Introduction to Cryptography, 2021 Fall

Homework 4, due 4pm, 12/2/2021 (Thursday)

Part 1: Written Problems

1. Consider to use RSA with a known key IK to construct a cryptographic hash function H as follow: Encrypt the first block, XOR the result with the second block and encrypt again, etc. Then, the last ciphertext block is the hash value. For example,
   $$H(M_1M_2) = \text{Enc}(IK, \text{Enc}(IK, M_1) \oplus M_2) = h.$$
   Show that this H does not satisfy the property of second image resistance. That is, we can find N_1 and N_2 such that $H(N_1N_2)=h$.
2. Do convolution on the function $\sin 2\pi\left(\frac{f}{8}\right)x$ and the 8-sample vector $\vec{a} = [0\ 1\ 0\ 3\ 0\ 1\ 0\ 3]$ for f=0, 1, 2, 3.
3. Use the continued fraction method to find a rational number to approximate e with accuracy up to 3 decimal digits under the decimal point.
4. Use the DFT method to factor M=39 by choosing a=7. We sample N=1024 points for $g(x) = a^x \bmod M$. Use an online tool or Matlab to compute DFT.
   a) Show all steps of computation.
   b) What is the probability of the frequencies of form $[\frac{kN}{s}]$ in the result of DFT, where k is an integer and s is the period of g(x).

Part 2: Programming Problem

This programming problem is to simulate the bitcoin mining of computing hash values with some specific form. Note that this is not the real bitcoin mining. It only verifies the difficulty of finding hash values with many leading zeros. Use Crypto++ for computing sha256.

A. Build the blockchain in the following table. Start with the initial hash value that is Sha256(YouID). For example, if your ID is "Bitcoin", then the initial hash value is Sha256("Bitcoin") = `B4056DF6691F8DC72E56302DDAD345D65FEAD3EAD9299609A826E2344EB63AA4`

The mining record is kept as following:

# of leading zeros	Preimage = Previous hash (in Hex)+ Nonce (32 bits, in Hex)	Hash value (in Hex) with a specified number of leading zeros
0	Previous hash (Hex): **B4056DF6691F8DC72E56302DDAD345D65FEAD3EAD9299609A826E2344EB63AA4** Nonce (Hex): 00000000	**2767667C2AF3BE01EFAC4FB387EC27C10B9D3BEE9C5D48CFF4CFB9F523560B24**
1	Previous hash (hex): **2767667C2AF3BE01EFAC4FB387EC27C10B9D3BEE9C5D48CFF4CFB9F523560B24** Nonce (Hex): 0000000A	**0DE32E85C2AC9D96659D42C8A3EA3D2C05FDE384B468E6EFE062B6E21288CBCA**

2	Previous hash (hex): **0DE32E85C2AC9D96659D42C8A3EA3D2C05FDE384B468E6EFE062B6E21288CBCA** Nonce (Hex): 000001e3	**00EE1063B3EB05C11A21D0F6302ABC473FEF1F97686DA8F44C73C9575FD842B7**
3	?	?
…	?	?

B. Submission: you need to upload two files: hashchain.cpp and out.txt, where out.txt contains the chain in the form (the number of leading zeros, previous hash, nonce, current hash, …):

0
B4056DF6691F8DC72E56302DDAD345D65FEAD3EAD9299609A826E2344EB63AA4
00000000
2767667C2AF3BE01EFAC4FB387EC27C10B9D3BEE9C5D48CFF4CFB9F523560B24
1
2767667C2AF3BE01EFAC4FB387EC27C10B9D3BEE9C5D48CFF4CFB9F523560B24
0000000A
0DE32E85C2AC9D96659D42C8A3EA3D2C05FDE384B468E6EFE062B6E21288CBCA
…

C. Grading: the more leading zeros your hash values have, the higher your grade is.