs (1997-2000).
- Rijndael won out as the basis of AES over Blowfish-variant.
- 128, 192 or 256 bit keys.
- Widely used, Windows 2000 onwards.



Blowfish

- Early 1990s, potential drop-in replacement for DES
- Was unusual as having public domain status (you can use it for free).
- Variable key lengths from 1 to 448 bits.
- No successful attempt to break it as long as setup key right (they can take a while and people skip setting them up properly).