CS 4371

Jason McKinnerney

Jlm573

Homework 5

1. (15%) Decipher the following ciphertext, which was encrypted with the Caesar cipher: TEBKFKQEBZLROPBLCERJXKBSBKQP. Follow the example on pages 13-16 of lec06.pdf to find the key and the plaintext.

(a) Make a program to compute the table of correlation Phi(i) for this question. The table looks like the table on page 16 of lect06.pdf. But no need to submit your program. Show the table in your report.

(b) Show the key and the plaintext.

{'T': 1, 'E': 3, 'B': 5, 'K': 4, 'F': 1, 'Q': 2, 'Z': 1, 'L': 2, 'R': 2, 'O': 1, 'P': 2, 'C': 1, 'J': 1, 'X': 1, 'S': 1}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Let | T | E | B | K | F | Q | Z | L | R | O | P | C | J | X | S |
| Occ | 1 | 3 | 5 | 4 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 |
| Prob | 0.0357 | 0.1071 | 0.1786 | 0.1429 | 0.0357 | 0.0714 | 0.0357 | 0.0714 | 0.0714 | 0.0357 | 0.0714 | 0.0357 | 0.0357 | 0.0357 | 0.0357 |

|  |  |
| --- | --- |
| Shift (i) | Correlation |
| 0 | 0.036 |
| 1 | 0.0427 |
| 2 | 0.0344 |
| 3 | 0.0366 |
| 4 | 0.0355 |
| 5 | 0.03 |
| 6 | 0.0408 |
| 7 | 0.0347 |
| 8 | 0.0422 |
| 9 | 0.0378 |
| 10 | 0.046 |
| 11 | 0.0415 |
| 12 | 0.0371 |
| 13 | 0.0423 |
| 14 | 0.0367 |
| 15 | 0.0307 |
| 16 | 0.038 |
| 17 | 0.0428 |
| 18 | 0.0322 |
| 19 | 0.0419 |
| 20 | 0.0271 |
| 21 | 0.0254 |
| 22 | 0.0408 |
| 23 | 0.0686 |
| 24 | 0.0432 |
| 25 | 0.0345 |

KEY = 23

WHENINTHECOURSEOFHUMBNEVENTS

2. (10%) Let k be the encryption key for a Caesar cipher. Then, the decryption key is 26-k. One of the characteristics of a public key system is that the encryption and decryption keys are different. Why is the Caesar cipher not a public key system, even though its encryption and decryption keys are different?

The issue is that if you have access to one of the keys, the other key is basically not needed due to the process of cracking the code can be done fairly quickly and easily. This then does not allow for security to be enforced on the system.

3. (15%) We have a law: (ab)%n = ((a%n)(b%n))%n. We want to compute (35^77)%83, i.e. 35 powers to 77 then modulo over 83.

(a) Show how to use the law to reduce the number of multiplications of this computation from 76 multiplications to 9 multiplications.

1 35^77 %83 == 35^(64 + 8 + 4 + 1) % 83

2 35^1 % 83 = 35

3 35^2 % 83 = 63

4 35^4 % 83 == 63 \* 63 % 83 = 68

5 35^8 % 83 == 68 \* 68 % 83 = 59

6 35^16 % 83 == 59 \* 59 % 83 = 78

7 35^32 % 83 == 78 \* 78 % 83= 25

8 35^64 % 83 == 25 \* 25 % 83 = 44

9 35^77 % 83 == ((44)(59)(68)(35))%83 == 43

(b) Make a C or C++ or Java program to implement an integer exponentiation function dexp(unsigned int x, unsigned int y, unsigned int n) that returns (x^y)%n with reduced multiplications. Copy and paste your code in your homework submission.

A screenshot of a computer

Description automatically generated with medium confidence

(c) Use your program to get the result of (35^77)%83. Show the result.

PS C:\Users\jason\Desktop\TXState\Fall2021\Security> cd "c:\Users\jason\Desktop\TXState\Fall2021\Security\" ; if ($?) { g++ exp.cpp -o exp } ; if ($?) { .\exp }

Answer: 43

4. (10%) A program exclusive or's all bytes in its input to produce a one-byte hash. Is this program a secure hash function or not? Show one example of inputs to justify your answer.

No it is not. If it xor’s two values into 1 byte then the system can replicate itself such that values repeat and the hash does not provide protection:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |

5. (10%) Multiply two large numbers p and q (you will need to find a tool or a library by yourself)

p = 0xc315d99cf91a018dafba850237935b2d981e82b02d994f94db0a1ae40d1fc7ab9799286ac68d620f1102ef515b348807060e6caec5320e3dceb25a0b98356399

q = 0xe90bbb3d4f51311f0b7669abd04e4cc48687ad0e168e7183a9de3ff9fd2d2a3a50303a5109457bd45f0abe1c5750edfaff1ad87c13eed45e1b4bd2366b49d97f

0XB197D3AFE713816582EE988B276F635800F728F118F5125DE1C7C1E57F2738351DE8AC643C118A5480F867B6D8756021911818E470952BD0A5262ED86B4FC4C2B7962CD197A8BD8D8AE3F821AD712A42285DB67C85983581C4C39F80DBB21BF00DBD2AE9709F7E307769B5C0E624B66144C1DDB62EF1FE7684BBE61D8A19E7

6. (10%) Factorize a short large number (you will need to download and use the tool yafu)

N=359567260516027240236814314071842368703501656647819140843316303878351

Prime =

|  |
| --- |
| 17963604736595708916714953362445519 |
| 20016431322579245244930631426505729 |

=======

All CTF docker images are at https://userweb.cs.txstate.edu/~qg11/ctf/

No partial points will be assigned to the CTF questions.

7. (10%) Get the CTF docker image of "10.caesar". Show the screen shot when you catch the flag. You may read the partial solution.

Graphical user interface

Description automatically generated

8. (10%) Get the CTF docker image of "11.aes". Show the screen shot when you catch the flag. You may read the partial solution.

Text

Description automatically generated

Text

Description automatically generated

Flag = ‘looks like you can break aes’

9. (10%) Get the CTF docker image of "12.rsa". Show the screen shot when you catch the flag. You may read the partial solution.

A screenshot of a computer

Description automatically generated with medium confidence

Flag = RSA\_is\_NOOB