CS 526 **Homework 1** **Fall 2016**

Question 1. (26 points) This question is about the use of public and private salts in password files, as explained in class (and on slide 11 of the online notes on authentication). The adversary X has managed to obtain the password file for a system that uses p bits of public salt and s bits of private salt. The number of user records in the password file is N. X carries out a dictionary attack against all the password file, hoping to figure out the weak passwords in it. The dictionary that X uses has D candidate passwords in it (in clear text, i.e., neither hashed nor encrypted). For each of the following two cases, describe how X can best attack the password file, and state how many cryptographic hash computations the attack would involve (assuming that the attack ends in failure, i.e., that none of the N passwords is in the dictionary). For each case, first give an answer as a function of the symbols p, s, N, and D (an exact answer – no "big oh" notation), then fill numerical values for p, s, N and write the answer as a function of D.

1. $p = s = 12$, $N = 500{,}000$.
2. $p = s = 12$, $N = 60$.

Question 2. (24 points) Suppose your employer asks you to add a private-salt capability to a legacy system that does not currently support private salt. The legacy system does not limit the number of allowed failed login attempts (i.e., no user gets locked out because of having entered the wrong password too many times), and you need to maintain this "no lockout" property when adding the private salt capability. You are asked to avoid modifying the internals of the legacy system, by writing a front-end to it, for both the login and the password selection and modification. Of the following possible choices for the contents of a user entry in the password file, some are more compatible than others with the task you were assigned. Identify the choices that are more compatible with your assignment, and explain in detail why.

```
Choice 1: username, salt, h( password, private_salt, salt )
Choice 2: username, salt, h( password, salt, private_salt )
Choice 3: username, salt, h( salt, password, private_salt )
Choice 4: username, salt, h( salt, private_salt, password )
Choice 5: username, salt, h( private_salt, password, salt )
Choice 6: username, salt, h( private_salt, salt, password )
```

Question 3. (26 points) A company issues each employee A a small calculator-like device that has a unique and secret 4-digit integer S_A stored in it. That S_A is known only to: (i) the employee A who is given that device; and (ii) the corporate authentication server that grants or denies access requests by employees (it stores S_A along with the username for A in the password file). No two devices have the same secret S_A, hence an employee's S_A is not known to any other employee. A device has a small 10-key numeric keypad (0 to 9), a display, and operates as follows: When user A enters S_A, the device computes and displays a one-time password to be used by A, by carrying out a cryptographic computation that depends only on the entered S_A and on the *previous* one-time password (call it P_A). The P_A

entry stored in the device is updated only if the entered S_A matches the internally stored S_A (otherwise P_A is left unchanged, because the displayed one-time password will then result in a failed login attempt). The authentication server stores, along with the username for A and the S_A, the previous one-time password successfully used by A (i.e., P_A), and it can therefore carry out the exact same computation as the one done in the user's device, so that access is granted only if the server-computed result matches the device-computed result (as entered by the user after the device displays it). Only if the correct S_A is entered does the P_A entry get updated (in both the device and the server): If the user enters the wrong S_A then the device displays an answer that differs from what the server computes, and therefore the login attempt fails and the server does not update its stored P_A value. Assume that a user's device is reliable and sturdy enough to never need replacement, that changing its battery does not erase its contents, and that any attempt at physical tampering is detected by the device and causes it to self-destruct (by erasing its contents). Answer the following questions about this scheme.

1. Suppose someone steals user A's device and attempts to promptly use it to gain unauthorized access before A reports the loss (the loss is certain to be reported within the next 24 hours). The thief knows A's username but has no prior knowledge of S_A (i.e., no shoulder-surfing ever happened). Corporate policy is to tolerate at most 10 failed logins in any 24-hour period, so the thief gets only 10 attempts at guessing the S_A. We are interested in the probability of a successful guess of S_A at the first such guessing attempt. Describe two different plausible situations under which the thief can improve his odds, for a correct first guess of S_A, to be considerably less than the theoretical 1 in 10,000. Quantify the odds of a correct first guess for what you describe.

2. Give drawbacks and advantages of modifying the scheme so that neither the device nor the server stores P_A: Instead, the server displays a 4-digit random challenge R that the user keys into the device right after entering the 4 digits of S_A, and the device and server both compute a one-time password that depends on S_A and R (rather than on S_A and P_A).

Question 4. (24 points) In this question $\hat{h}$ is a cryptographic hash function that has the following properties:

1. Given $\hat{h}(x)$ and y where x and y are integers, it is possible to quickly compute $\hat{h}(x * y)$ (where "$*$" denotes multiplication).

2. Given $\hat{h}(x)$ and $\hat{h}(y)$ where x and y are integers, it is impractical to compute $\hat{h}(x * y)$.

Assume A and B have been participating in an ongoing non-confidential online exchange (hence not using any form of encryption). Suddenly A tells B that the rest of their online interaction is confidential and should be encrypted using a session key that they must somehow agree on. They do not have a shared secret, and neither of them has a public key, therefore the information they exchange in any protocol they use for agreeing on a session key k, will be visible to adversaries like Eve or Mallory. This question is about the suitability of the following protocol for A and B to agree on a session key k and using it to communicate confidentially.

1. $A \overset{\hat{h}(r_A)}{\longrightarrow} B$

   where r_A is a large integer randomly selected by A

2. $B \overset{\hat{h}(r_B)}{\longleftarrow} B$

   where r_B is a large integer randomly selected by B

3. A uses $\hat{h}(r_B)$ and r_A to compute $h(r_B * r_A)$, and B uses $\hat{h}(r_A)$ and r_B to compute $\hat{h}(r_A * r_B)$. A and B use as many of the bits of $\hat{h}(r_A * r_B)$ as they need for their session key k.

Is the above protocol safe from Eve ? Is it safe from Mallory ? Justify your answers (i.e., briefly argue why the protocol is safe if that is your answer, otherwise provide a detailed step by step attack against it).

Note. Recall that adversaries Eve and Mallory are defined on slide 24 of the lecture slides on authentication.

Date due: Tuesday September 6, 2016 by 4:30 EDT