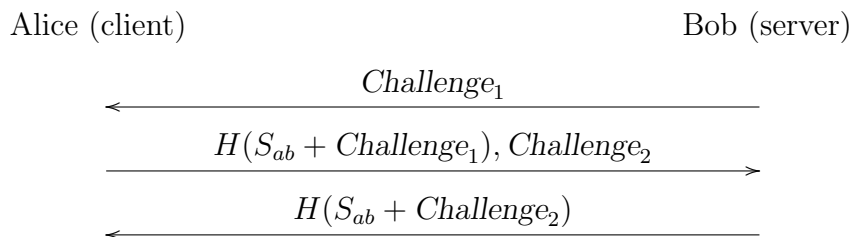


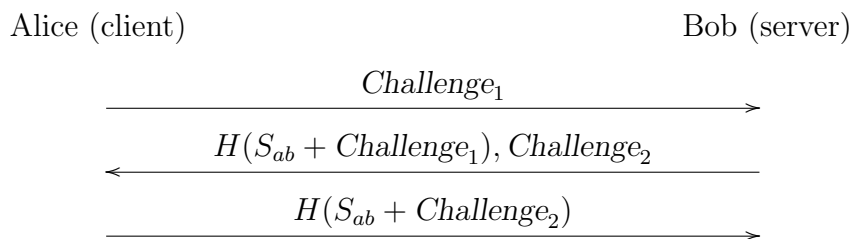
Homework #2

Due: Tuesday, March 11th, 2008, 3:30 PM.

Problem 1 In class, Eric Rescorla presented a simple mutual-authentication protocol in which a client connects to a server and each verifies that the other knows the shared secret S_{ab} :



Suppose that we changed this protocol so that it is *the client* who first sends a challenge. We would obtain the following:



This appears to be an innocuous change, but in fact the modified protocol is *insecure*: Veronica, a malicious client, can convince the server Bob that she is Alice even though she doesn't know the secret S_{ab} .

Explain the security problem in the modified protocol. What exactly makes the original protocol immune to it?

Suggest one way to fix the modified protocol without changing the direction of the message flows.

(The subtlety of such flaws is one reason that it is unwise to attempt to design new cryptographic protocols.)

Problem 2 In class, we discussed the vulnerability of TENEX's password-checking code to a timing attack:

```
static char *realpw;

int checkpw(char *userpw)
{
    int i;

    /* <= to include NUL byte in comparison */
    for (i=0; i <= strlen(realpw); i++)
        if (realpw[i] != userpw[i])
            return 0;
    return 1;
}
```

How might you fix this function so that it doesn't leak information about the length of the real password or of the matching prefix between the real password and the user password?

You may assume that the real password has some maximum length, say, 32 bytes. Be explicit about the assumptions you are making regarding the running time of various operations.

Assume that we want to maintain, at a high level, the string-to-string comparison; hashing and salting—the modern approach here—isn't in scope for this problem.

(Note: adding random delays won't work, since the effect of random noise can typically be negated by means of statistical analysis.)