|  |
| --- |
| **Protecting Data** |

|  |
| --- |
| **Your Tasks (Mark these off as you go)** |
| Decrypt a message  Decrypt a message encrypted with a random substitution cipher  Apply the XOR algorithm to encrypt and decrypt a message  Watch: The Internet: Encryption and Public Keys  Define key vocabulary  Receive credit for this lab guide |

* **Decrypt a message**

|  |
| --- |
| You have been provided a message which has been encrypted. |
| In the space below, write your encrypted message. |
|  |
| Take 5 minutes and work with your group and try to decode the message. In the space below, write your decoded message. If you were unable to decode your message, that is ok! Just indicate “I have no idea” |
|  |
| Describe the process or techniques your group used to try to decode the message. What information would have been useful for the decoding process? |
|  |

Logo, company name

Description automatically generated

Throughout our daily lives, data is collected. Some data is more sensitive than other data and if not properly protected can be stolen and misused.

Many of the ideas we use to keep our data secret in the digital age are far older than the Internet. The process of encoding data in some secret way is called Encryption.

For example, in Roman times Julius Caesar is reported to have encrypted messages to his soldiers and generals by using a simple alphabetic shift - every character was encrypted by substituting it with a character that was some fixed number of letters away in the alphabet.

As a result, an alphabetic shift is often referred to as the Caesar Cipher. Below are some examples.

|  |  |  |
| --- | --- | --- |
| **Encrypted Message** | **Shift** | **Decrypted Message** |
| serr cvmmn va gur pnsrgrevn | 13 | free pizza in the cafeteria |
| ridiakzqxb qa lwxm | 18 | Javascript is dope |
| oggv og cv ejkrqvng chvgt uejqqn | 24 | meet me at chipotle after school |

|  |  |
| --- | --- |
| Now, navigate to the following link shown to the right | https://studio.code.org/s/hoc-encryption/lessons/1/levels/1 |
| Click on the *Write your own* button, paste your encrypted message in the box, then click the *Add* button. |  |
| Click on the left and right arrows in the Caesar Substitution tab until the message makes sense. |  |
| What is your decoded message? | |
|  | |
| What is the shift? | |
|  | |

* **Decrypt a message encrypted with a random substitution cipher**

With the tool, cracking a Caesar Cipher is easy. Once you've done one, it only takes a matter of seconds to do others.

What if instead of shifting the whole alphabet, we matched every letter of the alphabet to a different random letter of the alphabet? This is called a random substitution cipher.

An example of such random mapping is shown below,



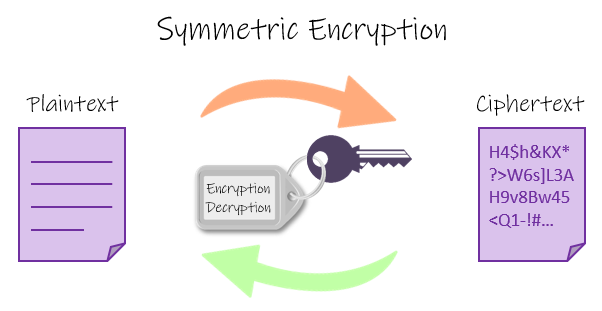
Using the mapping above results in the following encrypted/decrypted pairs,

|  |  |
| --- | --- |
| **Encrypted Message** | **Decrypted Message** |
| rtee nqggo qk fye uorefetqo | free pizza in the cafeteria |
| pomosutqnf qs vbne | javascript is dope |
| ceef ce of uyqnbfxe orfet suybbx | meet me at chipotle after school |
| Now, navigate to the following link shown to the right | https://studio.code.org/s/frequency\_analysis/lessons/1/levels/1 |
| Experiment with the tool: click things, poke around, figure out what it's doing |  |
| How does the widget work? What steps would you take to crack the code? | |
|  | |
| Select *Sample Message (easy)* from the *Message* dropdown,    Then, crack the message using the tips we just talked about   * Find the short words and "crack" them first. How many one-letter words do you know? ("a"). A very common 3-letter word is "the". * Once you've done that, you have substitutes for some of the most common letters. You should be able to use intuition to look at other words with these partial substitutions and make good guesses. * After finding only a handful of hard-fought letters, the rest will tumble quickly. * Comparing the frequencies of letters gives good insight for making sensible guesses.   Copy and paste a portion of the decrypted message below. | |
|  | |

* **Apply the XOR algorithm to encrypt and decrypt a message**

The XOR Encryption algorithm is a very effective yet easy-to-implement method of symmetric encryption. Due to its effectiveness and simplicity, XOR Encryption is an extremely common component used in more complex encryption algorithms used nowadays.

The XOR encryption algorithm is an example of *symmetric encryption* where the same key is used to both encrypt and decrypt a message.



To apply XOR Encryption requires that we first convert the text to be encrypted to binary. Recall, that each character in our alphabet has an ASCII decimal equivalent, which can, in turn, be converted to binary. Below, are the ASCII and binary equivalents of all the letters in our alphabet.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Letter** | **ASCII** | **Binary** |  | **Letter** | **ASCII** | **Binary** |
| A | 065 | 01000001 |  | N | 078 | 01001110 |
| B | 066 | 01000010 |  | O | 079 | 01001111 |
| C | 067 | 01000011 |  | P | 080 | 01010000 |
| D | 068 | 01000100 |  | Q | 081 | 01010001 |
| E | 069 | 01000101 |  | R | 082 | 01010010 |
| F | 070 | 01000110 |  | S | 083 | 01010011 |
| G | 071 | 01000111 |  | T | 084 | 01010100 |
| H | 072 | 01001000 |  | U | 085 | 01010101 |
| I | 073 | 01001001 |  | V | 086 | 01010110 |
| J | 074 | 01001010 |  | W | 087 | 01010111 |
| K | 075 | 01001011 |  | X | 088 | 01011000 |
| L | 076 | 01001100 |  | Y | 089 | 01011001 |
| M | 077 | 01001101 |  | Z | 090 | 01011010 |

The XOR Encryption algorithm is based on applying an XOR mask using plaintext and a key. Consider the following encrypted message which has been encrypted with the following key: 11001100

01000110010100100100010101000101001000000101000001001001010110100101101001000001

To decrypt the message, we need to do the following:

* + First map each digit in the key to every digit in the message. Our key is only 8 digits long, so we need to repeat the key over and over until we reach the end of the encrypted message.
  + For each bit in the encrypted message, if the bit in the key is the same, we decrypt the bit as a 1, if they are different we decrypt the bit as a 0.
  + Each byte (8 bits) represents a letter. Map each by in the decrypted message to the corresponding letter

Below illustrates how to decrypt the first 24 bits of the message

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Encrypted message | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| Key | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Decrypted message | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | F | | | | | | | | R | | | | | | | | E | | | | | | | |

|  |
| --- |
| Consider the encrypted 5-letter word below,  00000101 00011100 00001111 00001111 00010100  Using the key below, decrypt the message.  10101010 |
| |  |  | | --- | --- | | Encrypted 5 letter word | 00000101 00011100 00001111 00001111 00010100 | | 8-digit key repeated | 10101010 10101010 10101010 10101010 10101010 | | Decrypted binary word |  | | 5-letter word |  | |
| Think of a five-letter word. Create an 8-bit key and encrypt your word. |
| |  |  | | --- | --- | | 5-letter word |  | | 5-letter binary word |  | | 8-digit key |  | | Encrypted message |  | |

|  |
| --- |
| In the above examples of XOR encryption, we used an 8-bit key. What is a security concern associated with using an 8-bit repeatable key? How could we make our encryption more secure? |
|  |
| Consider the 8-bit keys we used in the previous examples, how many different values can be represented with 8 bits? |
|  |
| Now consider a 128-bit key which was the basis for encryption in 1999. How many values can be represented with 128 bits? |
|  |
| Most protocols today used 256-bit encryption. How many values can be represented with 256 bits? |
|  |

* **Watch: The Internet: Encryption and Public Keys**

|  |
| --- |
|  |
| <https://youtu.be/ZghMPWGXexs> |

* **Define key vocabulary**

**Encryption**

|  |
| --- |
|  |

**Decryption**

|  |
| --- |
|  |

**Cipher**

|  |
| --- |
|  |

**Caesar’s Cipher**

|  |
| --- |
|  |

**Symmetric Encryption**

|  |
| --- |
|  |

**Asymmetric Encryption**

|  |
| --- |
|  |

* **Receive Credit for this lab guide**

Submit this portion of the lab to Pluska to receive credit for the lab guide.