# Vernam Cypher Bitwise XOR Method

### Recap… Complete the tables

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | AxorB |  |  |  | 25 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 |  |  |  |  | 76 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 |  |  |  |  | XOR (R) |  |  |  |  |  |  |  |
| 1 | 0 |  |  |  |  |
| 1 | 1 |  |  |  |  |

Thus bitwise 25 Xor 76 = \_\_\_\_\_\_ ( in decimal)   
What happens if you bitwise Xor the result (R) with 76 again?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| (R) (25 Xor 76) |  |  |  |  |  |  |  |
| 76 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| XOR |  |  |  |  |  |  |  |

Try this with a couple of other values… check you get the same results.   
(don’t forget you can bitwise XOR integers in VB. There is also the BitXor Function in Excel)

### How used in Vernam (one-time-pad)

As the same pad is used for encryption and decryption hopefully you can see that the symmetrical nature of Xor means the same algorithm can be used. The binary words for the character codes need to be used … See <http://pastebin.com/x6GNWENz> for a VB Vernam program

### Example Encrypt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Msg | H | E | L | L | O |
| Key | M | Y | K | E | Y |
| Msg | 01001000 | 01000101 | 01001100 | 01001100 | 01001111 |
| Key | 01001101 | 01011001 | 01001011 | 01000101 | 01011001 |
| XOR | 00000101 | 00011100 | 00000111 | 00001001 | 00010110 |
| Cypher | 5 | 28 | 7 | 9 | 22 |

### Decrypt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cypher | 5 | 28 | 7 | 9 | 22 |
| Key | M | Y | K | E | Y |
| Cypher | 00000101 | 00011100 | 00000111 | 00001001 | 00010110 |
| Key | 01001000 | 01000101 | 01001100 | 01001100 | 01001111 |
| XOR | 01001000 | 01000101 | 01001100 | 01001100 | 01001111 |
| Dec | 72 | 69 | 76 | 76 | 79 |
| Decrypt | H | E | L | L | O |

### Practice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Msg | Z | E | B | R | A |
| Key | R | A | N | D | O |
| Msg |  |  |  |  |  |
| Key |  |  |  |  |  |
| XOR |  |  |  |  |  |
| Cypher |  |  |  |  |  |
|  |  |  |  |  |  |
| Cypher |  |  |  |  |  |
| Key | R | A | N | D | O |
| Cypher |  |  |  |  |  |
| Key |  |  |  |  |  |
| XOR |  |  |  |  |  |
| dec |  |  |  |  |  |
| Decrypt |  |  |  |  |  |

You have been sent the following message a binary stream. 00010010000110000000101100010101000101010000110100000110 using the key “QWERTYU” decrypt the message.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Key | Q | W | E | R | T | Y | U |
| Cypher | 00010010 | 00011000 | 00001011 | 00010101 | 00010101 | 00001101 | 00000110 |
| Key |  |  |  |  |  |  |  |
| XOR |  |  |  |  |  |  |  |
| dec |  |  |  |  |  |  |  |
| Decrypt |  |  |  |  |  |  |  |

### Program Extentions.

Add a function that creates a file of 1000 random characters call it BigOTP. Make a copy of the file.

Use BigOTP to encrypt and decrypt long messages