



# Cryptology for IoT

**Modules M4, M6, M8**  
**Session of 24th May, 2022.**

M6.1 Briefing of the session  
M6.2 Friedman Test  
M6.3 Hill Climbing  
M6.4 Final Exercise using Hill Climbing

Prof.: Guillermo Botella



# Cryptology for IoT

**Modules M4, M6, M8**  
**Session of 24th May, 2022.**

**M6.1 Briefing of the session**  
M6.2 Friedman Test  
M6.3 Hill Climbing  
M6.4 Final Exercise using Hill Climbing

Prof.: Guillermo Botella



# M6.1 Briefing of today

- Keeping going with Cryptography and Cryptoanalysis (Crypto lab v1 and v2)
  - Slides and supplementary videos
  - Deal with Unknown cipher
  - Friedman Test
  - Hill Climbing
- Create and break our own Homophonic Substitution code
- I warn you for tomorrow 25<sup>th</sup> May. We will go to the Socrative. Second quiz (continuation of First quiz)
  - Please study the slides!



# Cryptology for IoT

Modules M4, M6, M8  
Session of 24th May, 2022.

M6.1 Briefing of the session  
**M6.2 Friedman Test**  
M6.3 Hill Climbing  
M6.4 Final Exercise using Hill Climbing

Prof.: Guillermo Botella

# Friedman Test for classical and modern crypto



## Cryptanalysis

### Cost / Fitness Function

Where Do We Need and Use Them?

- . Entropy
- . Language Models
- . Index of Coincidence

# Friedman Test for classical and modern crypto

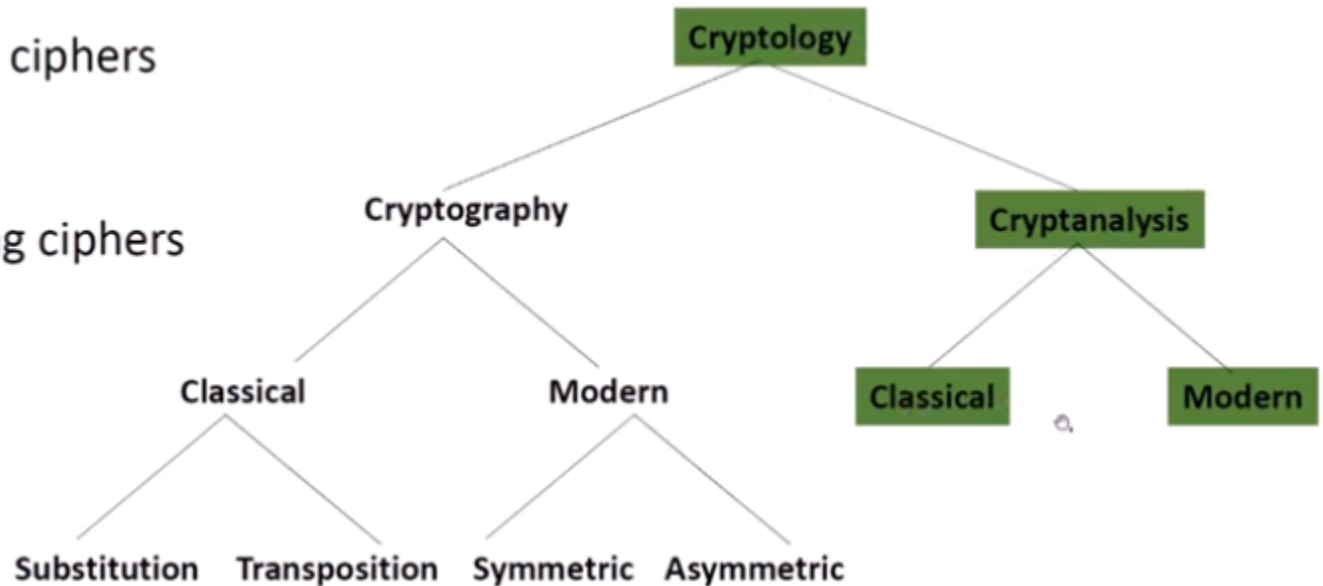


## Cryptography

Art of making ciphers

## Cryptanalysis

Art of breaking ciphers





# Friedman Test for classical and modern crypto

- The Index of Coincidence (IoC) is a **statistical value**. It is the **probability** that **two letters randomly drawn** (from **different positions** of a given text) **are the same**
- The inventor of the Index of Coincidence was **William Friedman**
- It is defined as

$$IoC = \sum_{i=A}^Z \frac{n_i \cdot (n_i - 1)}{N \cdot (N - 1)}$$

- The **sum of the number of each letter  $n_i$**  multiplied with the **sum of each letter minus one ( $n_i - 1$ )** divided by the **sum of all letters  $N$**  multiplied by the **sum of all letters minus one ( $N - 1$ )**
- English texts have an IoC of about 0.066 and German texts have an IoC of about 0.078
- Random texts have an IoC of about  $\frac{1}{\#(\text{letters in alphabet})} \rightarrow \frac{1}{26} = 0.038$

# Friedman Test for classical and modern crypto



- The IoC was used by Friedman in the **Friedman Test** to determine the keylength of a polyalphabetic substitution cipher, e.g. the Vigenère cipher (not part of this video)
- We use the IoC as **cost or fitness function** in **modern heuristics** to **improve our (putative) key** during e.g. hill climbing
- The **closer the IoC** of the current decrypted plaintext is to the **IoC of the assumed language**, the **“better” is our key**
- The IoC can be used when other statistical values can't be used (e.g. ADFGVX, Enigma rotors)



# Friedman Test for classical and modern crypto



- Example Calculation:

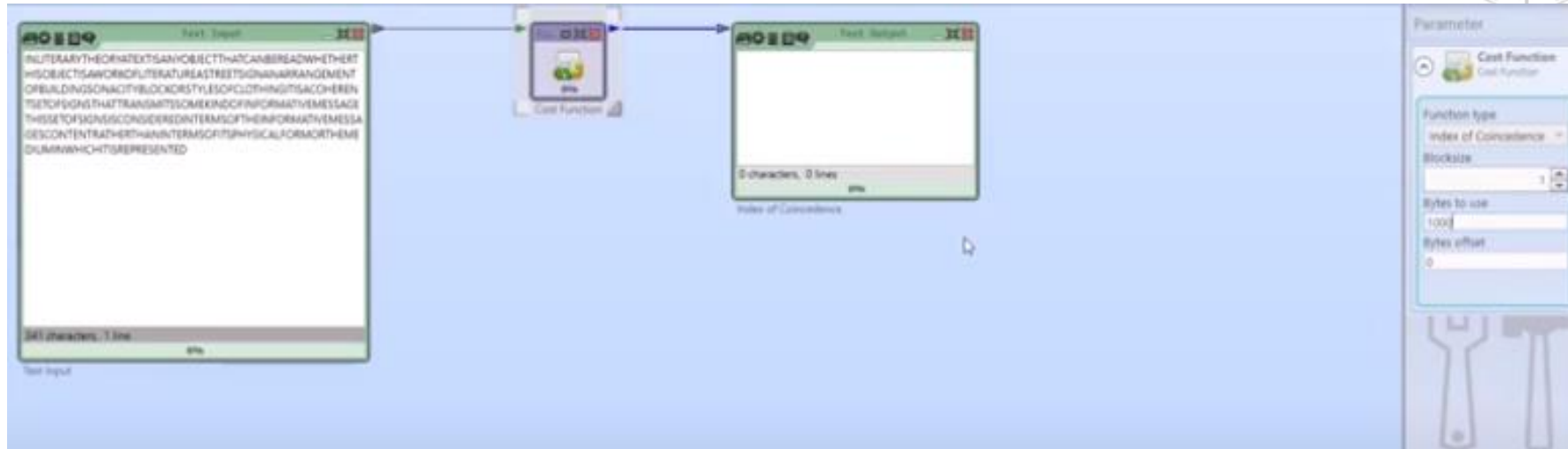
INLITERARYTHEORYATEXTISANYOBJECTTHATCANBEREADWHETHERTHISOBJECTISAWORKOFLITERATUREASTREETSIGNANARRANGEMENTOFBUILDINGSONACITYBLOCKORSTYLESOF CLOTHINGITISACOHERENTSET OF SIGNSTHAT TRANSMIT SOME KIND OF INFORMATIVE MESSAGE THIS SET OF SIGNS IS CONSIDERED IN TERMS OF THE INFORMATIVE MESSAGE CONTENT RATHER THAN IN TERMS OF ITS PHYSICAL FORM OR THE MEDIUM IN WHICH IT IS REPRESENTED

$$N = 341$$

$$n_A = 23, n_B = 5, n_C = 11, n_D = 7, n_E = 40, n_F = 11, n_G = 8, n_H = 16, n_I = 33, \\ n_J = 2, n_K = 3, n_L = 7, n_M = 12, n_N = 25, n_O = 24, n_P = 2, n_R = 25, n_S = 32, \\ n_T = 40, n_U = 3, n_V = 2, n_W = 3, n_X = 1, n_Y = 6$$

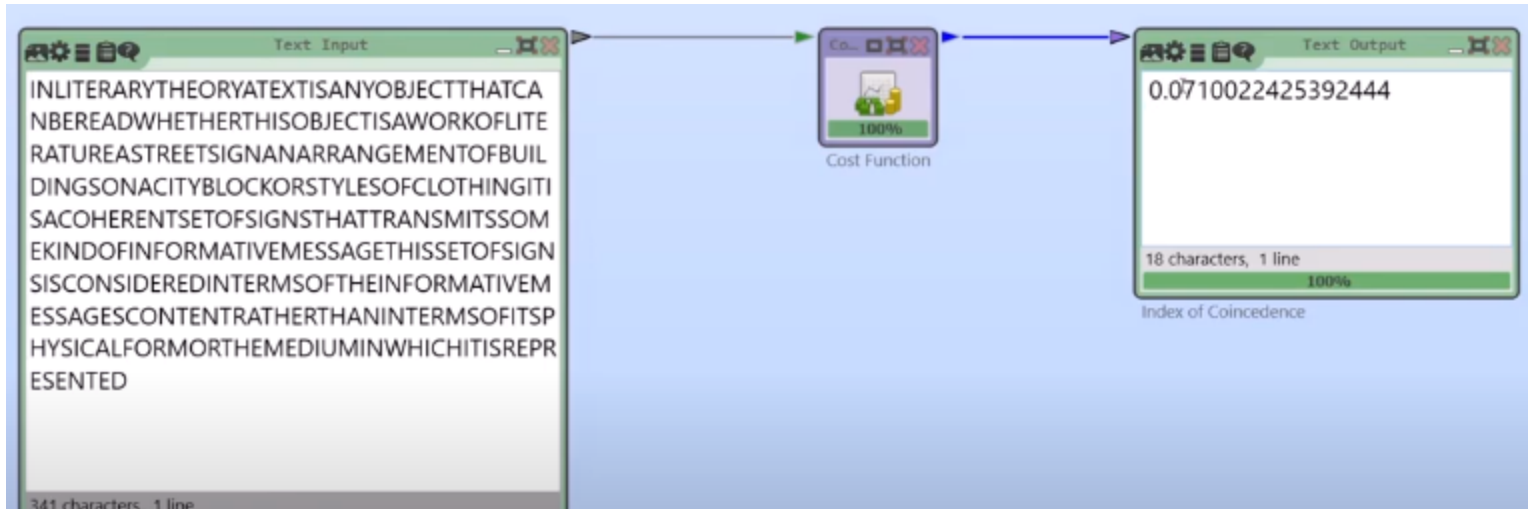
$$IoC = \frac{23 \cdot 22}{341 \cdot 340} + \frac{5 \cdot 4}{341 \cdot 340} + \frac{11 \cdot 10}{341 \cdot 340} + \dots + \frac{6 \cdot 5}{341 \cdot 340} = 0.071002243$$

# Friedman Test for classical and modern crypto



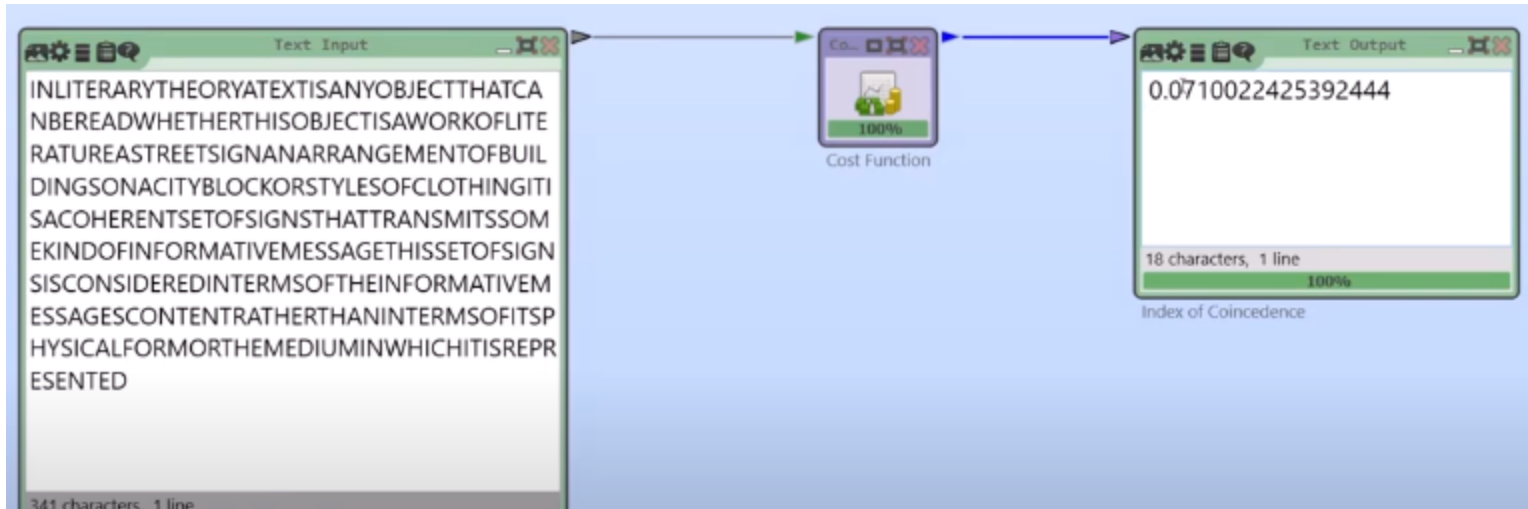
- Blocksize = 1
- 1000 bytes to use!

# Friedman Test for classical and modern crypto



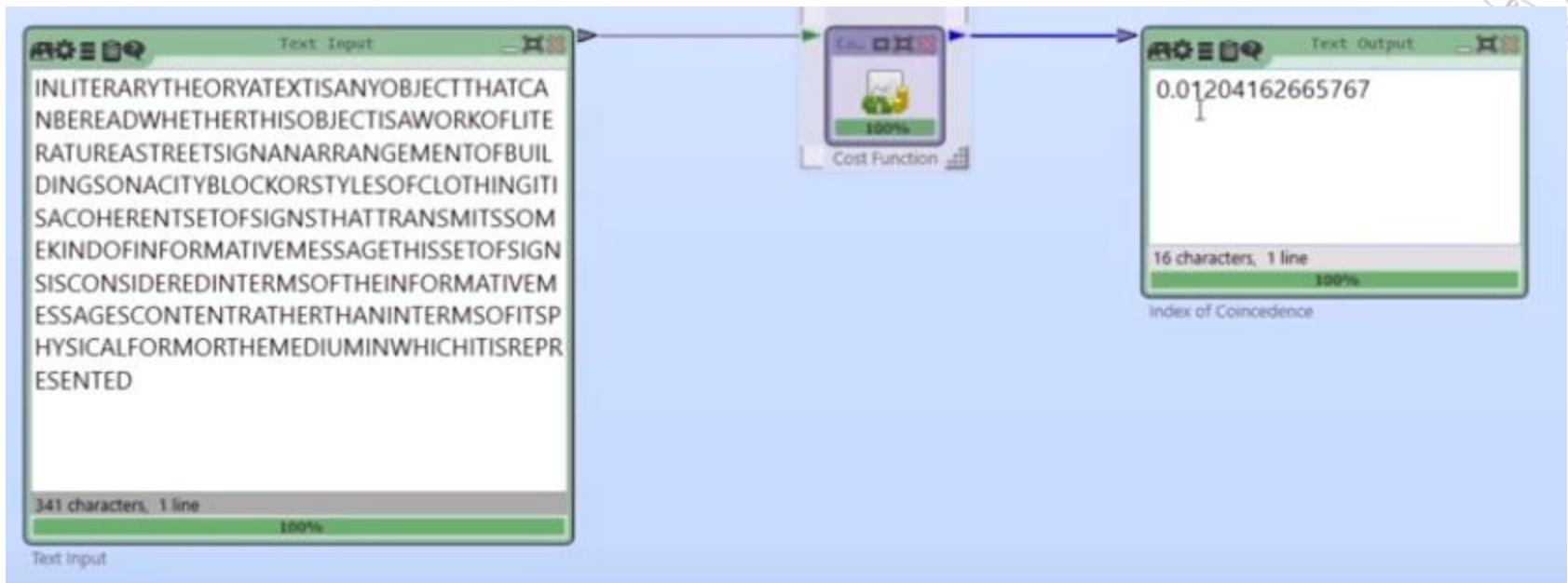
- Blocksize = 1
- 1000 bytes to use!

# Friedman Test for classical and modern crypto



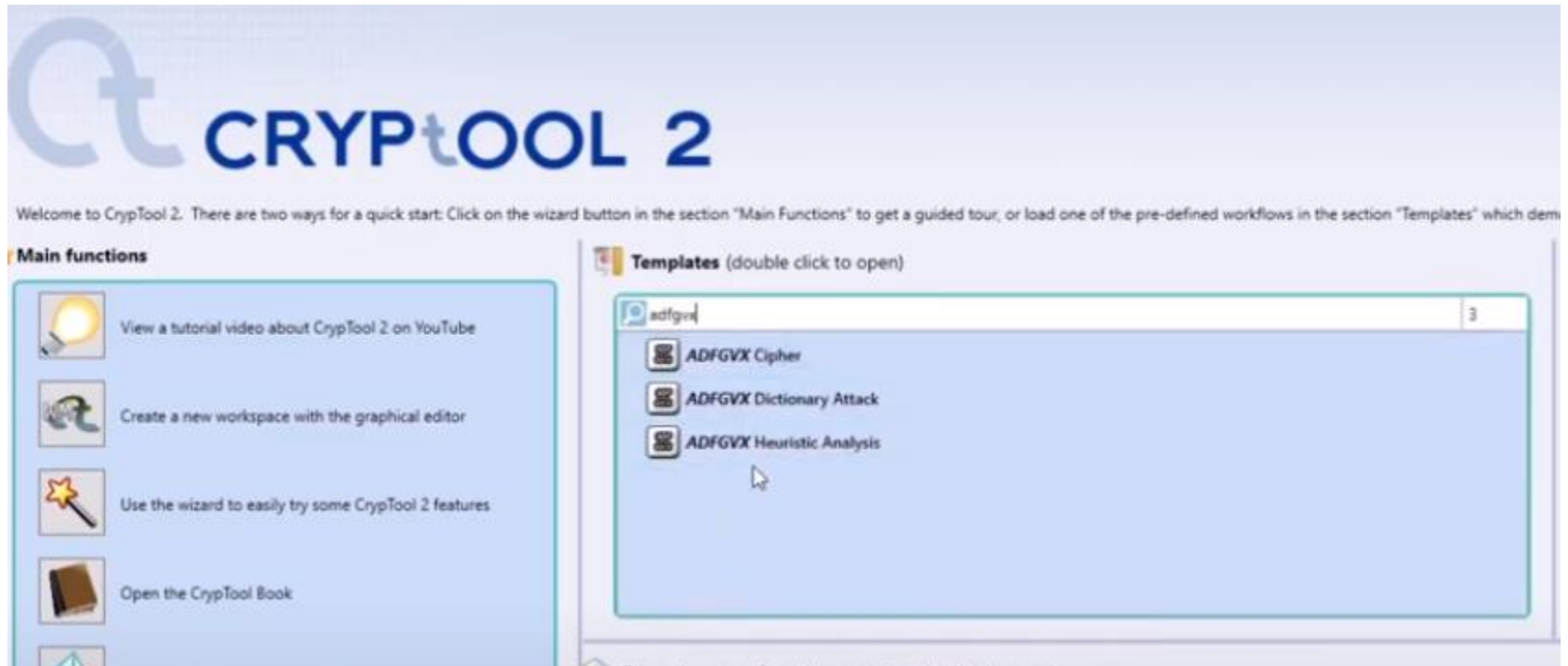
- Blocksize = 1
- 1000 bytes to use!

# Friedman Test for classical and modern crypto



- Blocksize = 2
- Probability goes down!

# Friedman Test for classical and modern crypto





# Friedman Test for classical and modern crypto

The screenshot shows the ADFGVX Analyzer interface. The central window displays a table with the following data:

| MessageLabel | ID    | Analyzed key/length | IS   |
|--------------|-------|---------------------|------|
| 1            | 79366 | 8                   | 0.28 |
| 2            | 79366 | 4.94                | 0.26 |
| 3            | 79365 | 4.88                | 0.25 |
| 4            | 79366 | 4.88                | 0.25 |
| 5            | 79379 | 4.88                | 0.25 |
| 6            | 79362 | 4.87                | 0.25 |
| 7            | 79359 | 4.83                | 0.24 |
| 8            | 89422 | 4.84                | 0.23 |
| 9            | 89423 | 4.89                | 0.23 |
| 10           | 89421 | 4.97                | 0.23 |

Below the table, there are sections for 'Decrypted messages' and instructions on how to use the tool, including a note about the assumed key length and natural language of the messages.

- Blocksize = 2
- Probability goes down!

# Friedman Test for classical and modern crypto



| Analysis |       | Start Time:   | 1/15/2021 12:54:14 PM | End Time:           | 1/15/2021 12:54:17 PM                       |  |
|----------|-------|---------------|-----------------------|---------------------|---|--|
|          |       | Elapsed Time: | 00:00:03              | keys / sec:         | 9426 (30101)                                |  |
|          |       | MessageLabel  | 10                    | Analyzed keylength: | 15  |  |
| Bestlist | #     | Score         | IoC 1                 | IoC 2               | Transposition key                           | Transposition result                           |
|          | 1     | 93752         | 5.54                  | 0.34                | NHBIAFEDCJJKOGLM                            | QGGKWNNELEKJRKUIHQHQRLLIOVKGAQGEFTJHRDEGHTJKU  |
|          | 2     | 93033         | 5.75                  | 0.33                | BHNI AFEDCJJKOGLM                           | KMGKWNNEFKJRKUINKHQRLIPUKGAQGEJBHRDEHGTJKU     |
|          | 3     | 83665         | 5.23                  | 0.29                | BHNI AFEDCLMJKOG                            | KMGKTKNQFKJRIKUNKHQORLJUKGAEQGXBJHQLDLDBGTJKA  |
|          | 4     | 83304         | 5.4                   | 0.28                | BINHAFELMJKOGDC                             | GQGHKNQWLEJIKURQHHORLKP IWGEQGFATFJKLDBNSHJAU  |
|          | 5     | 83067         | 5.42                  | 0.28                | BHNI AFELMJKOGDC                            | KMGHKNQWFKJIKURNKHORLKP UKGEQGXBJKLDLDBNGTJAU  |
|          | 6     | 82310         | 5.31                  | 0.28                | BHNI AFEJJKLMOGDC                           | KMGKNHQWFKJKIURNKHRIKPKUKGEQGXBJLEJBNGTJUA     |
|          | 7     | 81538         | 5.31                  | 0.28                | BINHAFEJJKLMOGDC                            | GQGHKNQWLEJIKIURQHHRIKPK IWGEQGFATFJLEJBNSHJUA |
|          | 8     | 80481         | 5.3                   | 0.27                | BINHAFEJMLKOGDC                             | GQGHKNQWLEJIKURQHHORLKP IWGEQGFATFJKLDBNSHJSC  |
|          | 9     | 76574         | 5.16                  | 0.25                | BINHAFEOMJKLGDC                             | GQGHKNQWLEJUKIRQHHRIKPK IWGKQAATFJJLEBNSHJAU   |
| 10       | 75966 | 5             | 0.26                  | DINHAFEOMJKLGC      | 4QGHKNNELEJUKIWHHRIKPK IWGKQAPBFJJLEATSHJAU |  |

- See difference between (IoC) unigrams and bigrams







# Cryptology for IoT

Modules M4, M6, M8  
Session of 24th May, 2022.

M6.1 Briefing of the session  
M6.2 Friedman Test  
**M6.3 Hill Climbing**  
M6.4 Final Exercise using Hill Climbing

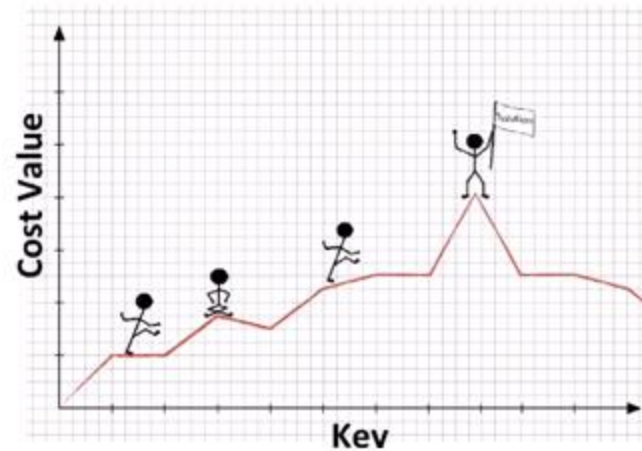
Prof.: Guillermo Botella

# Hill climbing for classical and modern crypto



## Cryptanalysis of Classic Ciphers Using Heuristics

- . What is a Heuristic?
- . Hill Climbing – Basics
- . Genetic Algorithm
- . Simulated Annealing – Improved Hill Climbing



# Hill climbing for classical and modern crypto

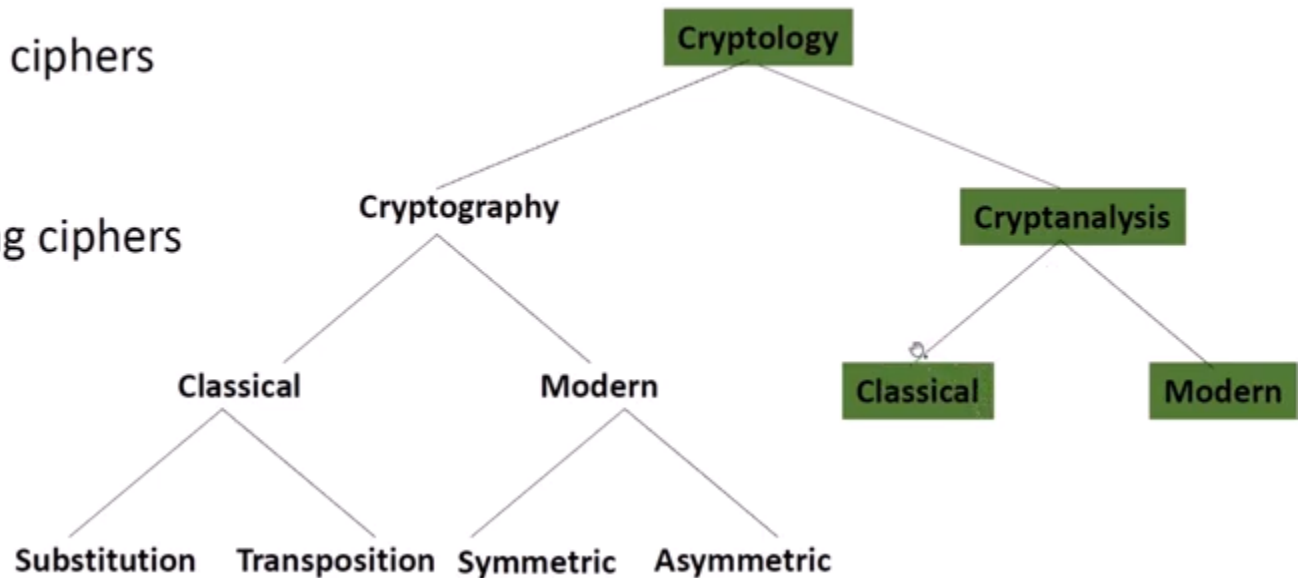


## Cryptography

Art of making ciphers

## Cryptanalysis

Art of breaking ciphers



# Hill climbing for classical and modern crypto



- A **heuristic technique** (/hjʊəˈrɪstɪk/; Ancient Greek: εὕρισκω, "find" or "discover"), or a **heuristic**, is any approach to **problem solving** or self-discovery that employs a **practical method** that is **not guaranteed to be optimal**, perfect or rational, but which is nevertheless sufficient for **reaching an immediate, short-term goal**. Where **finding an optimal solution** is impossible or **impractical**, heuristic methods can be used to speed up the process of finding a satisfactory solution. (English Wikipedia)
- Our problem: Finding the key of a classical cipher with brute-force is (mostly) **impractical**
- Can we **speed up the search** for the key using a heuristic, making the search **practical**? → Yes, of course 😊
- Hill climbing is a well suited heuristic to do so. Why? → On next slides 😊

# Hill climbing for classical and modern crypto



- First, we need to know the **vulnerabilities** of **classical ciphers** we **exploit** with **hill climbing**
- I focus now on substitution ciphers, but transposition ciphers also work
  1. **Letter frequencies** are **still visible** in ciphertext
  2. **Low diffusion** after changing key or plaintext
  3. We can measure **“how good (even a wrong) key is”**
  4. **Inventors** of (historical/classical) ciphers **didn't know computers** 😊

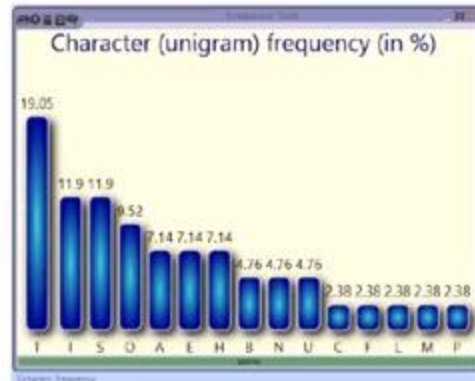
# Hill climbing for classical and modern crypto



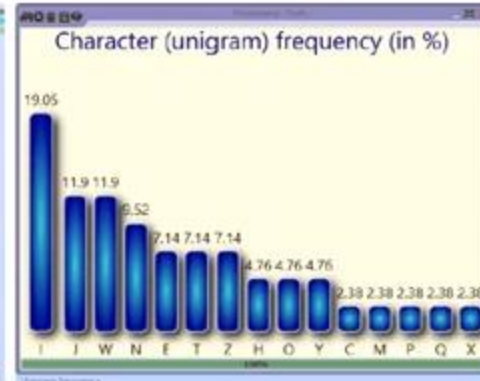
1. Letter frequencies are still visible in ciphertext

Example: Simple monoalphabetic substitution cipher

THIS IS A TEST OF THE MONOALPHABETIC SUBSTITUTION,  
ITWJ WJ Z IEJI NC ITE PNONZQMTZYEIWX JHYJIWIHIWNO



Plaintext Frequencies



Ciphertext Frequencies

# Hill climbing for classical and modern crypto



## 2. Low diffusion after changing key or plaintext

THIS IS A TEST OF THE MONOALPHABETIC SUBSTITUTION

Key: ZYXSECRTWVUQPONMLKJIHGFDBA

```
ITWJ WJ Z IEJI NC ITE PNONZQMTZYEIWX JHYJIWIHIWNO
```

51 characters, 1 line

Ciphertext

Changed Key: ZYISECRTWVUQPONMLKJXHGFDBA

```
XTWJ WJ Z XEJX NC XTE PNONZQMTZYEYXWJ JHYJXWXHXWNO
```

51 characters, 1 line

Ciphertext

In this example, only nine letters changed in the ciphertext. This is about 17% of the letters. Modern ciphers change on average about 50% of the bits when their inputs are changed.



# Hill climbing for classical and modern crypto



3. We can measure “how good (even a wrong) key is”

- “The better a key the higher is its fitness”
- Example (bigram log<sub>2</sub> cost function of CrypTool 2; performed on decrypted text)

|  |               |              |
|--|---------------|--------------|
| EYISZWRTCVUQOPNMLKJAHGFDBX                           | <b>193.24</b> | <- start key |
| EYISZ <b>CRTW</b> VUQOPNMLKJAHGFDBX                  | <b>214.27</b> |              |
| EYISZC <b>RTWVUQ</b> PONMLKJAHGFDBX                  | <b>218.16</b> |              |
| <b>ZYISE</b> CRTWVUQPONMLKJAHGFDBX                   | <b>231.14</b> |              |
| ZYISEC <b>RTWVUQ</b> PONMLK <b>JX</b> HGFDB <b>A</b> | <b>271.28</b> | <- final key |

# Hill climbing for classical and modern crypto



4. **Inventors** of (historical/classical) ciphers **didn't know computers**
  - **120,000 keys/sec** with **Vigenère analyzer** (on a single CPU core)
  - Enables optimization techniques (heuristics) to use vulnerabilities mentioned before
    - Hill climbing
    - Simulated annealing
    - Genetic algorithms
    - ...
  - **Parallelization and distribution** additionally speeds up the analysis 😊

# Hill climbing for classical and modern crypto



- Why not **just test every key** (search for the correct one via a **brute-force attack**) ? (by **rating all keys** and the **best should have the highest rating**)
- Simplest cipher: Simple monoalphabetic substitution  
We can achieve **~120,000 keys/sec**
- **Computation time** to search through all keys

$$26! = 2^{88.4} \text{ key}$$

$$\text{Search time} = \frac{2^{88.4} \text{ keys}}{120,000 \text{ keys/sec}} \approx 10^{21.5} \text{ sec} \approx 10^{14} \text{ years}$$

- Maybe, brute-force is **not a good idea...** ☹️

# Hill climbing for classical and modern crypto



Step 1: Create an **initial random key**

Step 2: **Decrypt ciphertext** using the initial key and **compute fitness** (e.g. trigram frequencies sum)

Step 4: **Modify key** (e.g. randomly swap letters)

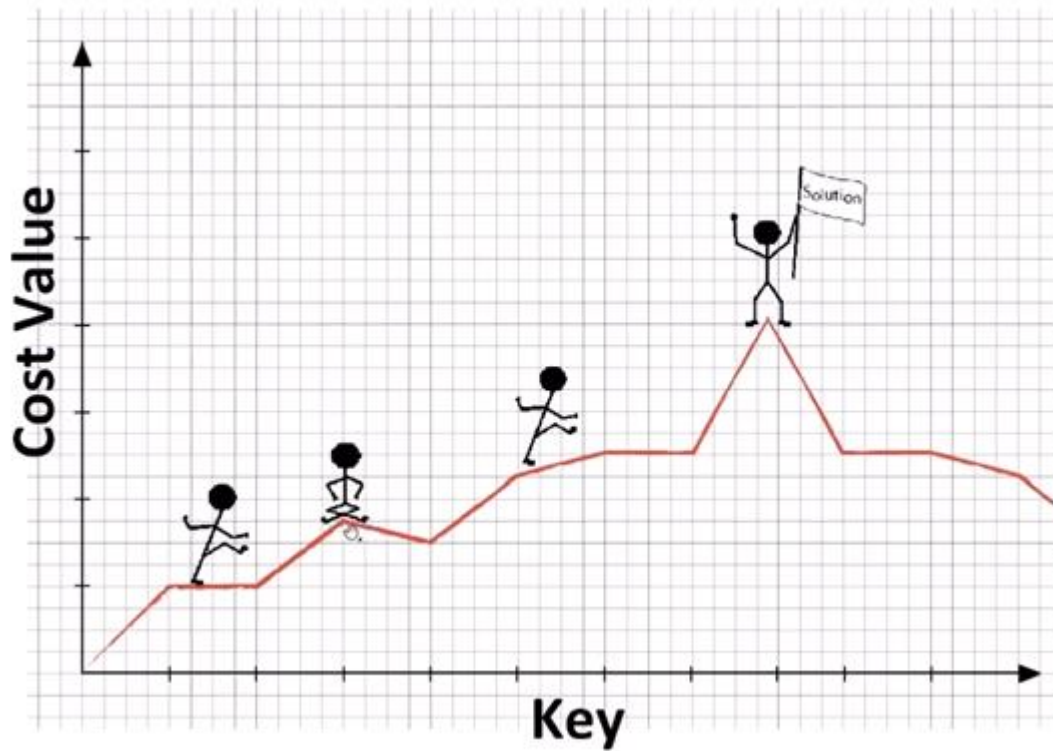
Step 5: **Decrypt ciphertext** using the modified key and **compute fitness**

Step 6: If fitness **is worse** than before **revert modifications**

Step 7: Increment a **counter**;  
If **counter is above a defined value**, we stop algorithm

Step 8: Jump to Step 4

# Hill climbing for classical and modern crypto





# Hill climbing for classical and modern crypto



The screenshot displays a software interface for a Transposition Analyzer. It consists of three main windows: 'Text Input', 'Transposition Analyzer', and 'Text Output'. The 'Text Input' window contains a block of ciphertext: 'RNLYTAVISSEAEWOHHRTAHWOGENKAUE FFAETROTHSEOOROYGVITBLRFJOOOWOMS KSVADRINAGREEDESHWLRTAEHBOTEETSE ODEDNERILYHPEESRLNRTSEODTSEALFI CEYLAHWOKLVREPHTHDRIGLITISEIROTFU NMDSLHTEHOFQKWSWFDHSAWVEANAT ROIGNINHROHTCOMMEOTEEDERTOHAERA FTTTUS FLDNNOOOOLEEM RTOH'. Below the text, it indicates '250 characters, 1 line' and '0%'. The 'Transposition Analyzer' window shows a progress bar at 0% and a table with columns for 'Start', 'End', 'Elapsed', 'Remaining', and 'Keys / sec'. A dropdown menu is set to 'Value Key Mode Text'. The 'Text Output' window is currently empty, showing '0 characters, 0 lines' and '0%'. To the right, a 'Parameter' panel for 'TA Transposition Analyzer' is visible, with 'Analysis method' set to 'Hill Climbing', 'Repetitions' at 10, and 'Iterations' at 50000. A tooltip提示 'Enter the number of iterations the algorithm perform for one random start value.' is shown over the iterations field.

This template shows the analysis of a ciphertext that was encrypted with a columnar transposition and a key of length 25. The analysis uses a hill climbing algorithm. This approach is not deterministic, hence, if the result (WHYHEWASSOANGRY...) is not found at the first run, simply try to run again.

The ciphertext was retrieved from the CC1-3 challenge of

- Different uses of hill climbing
  - Transposition analyzer



# Hill climbing for classical and modern crypto

This template shows the analysis of a ciphertext that was encrypted with a columnar transposition and a key of length 25. The analysis uses a hill climbing algorithm. This approach is not deterministic, hence, if the result (WHYHEWASSOANGRY...) is not found at the first run, simply try to run again.

The ciphertext was retrieved from the CC1-3 challenge of MysteryTwister I, where you can find many more ciphertexts that were encrypted with a columnar transposition. You can try to

| #  | Value     | Key   | Mode  | Text                  |
|----|-----------|---|-------|-----------------------|
| 1  | 721.01750 | [15, 17, 7, 18, 12, 23, 13, 11, 10, 24, 6, 2, 25, 19, 4, 16, 3, 9, 8, 14, 1, 20, 22, 21, 5] | R-C-R | WHYHEWASSOANGRYOLIVE  |
| 2  | 689.16332 | [15, 17, 7, 18, 12, 11, 10, 24, 23, 13, 6, 2, 25, 19, 4, 16, 3, 9, 8, 14, 1, 20, 22, 21, 5] | R-C-R | WHYHESSOWAANGRYOLIVE  |
| 3  | 684.78153 | [15, 17, 7, 18, 12, 23, 13, 2, 25, 19, 11, 10, 24, 6, 4, 16, 3, 9, 8, 14, 1, 20, 22, 21, 5] | R-C-R | WHYHEWANGRSSOAYOLIVE  |
| 4  | 676.56052 | [15, 6, 2, 25, 19, 4, 16, 17, 7, 18, 12, 23, 13, 11, 10, 24, 3, 9, 8, 14, 1, 20, 22, 21, 5] | R-C-R | WANGRYOHHYHEWASSOLIVE |
| 5  | 675.44185 | [15, 6, 2, 25, 19, 4, 16, 3, 9, 8, 14, 1, 17, 7, 18, 12, 23, 13, 11, 10, 24, 20, 22, 21, 5] | R-C-R | WANGRYOLIVRHHYHEWASSO |
| 6  | 675.40764 | [15, 6, 2, 25, 19, 4, 17, 7, 18, 12, 23, 13, 11, 10, 24, 16, 3, 9, 8, 14, 1, 20, 22, 21, 5] | R-C-R | WANGRYHHYHEWASSOOLIVE |
| 7  | 673.59066 | [16, 3, 9, 8, 14, 1, 20, 22, 21, 5, 11, 10, 24, 6, 2, 25, 19, 4, 15, 17, 7, 18, 12, 23, 13] | R-C-R | OLIVERTHATSSOANGRYWHY |
| 8  | 671.21902 | [15, 6, 2, 25, 19, 4, 16, 3, 9, 8, 14, 17, 7, 18, 12, 23, 13, 11, 10, 24, 1, 20, 22, 21, 5] | R-C-R | WANGRYOLIVHYHEWASSO   |
| 9  | 668.94294 | [15, 6, 2, 25, 19, 11, 10, 24, 4, 17, 7, 18, 12, 23, 13, 16, 3, 9, 8, 14, 1, 20, 22, 21, 5] | R-C-R | WANGRSSOHHYHEWIAOLIVE |
| 10 | 667.14966 | [15, 6, 2, 25, 19, 4, 16, 3, 9, 17, 7, 18, 12, 23, 13, 11, 10, 24, 8, 14, 1, 20, 22, 21, 5] | R-C-R | WANGRYOHHYHEWASSOIVE  |

- Different uses of hill climbing
  - Transposition analyzer





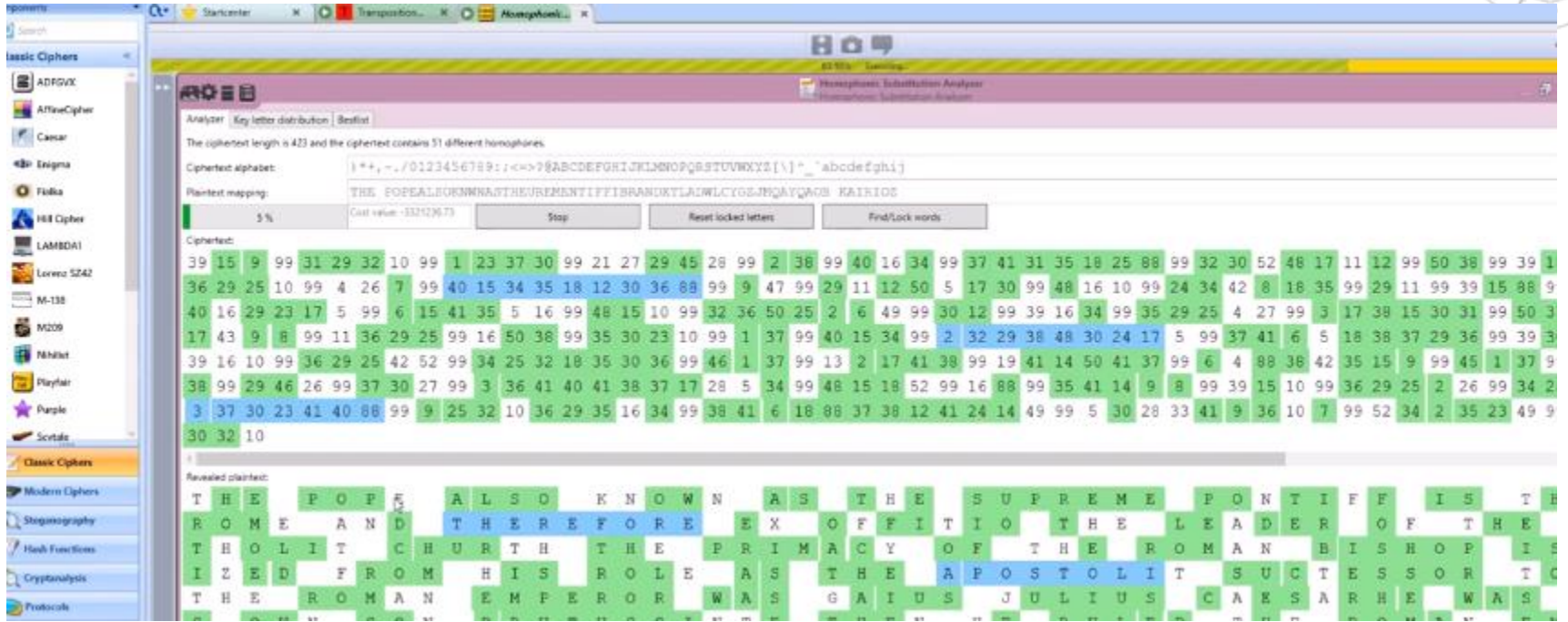
# Hill climbing for classical and modern crypto

**Cryptanalysis of the homophonic substitution cipher**

A homophonic substitution cipher tries to eliminate the possibility to analyze the ciphertext using simple language statistics. To do so, it flattens the frequencies of single letters, thus, in the perfect case the ciphertext letters are uniformly distributed. For example, instead of encrypting the letter 'E' only with one ciphertext letter, it can now be encrypted using one of several different "homophones", e.g. 01, 02, 03, 04, 05. Then the ciphertext consists of different pairs of digits — this method was often used in history, i.e. in letters kept in the Vatican's secret archive or in messages of the Spanish Civil War encrypted with the Spanish Ship Cipher. The keysize size of a homophonic cipher can be calculated by  $26^n$  where  $n$  is the number of homophones. For example, a homophonic encrypted text having only 52 homophones has a keysize size of  $26^{52}$  which is about  $2^{244}$ , where each homophone may be mapped to one of the 26 letters of the alphabet.

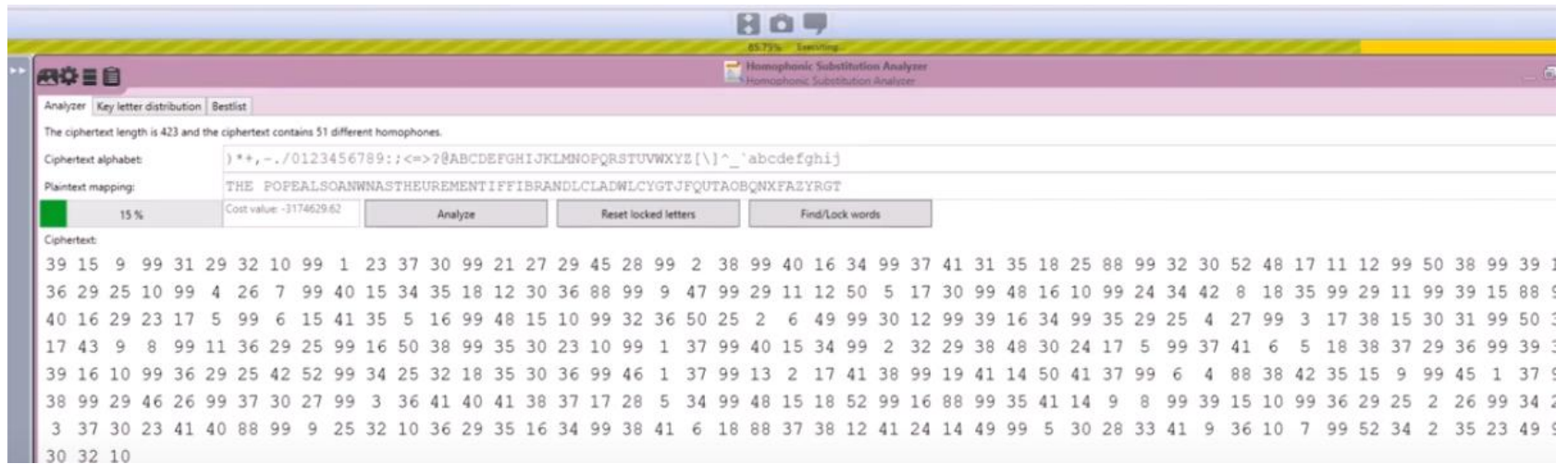
- Different uses of hill climbing
  - Homophonic Substitution analyzer

# Hill climbing for classical and modern crypto



- Different uses of hill climbing
  - Homophonic Substitution analyzer

# Hill climbing for classical and modern crypto



- Different uses of hill climbing
  - Homophonic Substitution analyzer

# Hill climbing for classical and modern crypto



- Different uses of hill climbing
  - Homophonic Substitution analyzer



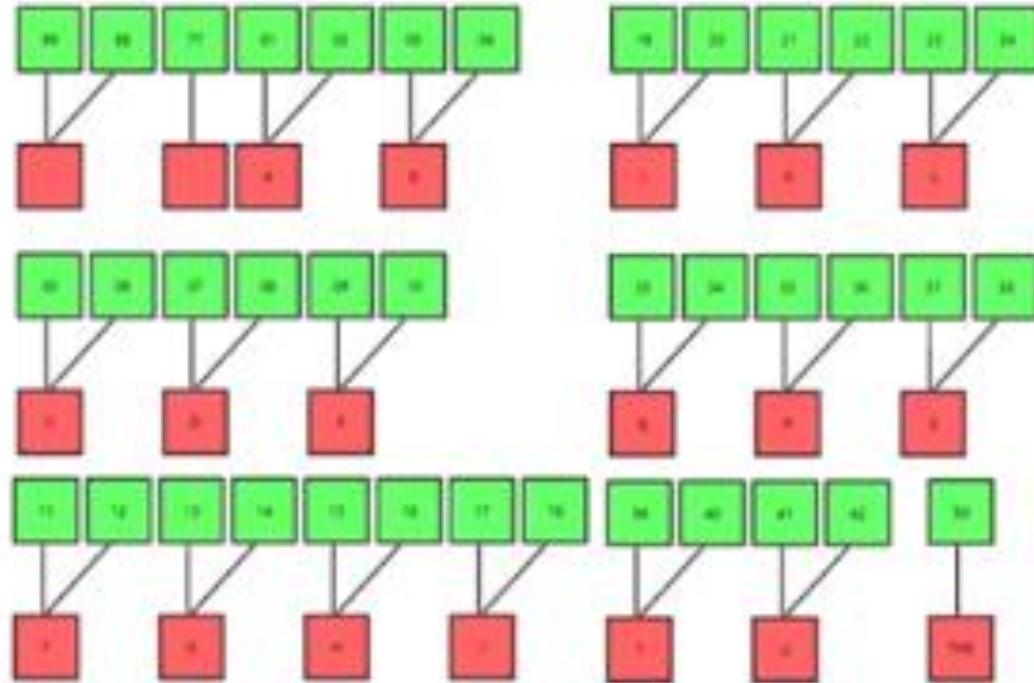
# Cryptology for IoT

Modules M4, M6, M8  
Session of 24th May, 2022.

M6.1 Briefing of the session  
M6.2 Friedman Test  
M6.3 Hill Climbing  
**M6.4 Final Exercise using Hill Climbing**

Prof.: Guillermo Botella

# Homophonic Substitution



Visualization of a homophonic substitution cipher  
 Green boxes = ciphertext symbols  
 Red boxes = plaintext symbols



# Homophonic Substitution

- A **simple substitution** replaces **single letters** by single letters, digit groups, or (in general) symbols
  - Examples: **Caesar** cipher, **Simple MASC**, **Pigpen** cipher
  - Simple substitution ciphers can be broken easily by hand (with frequency analysis)

Example encryption with a simple MASC\*:

Key: H=01, E=02, L=03, O=04, W=05, R=06, D=07

HELLO WORLD → 01 02 03 03 04 05 04 06 03 07

- To make ciphers more secure, cryptographers invented **homophonic substitution** ciphers
- A homophonic substitution has more choices (**homophones**) for encrypting letters (or pieces of text)

Example encryption with a homophonic substitution cipher\*:

Key: H=01, E=02, L=03 | 04, O=05 | 06, W=07, R=08, D=09

HELLO WORLD → 01 02 03 04 05 07 06 08 03 09

\* To make it easy to "produce" many ciphertext symbols, we use digits



# Homophonic Substitution

- Example **plaintext**:

IN CRYPTOGRAPHY A SUBSTITUTION CIPHER IS A METHOD OF ENCRYPTING IN WHICH UNITS OF PLAINTEXT ARE REPLACED WITH THE CIPHERTEXT IN A DEFINED MANNER WITH THE HELP OF A KEY THE UNITS MAY BE SINGLE LETTERS THE MOST COMMON PAIRS OF LETTERS TRIPLETS OF LETTERS MIXTURES OF THE ABOVE AND SO FORTH. THE RECEIVER DECIPHERS THE TEXT BY PERFORMING THE INVERSE SUBSTITUTION PROCESS TO EXTRACT THE ORIGINAL MESSAGE.

- Encrypted ciphertext using **simple MASC** (A=01, B=02, C=03, ..., Z=26):

09 14 00 03 18 25 16 20 15 07 18 01 16 08 25 00 01 00 19 21 02 19 20 09 20 21 20 09 15 14 00 03 09 16 08 05 18 00 09 19 00 01 00 13 05 20 08 15 04 00 15 06 00 05 14 03 18 25 16 20 09 14 07 00 09 14 00 23 08 09 03 08 00 21 14 09 20 19 00 15 06 00 16 12 01 09 14 20 05 24 20 00 01 18 05 00 18 05 16 12 01 03 05 04 00 23 09 20 08 00 20 08 05 00 03 09 16 08 05 18 20 05 24 20 00 09 14 00 01 00 04 05 06 09 14 05 04 00 13 01 14 14 05 18 00 23 09 20 08 00 20 08 05 00 08 05 12 16 00 15 06 00 01 00 11 05 25 00 20 08 05 00 21 14 09 20 19 00 13 01 25 00 02 05 00 19 09 14 07 12 05 00 12 05 20 20 05 18 19 00 20 08 05 00 13 15 19 20 00 03 15 13 13 15 14 00 16 01 09 18 19 00 15 06 00 12 05 20 20 05 18 19 00 20 18 09 16 12 05 20 19 00 15 06 00 12 05 20 20 05 18 19 00 13 09 24 20 21 18 05 19 00 15 06 00 20 08 05 00 01 02 15 22 05 00 01 14 04 00 19 15 00 06 15 18 20 08 00 20 08 05 00 18 05 03 05 09 22 05 18 00 04 05 03 09 16 08 05 18 19 00 20 08 05 00 20 05 24 20 00 02 25 00 16 05 18 06 15 18 13 09 14 07 00 20 08 05 00 09 14 22 05 18 19 05 00 19 21 02 19 20 09 20 21 20 09 15 14 00 16 18 15 03 05 19 19 00 20 15 00 05 24 20 18 01 03 20 00 20 08 05 00 15 18 09 07 09 14 01 12 00 13 05 19 19 01 07 05 00

- Encrypted ciphertext using **homophonic substitution cipher** (homophonicity = 3):

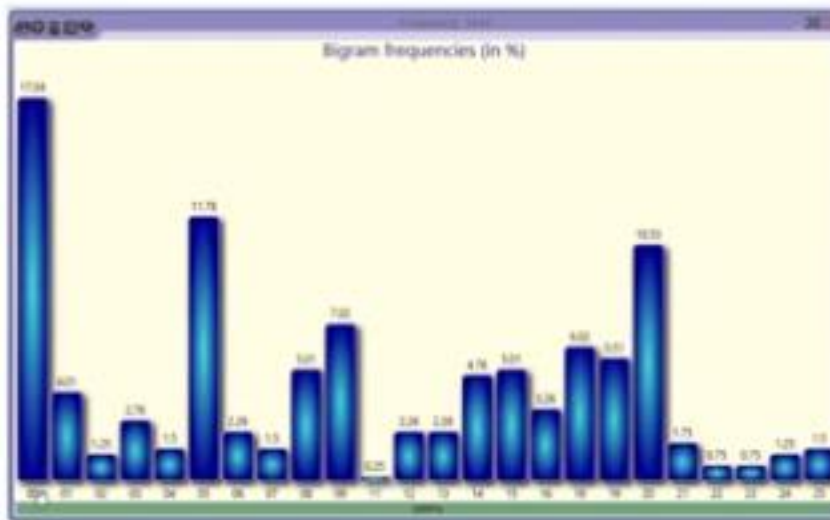
09 14 00 03 18 25 16 20 15 07 44 01 42 08 51 88 27 99 19 21 02 45 46 35 72 47 20 61 41 40 00 29 09 68 34 05 70 88 35 71 99 53 00 13 31 46 60 67 04 88 15 06 99 57 66 55 18 77 16 72 61 14 33 00 09 40 88 23 08 35 03 34 99 73 66 61 20 19 00 41 32 88 42 12 01 09 14 46 05 24 72 99 27 44 31 00 70 57 68 38 53 29 05 30 88 49 35 20 60 99 46 08 31 00 55 61 16 34 57 18 72 05 50 20 88 09 40 99 01 00 56 31 58 35 66 57 04 88 39 27 14 40 05 44 99 75 61 46 60 00 72 08 31 88 34 57 64 42 99 67 06 00 53 88 11 05 25 99 20 60 31 00 21 66 09 46 45 88 65 01 51 99 28 57 00 71 35 14 59 12 05 88 38 31 72 20 57 70 19 99 46 08 05 00 13 15 45 72 88 03 41 39 65 67 40 99 68 27 61 18 71 00 15 32 88 64 31 20 46 57 44 19 99 72 70 09 16 12 05 20 45 00 41 58 88 38 31 46 72 57 18 71 99 13 35 76 20 47 44 05 19 00 67 06 88 46 34 31 99 53 54 15 22 57 00 01 66 30 88 45 41 99 32 67 70 72 60 00 20 08 05 88 18 31 29 57 61 48 05 44 99 56 31 55 09 42 34 57 70 71 00 46 60 05 88 72 31 24 20 99 02 77 00 68 57 18 58 15 44 39 35 14 07 88 46 08 05 99 61 40 74 31 70 19 57 00 45 44 99 75 72 09 20 21 46 35 41 66 88 16 18 67 03 05 19 45 99 72 15 00 31 50 20 44 27 29 46 88 72 34 57 99 41 70 61 33 09 14 53 64 00 65 05 71 19 01 59 31 88



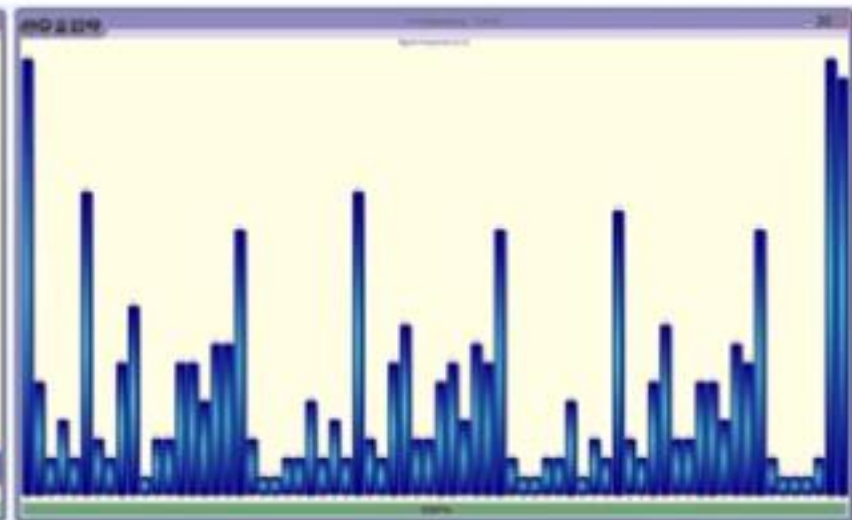


# Homophonic Substitution

Frequencies of two digit combinations



Simple substitution cipher  
27 different ciphertext symbols in total

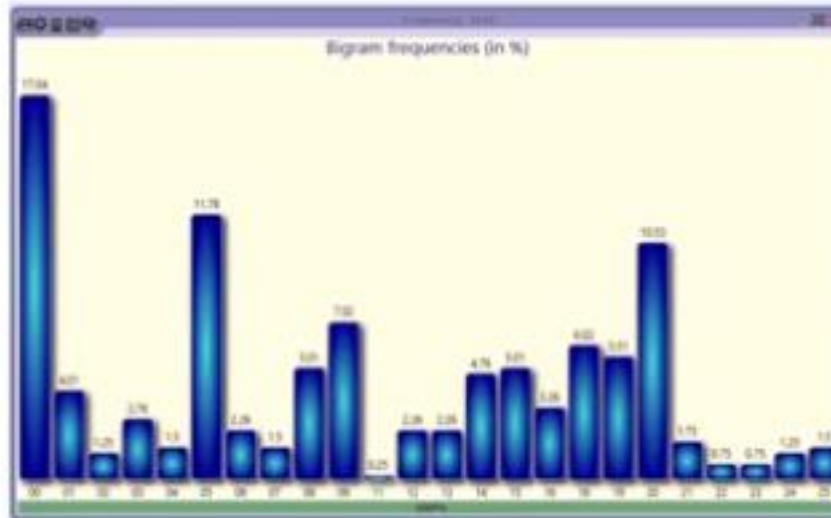


Homophonic substitution cipher – homophonicity = 3  
70 different ciphertext symbols in total

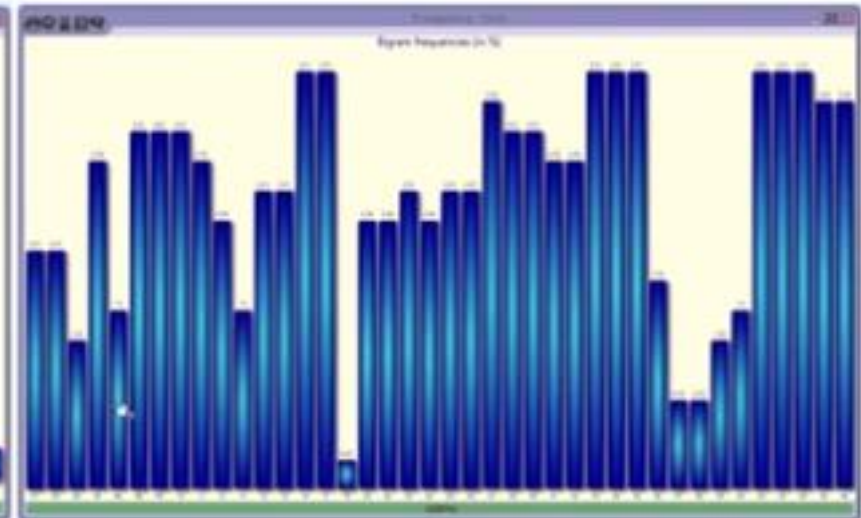


# Homophonic Substitution

Frequencies of two digit combinations



Simple substitution cipher  
27 different ciphertext symbols in total



Homophonic substitution cipher – homophones based on  
language frequencies  
41 different ciphertext symbols in total



# Homophonic Substitution

- Homophonic substitution ciphers were not only built based on single letters
- To improve security, additional cipher elements were introduced
  1. **Doubles / double letters** AA, NN, TT, LL, SS, ...
  2. Other **frequently used bigrams (or higher order n-grams)** TH, HE, IN, EN, NT, RE, ..., ING, THE,
  3. **Nulls** → Cipher elements that encode nothing to
    - a) Further change frequencies
    - b) Confuse attackers
    - c) Mark "special" cipher elements, e.g. nomenclature elements (= code words)
  4. **Complete Words** THE, AND, WITH, TO, BE, OF, FROM, IN, ...
  5. **Code Words** (in nomenclators) The King, the Pope, London, Maximilian, ...
- **Encrypting** (or even showing unencrypted) **word separators** (spaces) **helps the cryptanalyst** to break a cipher
- Creating a code/cipher which **shows separations** of ciphertext symbols **helps the cryptanalyst** to break a code/cipher
- **Ordering in a code/cipher helps the cryptanalyst** to break a code/cipher



# Homophonic Substitution

- Nomenclator (ciphers) contain a **nomenclature** (= code word table)

Example:

| <b>Cipher:</b> |          |    |          |    |          |     |  |
|----------------|----------|----|----------|----|----------|-----|--|
| A              | 01 02 03 | B  | 03 04 05 | C  | 06 07 08 | ... |  |
| AA             | 11 12    | LL | 13 14    | TT | 15 16    |     |  |

| <b>Nulls:</b>                 |  |  |  |  |  |  |  |
|-------------------------------|--|--|--|--|--|--|--|
| 70 71 72 73 74 75 76 77 78 79 |  |  |  |  |  |  |  |

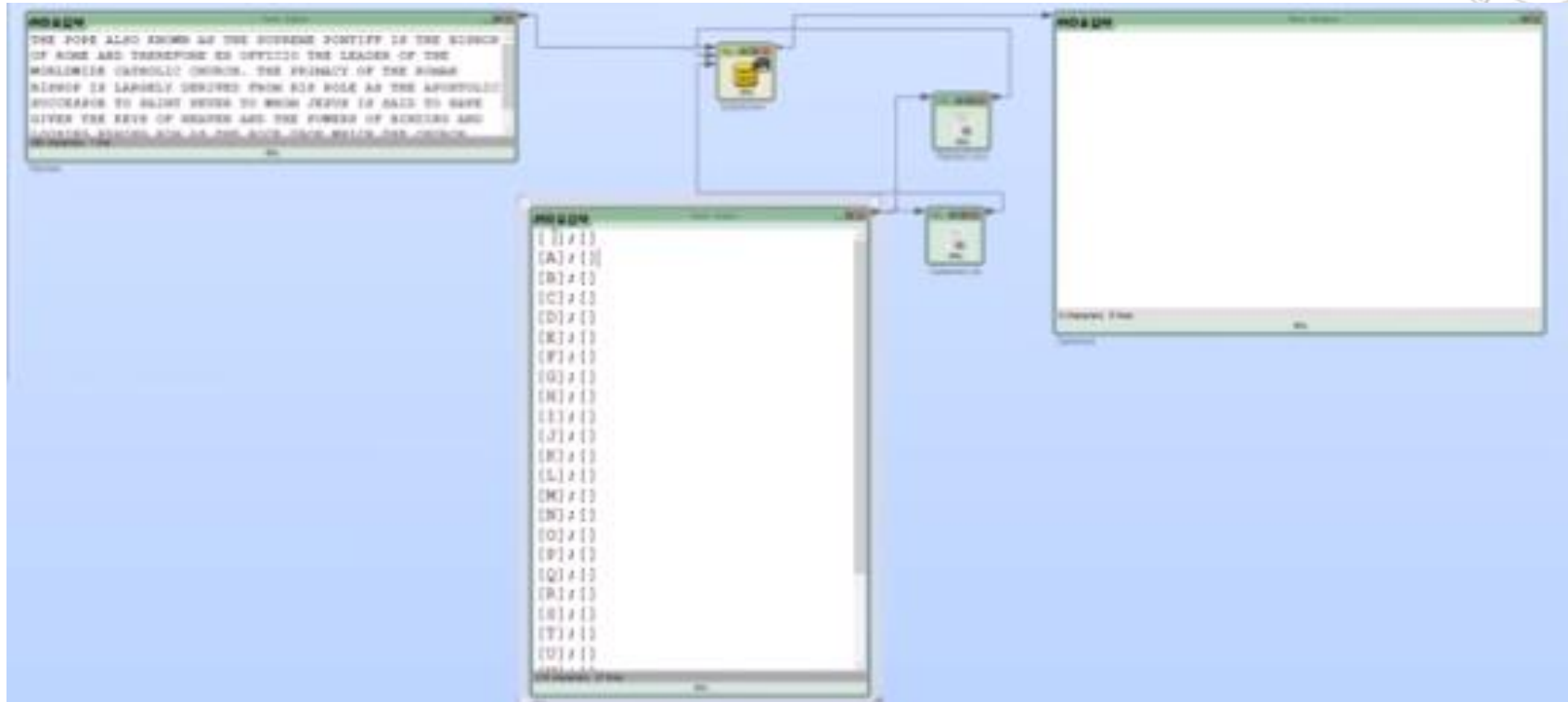
| <b>Nomenclature:</b> |         |     |         |      |         |     |  |
|----------------------|---------|-----|---------|------|---------|-----|--|
| THE                  | 840 841 | AND | 842 843 | WITH | 844 845 | ... |  |

|      |                 |      |       |                  |      |  |  |
|------|-----------------|------|-------|------------------|------|--|--|
| 9100 | The King        | 9200 | One   | We need help     | 9400 |  |  |
| 9101 | The Pope        | 9201 | Two   | We attack at     | 9401 |  |  |
| 9102 | Germany         | 9202 | Three | The enemy is at  | 9402 |  |  |
| 9103 | France          | 9203 | Four  | We will meet at  | 9403 |  |  |
| 9104 | England         | 9204 | Five  | We need urgently | 9404 |  |  |
| 9105 | The Netherlands | 9206 | Six   | We provide cover | 9405 |  |  |
| ...  | ...             | ...  | ...   | ...              | ...  |  |  |

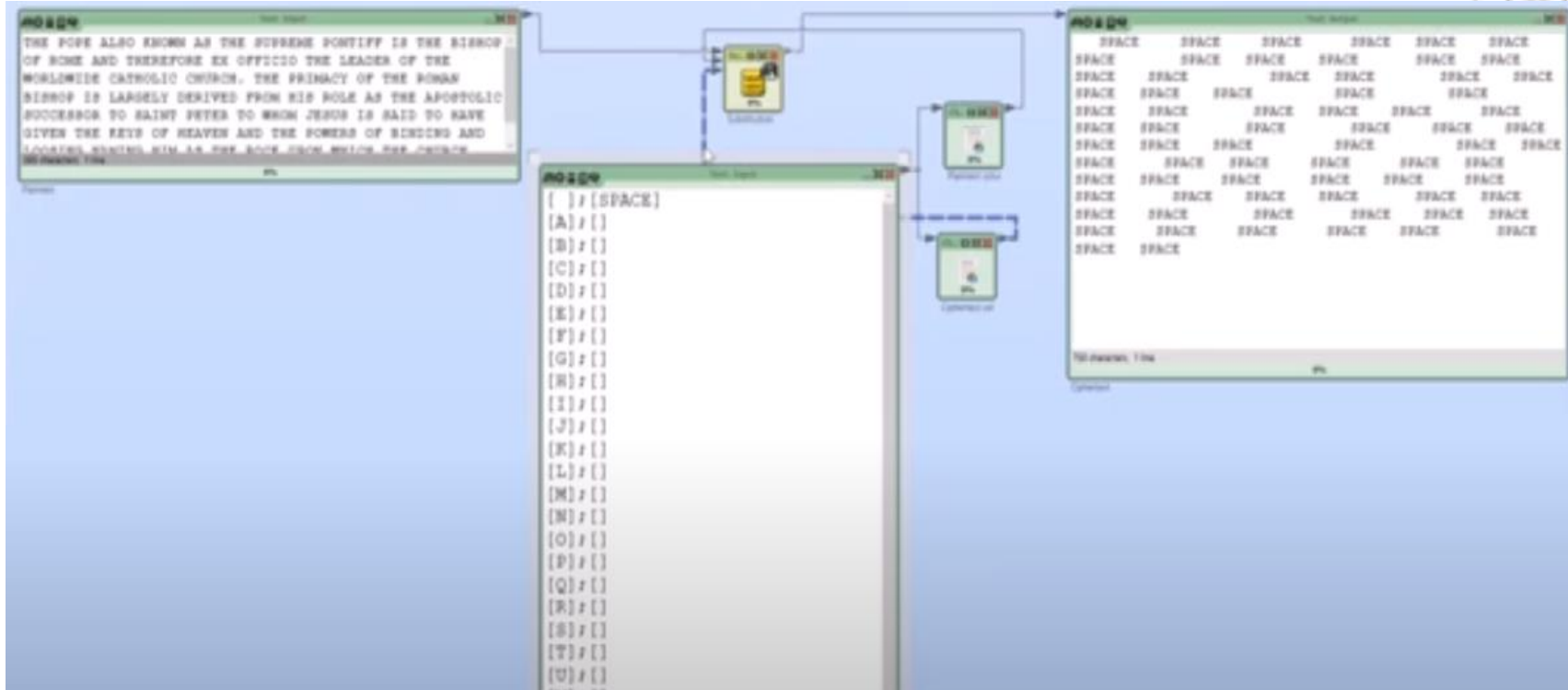


# Homophonic Substitution





# Homophonic Substitution





# Homophonic Substitution

The screenshot displays the Cryptool interface for a homophonic substitution cipher. It consists of several interconnected windows:

- Text Input:** Contains the plaintext: "THE POPE ALSO KNOWN AS THE SUPREME PONTIFF IS THE BISHOP OF ROME AND THEREFORE EX OFFICIO THE LEADER OF THE WORLDWIDE CATHOLIC CHURCH. THE PRIMACY OF THE ROMAN BISHOP IS LARGELY DERIVED FROM HIS ROLE AS THE APOSTOLIC SUCCESSOR TO SAINT PETER TO WHOM JESUS IS SAID TO HAVE GIVEN THE KEYS OF HEAVEN AND THE POWERS OF BINDING AND LOOSING NAMING HIM AS THE ROCK UPON WHICH THE CHURCH".
- Substitution:** A central window showing a mapping table for the cipher.
- Parameter:** A configuration window for the cipher type, currently set to "Substitution".
- Text Output:** Displays the resulting ciphertext: "03 14 05 28 27 43 09 11 21 23 45 25 2 38 09 52 29 23 16 28 26 35 08 05 43 4 40 34 38 03 14 09 17 39 23 52 27 28 01 29 11 42 21 09 14 35 26 05 40 45 41 07 40 17 20 34 45 06 52 35 47 05 15 39 03 14 29 18 45 41 21 21 18 39 39 35 35 17 20 35 14 37 26 20 52 06 14 05 43 4 19 45 40 03 52 09 26 27 08 11 22 34 17 23 21 15 41 12 29 47 19 21 35 26 34 41 27 01 52 39 38 26 45 21 09 11 23 43 27 38 06 45 47 17 20 23 16 35 20 35 38 45 38 11 34 42 09 28 05 06 09 26 03 2 20 29 23 37 38 39 23 38 15 17 21 09 2 51 34 24 05 22 06 52 09 48 29 19 23 4 36 05 42 11 22 39 03 52 09 43 27 18 2 17 39 42 21 17 22 12 15 42 39 21 27 45 42 11 08 39 22 12 14 17 01 15 38 09 5 25 16 28 45 42 18 14 34 20 52 06 14 0 20 14 10 03 16 07 31 24 00 13 03 00 2".

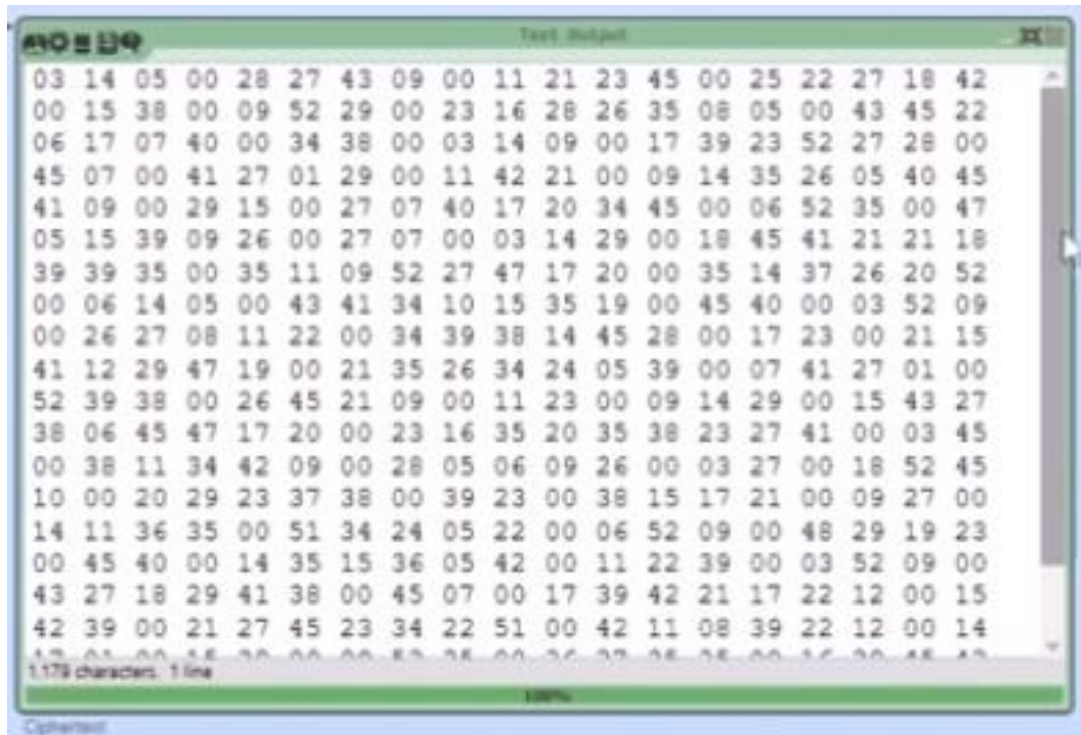
The substitution key window shows the following mappings:

```
[ ]:[ ]
[A]:[11|15]
[B]:[17|34]
[C]:[20|35]
[D]:[21|39]
[E]:[05|09|29|35]
[F]:[07|40]
[G]:[12|51]
[H]:[14|52]
[I]:[17|34|39]
[J]:[20|49]
[K]:[25|48]
[L]:[21|47]
[M]:[08|01|10]
[N]:[22|42]
[O]:[27|45]
[P]:[28|43]
[Q]:[11|44]
[R]:[26|41]
[S]:[23|38]
[T]:[03|09|06]
[U]:[16|37]
[V]:[24|36]
```

# Homophonic Substitution



[ ]# [00]







# Homophonic Substitution

Substitution  
Substitution

Action

Encrypt

Unknown symbols handling

Remove

Symbol Choice

Round Robin

Process polypartit

Unknown Symbol Replacement

?

Input Separator Symbol

Output Separator Symbol

Maximum number of mappings shown

100

Text Output

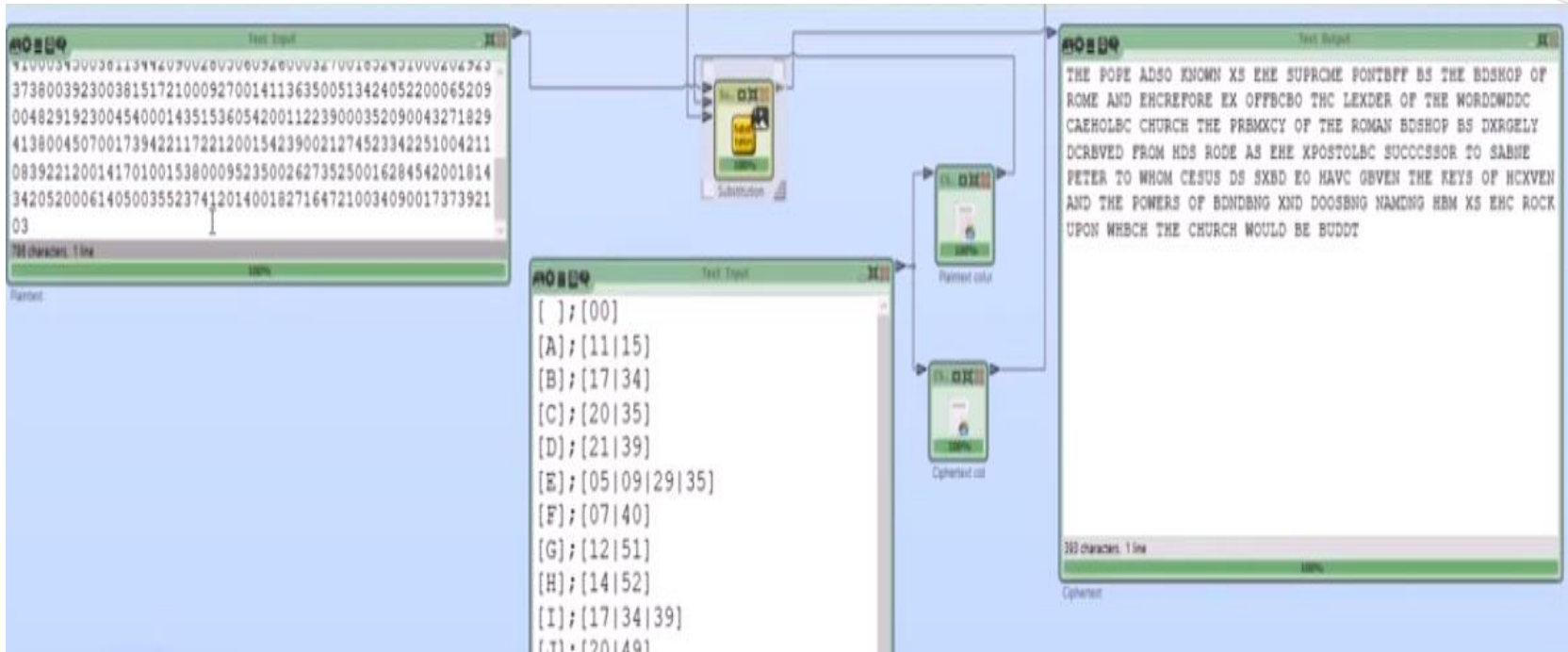
```
031405002827430900112123450025222718420015380009522900231628
263508050043452206170740003438000314090017392352272800450700
412701290011422100091435260540454109002915002707401720344500
065235004705153909260027070003142900184541212118393935003511
095227471720003514372620520006140500434134101535190045400003
520900262708112200343938144528001723002115411229471900213526
342405390007412701005239380026452109001123000914290015432738
064547172000231635203538232741000345003811344209002805060926
000327001852451000202923373800392300381517210009270014113635
005134240522000652090048291923004540001435153605420011223900
035209004327182941380045070017394221172212001542390021274523
342251004211083922120014170100153800095235002627352500162845
420018143420520006140500355237412014001827164721003409001737
392103
```

738 characters, 1 line

Ciphertext



# Homophonic Substitution



# Homophonic Substitution



```
ROZQO Text Input
[ ]:[00]
[A]:[11]
[B]:[17]
[C]:[20|35]
[D]:[21]
[E]:[05|29]
[F]:[07|40]
[G]:[12|51]
[H]:[14|52]
[I]:[34|39]
[J]:[20|49]
[K]:[25|48]
[L]:[47]
[M]:[08|01|10]
[N]:[22|42]
[O]:[27|45]
[P]:[28|43]
[Q]:[44]
[R]:[26|41]
[S]:[23|38]
[T]:[03|09|06]
[U]:[16|37]
[V]:[24|36]
[W]:[18]
[X]:[15]
[Y]:[19]
[Z]:[13]
```

```
ROZQO Text Input
[I]:[34|39]
[J]:[20|49]
[K]:[25|48]
[L]:[47]
[M]:[08|01|10]
[N]:[22|42]
[O]:[27|45]
[P]:[28|43]
[Q]:[44]
[R]:[26|41]
[S]:[23|38]
[T]:[03|09|06]
[U]:[16|37]
[V]:[24|36]
[W]:[18]
[X]:[15]
[Y]:[19]
[Z]:[13]

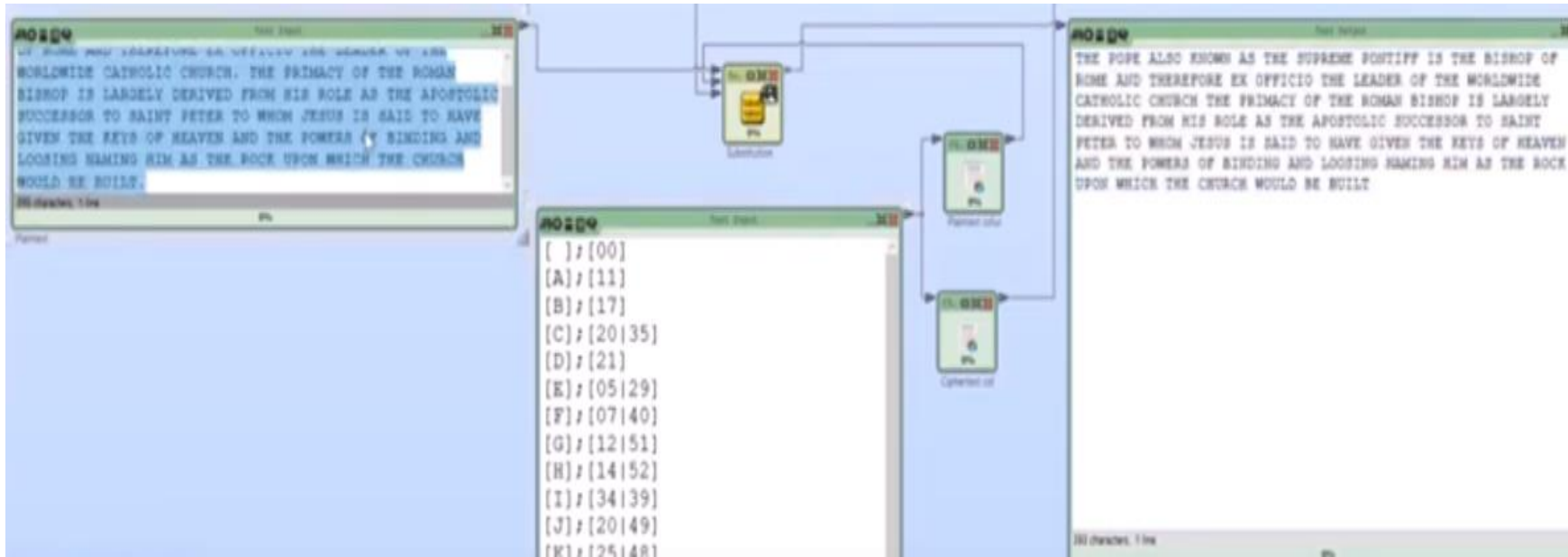
I

[THE]:[601]
[AND]:[602]
[TO]:[603]
[OF]:[604]

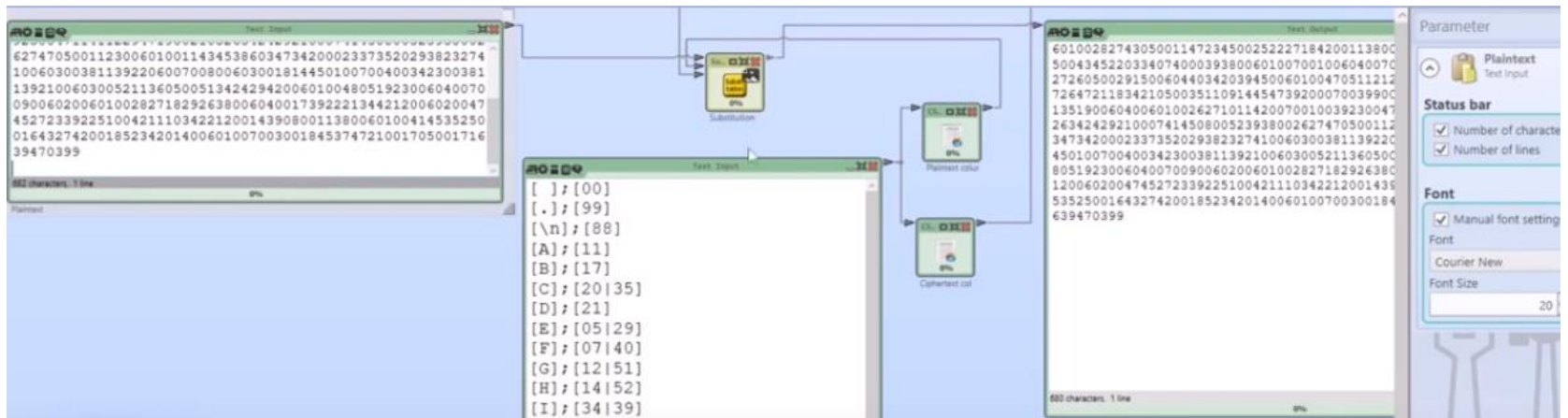
[BISHOP]:[7001]
[ROME]:[7002]
[CHURCH]:[7003]
[JESUS]:[7004]
```



# Homophonic Substitution



# Homophonic Substitution



- Try to break your own code by using the Hill Climbing technique !



# Cryptology for IoT

Modules M4, M6, M8  
Session of 24th May, 2022.

M6.1 Briefing of the session  
M6.2 Friedman Test  
M6.3 Hill Climbing  
**M6.4 Final Exercise using Hill Climbing**

Prof.: Guillermo Botella