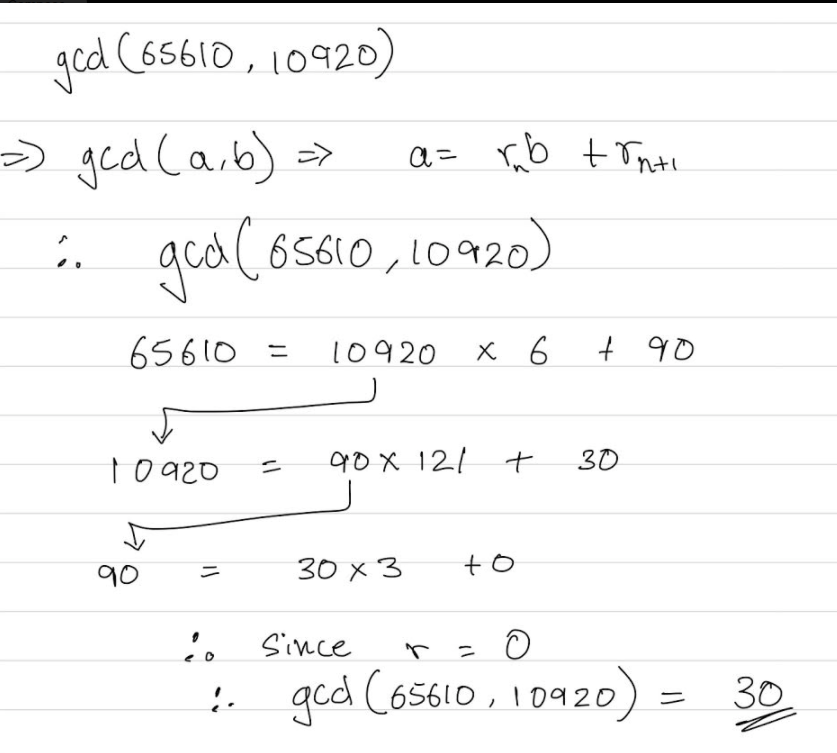
Introduction to Cryptography: Homework 2



1. [Need to find out]
2. Convert text into ASCII representation in bit. Due to the XOR property of a Ꚛ b = c, thus a Ꚛ c = b.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Plaintext ASCII | B | A | R | A | C | K | O | B | A | M | A |
| Plaintext bits | 01000010 | 01000001 | 01010010 | 01000001 | 01000011 | 01001011 | 01001111 | 01000010 | 01000001 | 01001101 | 01000001 |
| Cipher bits | 01000011 | 00011011 | 00010010 | 00110000 | 11111000 | 10100111 | 10001110 | 11101001 | 00010100 | 00011101 | 01100100 |
| Keystream+Nonce | 00000001 | 01011010 | 01000000 | 01110001 | 10111011 | 11101100 | 11000001 | 10101011 | 01010101 | 01010000 | 00100101 |

Since a nonce of 1 was added to each byte for the specific message, we can subtract that one to get the fixed key we are looking for. Moreover, this process doesn’t need to be done since we already know that the second message was encrypted with a nonce of 2. Therefore, the keystream that was used to encrypt the second message is found by adding 1 to the first keystream.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fixed Key + 1 | 00000001 | 01011010 | 01000000 | 01110001 | 10111011 | 11101100 | 11000001 | 10101011 | 01010101 | 01010000 | 00100101 |
| Fixed Key + 2 | 00000010 | 01011011 | 01000001 | 01110010 | 10111100 | 11101101 | 11000010 | 10101100 | 01010110 | 01010001 | 00100110 |

From this, we can now decrypt the second ciphertext from this new keystream. Using the XOR on each bit of this new keystream with ht second ciphertext, the entire plaintext can be achieved.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keystream | 00000010 | 01011011 | 01000001 | 01110010 | 10111100 | 11101101 | 11000010 | 10101100 | 01010110 | 01010001 | 00100110 |
| Ciphertext bits | 01000110 | 00010100 | 00001111 | 00110011 | 11110000 | 10101001 | 10010110 | 11111110 | 00000011 | 00011100 | 01110110 |
| Plaintext bits | 01000100 | 01001111 | 01001110 | 01000001 | 01001100 | 01000100 | 01010100 | 01010010 | 01010101 | 01001101 | 01010000 |

Now, after getting the plaintext bits from above, the plaintext can simply be converted back into ASCII values to fetch the results that is the “readable” text.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Plaintext bits | 01000100 | 01001111 | 01001110 | 01000001 | 01001100 | 01000100 | 01010100 | 01010010 | 01010101 | 01001101 | 01010000 |
| Plaintext ASCII | D | O | N | A | L | D | T | R | U | M | P |

Therefore, the answer is DONALDTRUMP

1. Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clock | S4 | S3 | S2 | S1 | S0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 2 | 1 | 0 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 | 1 | 1 |
| 4 | 0 | 0 | 1 | 0 | 1 |
| 5 | 0 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 1 |
| 7 | 1 | 1 | 0 | 0 | 0 |
| 8 | 0 | 1 | 1 | 0 | 0 |
| 9 | 1 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 | 1 |
| 11 | 0 | 0 | 1 | 0 | 1 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 13 | 1 | 0 | 0 | 0 | 1 |
| 14 | 1 | 1 | 0 | 0 | 0 |
| 15 | 0 | 1 | 1 | 0 | 0 |
| 16 | 1 | 0 | 1 | 1 | 0 |
| 17 | 0 | 1 | 0 | 1 | 1 |
| 18 | 0 | 0 | 1 | 0 | 1 |
| 19 | 0 | 0 | 0 | 1 | 0 |
| 20 | 1 | 0 | 0 | 0 | 1 |
| 21 | 1 | 1 | 0 | 0 | 0 |
| 22 | 0 | 1 | 1 | 0 | 0 |
| 23 | 1 | 0 | 1 | 1 | 0 |
| 24 | 0 | 1 | 0 | 1 | 1 |
| 25 | 0 | 0 | 1 | 0 | 1 |
| 26 | 0 | 0 | 0 | 1 | 0 |

1. The polynomial is:
2. According to the table formed above, it can be seen that from the clock value 6 onwards, the keystream bits have a constant repetition of 1000110. Thus there would be 6 keystream bits generated before this repetition occurs.
3. The key stream bits before the repetition: 110110

And the key stream bits that is the repetition: 1000110.