1172/DCP1323 Introduction to Cryptography, Spring 2019 **Due: 2019/5/20 (Mon.)**

Homework 5

Instructor: Prof. Wen-Guey Tseng Scribe: Hung-Yu Chiu

Part 1: Written Problems

1. Solve the following problems
   A. Consider the Davies and Price hash code scheme described in Section 11.4 and assume that DES is used as the encryption algorithm:
   $$H_i = H_{i-1} \oplus E(M_i, H_{i-1})$$
   Recall the complementarity property of DES (Problem 3.14): If $Y = E(K, X)$, then $Y' = E(K', X')$. Use this property to show how a message consisting of blocks $M_1, M_2, \ldots, M_N$ can be altered without altering its hash code.
   B. Show that a similar attack will succeed against the scheme proposed in [MEYE88]:
   $$H_i = M_i \oplus E(H_{i-1}, M_i)$$

2. Now consider the opposite problem: using an encryption algorithm to construct a one-way hash function. Consider using RSA with a known key. Then process a message consisting of a sequence of blocks as follows: Encrypt the first block, XOR the result with the second block and encrypt again, etc. Show that this scheme is not secure by solving the following problem. Given a two-block message B1, B2, and its hash
   $$\mathrm{RSAH}(B1, B2) = \mathrm{RSA}(\mathrm{RSA}(B1) \oplus B2)$$
   Given an arbitrary block C1, choose C2 so that $\mathrm{RSAH}(C1, C2) = \mathrm{RSAH}(B1, B2)$.
   Thus, the hash function does not satisfy weak collision resistance.

3. DSA specifies that if the signature generation process results in a value of s = 0, a new value of k should be generated and the signature should be recalculated. Why?

4. It is tempting to try to develop a variation on Diffie–Hellman that could be used as a digital signature. Here is one that is simpler than DSA and that does not require a secret random number in addition to the private key.

Public elements:	q	prime number
	α	$\alpha < q$ and α is primitive root of q
Private key:	X	$X < q$
Public key:	Y	$Y = \alpha^X \bmod q \bmod q$

   To sign a message M, compute $h = H(M)$, which is the hash code of the message. We require that $\gcd(h, q-1) = 1$. If not, append the hash to the message and calculate a new hash. Continue this process until a hash code is produced that is relatively prime to $(q-1)$. Then calculate Z to satisfy $Z \equiv X \times h \pmod{q-1}$. The signature of the message is $\sigma = \alpha^Z$. To verify the signature, a user compute t such that $t \times h = 1 \bmod (q-1)$ and verifies $Y = \sigma^t \bmod q$. Show that the scheme is unacceptable by describing a simple technique for forging a user's signature on an arbitrary message.

改成Y = σ ^ t mod q ...

5. Assume a technique for a digital signature scheme using a cryptographic one-way hash function (H) as follows. To sign an n-bit message, the sender randomly generates in advance 2n 64-bit cryptographic keys: $k_1, k_2, \dots, k_n, k'_1, k'_2, \dots, k'_n$, which are kept private. The sender generates the following two sets of validation parameters which are made public.

$$v_1, v_2, \dots, v_n \; and \; v'_1, v'_2, \dots, v'_n$$

where

$$v_i = \mathrm{H}(k_i||0), \qquad v'_i = \mathrm{H}(k'_i||1)$$

The user sends the appropriate k_i or K'_i according to whether M_i is 0 or 1 respectively.
For example, if the first 3-bits of the message are 011, then the first three keys of the signature are k_1, k'_2, k'_3

A. How does the receiver validate the message?
B. Is the technique secure?
C. How many times can the same set of secret keys be safely used for different messages?
D. What, if any, practical problems does this scheme present?

Part 2: Programming Problem

This programming problem is to simulate the bitcoin mining. Note that this is not the real bitcoin mining. It only verifies the difficulty of finding hash values with many leading zeros. You can use either OpenSSL or Crypto++. Nevertheless, using Crypto++ in Visual Studio is preferred. Please find the related library information and examples on the Internet.

I. Do sha2-256 (that is, sha256) on the following text, where the first row is the test sample:

Message (in ASCII)	Message digest (in Hex)
"Hello!"	`334d016f755cd6dc58c53a86e183882f 8ec14f52fb05345887c8a5edd42c87b7`
"Bitcoin is a cryptocurrency, a form of electronic cash."	?

II. Mine cryptocurrency:

A. Build the blockchain in the following table until the requirement of leading zeros cannot be met. We start with the hash value Sha256("Bitcoin") = `B4056DF6691F8DC72E56302DDAD345D65FEAD3EAD9299609A826E2344EB63AA4`

# of leading zeros	Preimage = Previous hash (in Hex)+ Nonce (32 bits, in Hex)	Hash value (in Hex), with the specified leading zeros
0	**B4056DF6691F8DC72E56302DDAD345D6 5FEAD3EAD9299609A826E2344EB63AA4** 00000000 <= nonce	**2767667C2AF3BE01EFAC4FB387EC27C1 0B9D3BEE9C5D48CFF4CFB9F523560B24**
1	**2767667C2AF3BE01EFAC4FB387EC27C1 0B9D3BEE9C5D48CFF4CFB9F523560B24** 0000000A	**0DE32E85C2AC9D96659D42C8A3EA3D2C 05FDE384B468E6EFE062B6E21288CBCA**
2	?	?
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.

D. On-site test: Will announce the venue and schedule later. You need to pass the onsite test to get the credits of this programming problem.

.