Auth – Homework

Please answer the following questions. **We don't expect formal proofs: rather, just a high-level argument from intuition.** Please submit your answers as a **PDF** to Gradescope. Collaboration is allowed and encouraged, but you must write up your own answers and acknowledge your collaborators in your submission.

Due Date: *Monday, February 23rd*

1 Block Cipher Constructions

(1) What is a block cipher?

(2) Consider a two-round SPN with 64-bit block length. Assume the first and third sub-keys are equal, and the second sub-key is independent, so the master key is 128 bits long. Given input/output (x, y) pairs, show a key-recovery attack using much less than 2^{128} time.

(3) (Extra Credit) Show that DES has the property that $DES_k(x) = \overline{DES_{\bar{k}}(\bar{x})}$ for every key k and input x, where $\bar{x}$ denotes the bitwise complement of x.

2 Block Cipher Modes of Operation

(1) Consider the output feedback (OFB) mode of block cipher (see Post07.pdf, page 10). Given a ciphertext $c = (IV \| c_1 \| \dots \| c_n)$ and secret key k, how does decryption work?

(2) Is the OFB mode CPA secure (assuming F_k is a pseudorandom permutation)? Why?

(3) (Extra Credit) In "stateful" OFB mode, the output of the last F_k block will be used as the IV for the next encryption. Is the stateful OFB mode CPA secure? Why?

(4) If you were a security engineer and would like to adopt block cipher in your system for symmetric-key encryption, which mode of operation (among ECB/CBC/CTR/OFB modes) would you choose? Why?

(5) What do you need to pay attention to during the deployment of your chosen mode of operation?

3 Password-Based Authentication

In the Auth project, our password-based authentication scheme is designed to protect against adversaries that could potentially corrupt the entire storage of the server. Recall that given a hash function H (modeled as a random oracle) and a password pwd, the following is how we register and login:

Registration: First, the user sends their id and the server sends a random λ-bit salt to the user. Next, the user computes $h = H(\mathsf{pwd} \,\|\, \mathsf{salt})$ by hashing the password with the salt appended. The user then sends h to the server. Next, the server will choose a random $\mathsf{pepper} \in \{0,1\}^p$ for some small p and compute $h' = H(h \,\|\, \mathsf{pepper})$. Finally, the server stores a row $(\mathsf{id}, h', \mathsf{salt})$.

Login: First, the user sends their id and the server responds with the stored salt to the user. Next, the user computes and responds with $h = H(\mathsf{pwd} \,\|\, \mathsf{salt})$ by hashing the password with the salt appended. The server will then try for all $\mathsf{pepper}^* \in \{0,1\}^p$ computing $h^* = H(h \,\|\, \mathsf{pepper}^*)$, and authenticating the user if h^* matches the stored value h' for any pepper^*.

(1) Explain why this verification scheme is correct; that is, a valid password should be cleared for login.

(2) Explain an (inefficient) attack to recover users' passwords in case the server's entire database is compromised.

(3) What roles do the salt and pepper play against offline dictionary attacks, respectively?

(4) Given salt-and-pepper hashing, can a user safely use a simple/weak password?

(5) (Extra Credit) Explain why we may want to use a slow (computation-heavy) hash function in authentication.

(6) (Extra Credit) Explain why we may want to use a memory-hard hash function in authentication.

4 Authentication

(1) In the Auth project, the user and server first perform an authenticated key exchange. In particular, the user first sends g^a to the server, then the server sends

back (g^b, g^a, σ_s), where σ_s is a signature on (g^b, g^a). Consider a slightly modified protocol where instead the server sent back (g^b, σ'_s) where σ'_s is a signature on g^b. Explain a potential man-in-the-middle attack that could arise in the modified protocol, and why such a man-in-the-middle adversary still cannot learn anything about the encrypted messages sent from the user to the server.

(2) Consider a weaker authentication scheme *without* two-factor authentication (2FA). Suppose an adversary was able to successfully learn the shared secret g^{ab} during the login phase. Explain why, even though we never send the password over the wire, and even if the user chooses a very strong password, the adversary may be able to authenticate as this user in the future. Explain why 2FA solves the problem.

(3) During the registration phase, we ask the user to provide a valid PRF response, even though we just sent the user the PRF key (or PRG seed). Why is this step necessary?

Hint: we allow each id *to register only once, and our network may not be stable.*

5 Delegated Trust

The way that our authentication scheme works is that since we trust the server, the server can **delegate trust** to others that it trusts, allowing us to verify the identity of a third party without consulting directly with the server. This is essentially the same idea behind larger systems such as Public Key Infrastructure (PKI).

(1) Propose a protocol that allows users to delegate trust to other users. What does delegation look like? What does verification look like?

(2) Suppose a secret signing key of some user u has been compromised, and suppose we have some way of invalidating certifications (e.g., a public revocation board). Which users should have their certificates invalidated and reissued in the case of such a compromise?

(3) (Extra Credit) Assume revocation of certificates is handled in the following way: when a user Alice claims that the private key corresponding to her public verification key vk_A has been stolen, Alice sends to the server a statement of this fact signed with respect to vk_A. Upon receiving such a signed message, the server revokes the appropriate certificate. Explain why it is *not* necessary for the server to check Alice's identity in this case. In particular, explain why it is of *no* concern that an adversary who has stolen Alice's private key can forge signatures with respect to vk_A.