# Practice Questions

CS161 Computer Security, Fall 2008

Name _____

Email address _____

Score _____ % / 100 %

Please do not forget to fill up your name, email in the box in the midterm exam – you can skip this here. These practice questions are **not** intended to be comprehensive for preparation or too closely representative of the midterm questions in any way. They are mereley intended to be used for initiating discussions, and help you identify your personal weaknesses that you need to emphasize on during preparation.

## 1. Terms Definitions

. Explain the following terms, in no more than 2 sentences.

(a) *(1 point)* Nonce

(b) *(1 point)* (n,t) threshold secret sharing scheme.

(c) *(1 point)* Known-plaintext attack

(d) *(1 point)* Time Of Check to Time Of Use bug.

## 2. True/False with Reason

. For each of the statements below, state whether they are true or false. Explain your answer in at most 2 sentences.

(a) *(2 points)* T/F: Suppose Alice generates a RSA public and private key pair, and publishes the public key. Then that's all that is needed for Alice to be able to receive a securely encrypted email from Bob.

(b) *(2 points)* T/F:If a cryptographic hash function is second preimage resistant, it implies that it is also collision resistant.

(c) *(2 points)* Let g be a generator for a prime p. Recall that the computational Diffie-Hellman problem is the task of computing $g^{ab}\ mod\ p$, given $g^a\ mod\ p$ and $g^b\ mod\ p$, where $a$ and $b$ are random exponents between 0 and $(p-1)$.

T/F: If the discrete logarithm problem can be efficiently solved then the computational Diffie-Hellman problem can be efficiently solved.

(d) *(2 points)* T/F: If we discover an efficient algorithm to compute the GCD of two extremely large numbers, this will make it possible to break RSA.

## 3. Hash Functions and MAC

A Message Authentication code (MAC) consists of two algorithms (S,V). Algorithm S(k,m) uses a secret key k to generate an integrity tag for message m. Alogrithm V(k,m,t) uses a secret key k to validate a given integrity tag t for m.

Recall that a cryptographic hash function $h$ is a non-keyed function that outputs a short hash $h(m)$ for an input message $m$. A hash function is said to be *collision resistant* if it is difficult to find two distinct messages $m_0$, $m_1$ such that $h(m_0) = h(m_1)$.

Let us consider two mechanisms for providing file integrity for a single file F on disk. The file system should be able to detect any unauthorized modification to this file. We say that the system is secure if an attacker cannot modify F without being detected. You can assume that the owner of the file F has a password known to the system, but not to the attacker.

**Integrity Protection Method** : Compute an integrity tag for the file F and store the integrity tag in header of F. Upon file open, the file system checks that the integrity tag is valid.

(a) *(4 points)* Suppose the integrity tag is computed using a collision resistant hash function applied to F. Validation of the integrity tag upon file open is done by rehashing the file and comparing the result to the value in the header of F. Is the system secure? Explain.

(b) *(4 points)* Suppose the integrity tag is computed as the MAC of F using the user's password as the MAC secret key. Is the resulting system secure? Explain.

(c) *(4 points)* Outline changes to the Integrity Protection method, detailing how to store integrity tags and how to compute and validate them using hashes and/or MAC only, such that the resulting system is secure.

## 4. Signatures and Encryption

Suppose Alice and Bob are employees of a company 'foobar'. All employees of the company use RSA for encrypting and signing their emails. RSA public keys of all employees are stored in a file on the company's server. The contents of this file are available for public view on foobar's non-SSL enabled key download web site, hosted at `http://foobar.com/pubkeys.html`.

The email client software used internally in the company has an email signing feature. It works as follows. When the user clicks on the 'Sign and Send' button in his email client, the client software appends a special 'signature' tag to the regular email header before sending the email. The 'signature' tag contains the author's name and date signed by the author's RSA private key.

The receiver's email client software automatically verifies the signature tag, if present, by first obtaining the RSA public key of the sender from the foobar's key web site. It then verifies that the email sender's name and the date in the regular email header matches the signed name and date in the signature tag.

(a) *(4 points)* Suppose Alice receives a signed, but unencrypted email from Bob saying that he has approved her month long vacation. Explain why the email client gives a completely false sense of security, by outlining two different[1] attacks that allow Mallory to send such an email and make it appear as if it is sent by Bob. You can assume that email header fields are spoofable (i.e Mallory can alter the date and the sender fields in a regular email header to reflect that someone else has sent email).

Attack 1:

Attack 2:

[1] 'different' here means that there are separate fixes for each attack

(b) *(4 points)* Suggest two fixes, one for each attack described in part (a)

Fix for attack 1:

Fix for attack 2:

(c) *(3 points)* Suppose a curious employee, Eve, is able to break into the server and alter the file containing RSA public keys. How should Eve alter that file so that she can read confidential, encrypted emails sent between Alice and Bob.

(d) *(3 points)* How can Alice and/or Bob detect Eve's intrusion.

## 5. Authentication Protocols

Alice and Bob share a secret key K. Suppose that someone suggests the following methods to allow Alice to securely authenticate to Bob.

(a) *(5 points)* Bob generates a random message r, enciphers it using K under a secure block cipher scheme and sends the encrypted message to Alice. Alice decrypts it, adds 1 to it and sends the result encrypted with K under the same block cipher to Bob. Bob decrypts the message and compares it with r. If the difference is 1, then he is sure he is communicating with Alice; or else, as no one else knows the secret K, he is talking to an impersonator. Is this protocol secure? Why or why not.

## 6. Reasoning about Code

Consider the following program and postcondition:

```
1. i = n;
2. product = 1;
3. while (i>k) do
4.    product = product * i;
5.    i = i-1;
6.    end
7. /* Post-Condition : {product = (n!/k!)} */
```

You may assume all values are integers. The operator `!` in the above code denotes factorial.

(a) *(3 points)* Determine the loop invariant $I$, that holds true at the end of each iteration of the loop. $I$ must include relation between variables `i`, `n`, `product`, and `k`.

(b) *(3 points)* Write the precondition $P$ as a logical expression, that must hold true at the start of the program for the postcondition on line 7 to hold true.

(c) *(2 points)* Argue that the program fragment always terminates, for any input that satisfies $P$.

## 7. Program Errors

This question is intended to give you practice at spotting program errors.
Identify as many errors as possible.

```
/* EncryptedLength: Returns the maximum number of bytes that 'src'
                    will be encrypted to (not including any NULL-termination). */
int EncryptedLength(uint8_t *src, uint32_t srclen);

/* Encrypt8bit: Encrypts 'src' into 'dest', using 'key'. Block size is 8
                bits. Does not NULL-terminate. Returns the size of the
encrypted string on success or -1 on failure. */
int Encrypt8bit (uint8_t *src, char * dest, , uint32_t srclen, uint32_t destsize, char key);

/* Encrypt: Encrypts the binary data in 'data' into a NULL-terminated
            character string returned in 's'. Returns size of encrypted string
    on success, 0 on failure. */
int Encrypt (uint8_t *data, char **s, size_t dataLen)
{
  Bool succeeded;
  int size;

  *s = (char *) malloc(EncryptedLength(data, dataLen));

  size = Encrypt8bit(data, s, dataSize, EncryptedLength (data, dataLen), 0xBC);

  if (size < 0) {
    free (s);
    size = 0;
    goto terminate;
  }

  succeeded = size;

  terminate:
  return succeeded;
}
```