Lab 5 Theory

This assignment is due **at 10:00pm ET** on **Thursday, December 2, 2021**.

Please make note of the following instructions:

- Remember that your solutions must be submitted on Gradescope. Please sign-up for 6.S060 Fall 2021 on Gradescope, with the entry code `KYRBPK`, using your MIT email.
- We require that the solution to the problems is submitted as a PDF file, **typeset on LaTeX**, using the template available on the course website (`https://6s060.csail.mit.edu/2021/`). Each submitted solution should start with your name, the course number, the problem number, the date, and the names of any students with whom you collaborated.
- We will have bug bounties for Lab 3 and beyond. If you are the first to report a bug in either the Coding or the Theory parts of the lab you will receive a $10 Toscanini's Gift Certificate! Serialization is via Piazza – report the bug as a private question on Piazza and we will respond as soon as possible as to whether it is a real bug or not.

Problem 5-1. More Efficient Measured Boot [20 points]

Key generation of public and private keys is a costly operation. In Lecture 17, computing $\{SK_S, PK_S\}$ each time the system boots therefore has significant cost. Recall that $cert_S$ ties $Hash_S$ to PK_S, and SK_S is necessary to prove the certificate corresponds to a specific, current software environment. Can we generate $\{SK_S, PK_S\}$ *once* as in Lecture 17, and then avoid continually re-generating SK_S on each boot by exploiting symmetric key and hash operations and untrusted storage? Explain how.

Problem 5-2. Secure deletion [80 points]

Your laptop has two types of non-volatile storage:

1. *A small amount of internal storage (e.g., in a secure enclave) that supports secure deletion.* Think of this storage as an array of bytes. An attacker who gets hold of your laptop learns only the values *currently stored* in the secure storage—the attacker learns nothing about values that were written into the array in the past.
2. *A large amount of external storage (e.g., SSD disk) on which deletion is impossible.* Think of this storage as an array of bytes. An attacker who gets hold of your laptop can learn the state of this array *at every point in the past*. So, while you can write new values to this external storage device, you can never delete old values that were written to it. (It turns out that it is difficult to safely erase old values from a hard disk, so this pessimistic model is actually quite realistic.)

In all parts of this problem, you may assume that you have an unlimited amount of volatile storage (RAM). This volatile storage is erased completely when the device is powered down. So, you do not need to worry about having enough working space to store intermediate values—you just need to worry about what is stored in non-volatile memory when the machine is powered off.

Throughout this problem, assume that you have only $O(\lambda)$ bits of internal storage, on security parameter λ, and an unbounded amount of external storage.

(a) [20 points] You want to store an n-byte array in external storage. Show how to use the two types of non-volatile storage given to implement the following three routines:

- `Write(bytes)` – Write the n bytes given to non-volatile storage. This overwrites whatever bytes were stored before.
- `Read() -> bytes` – Read the n bytes from non-volatile storage.
- `Delete()` – Securely delete the n bytes. That is, after `Delete` finishes running, an attacker who compromises the state of both the internal and external storage should learn no information, in a computational sense, about the bytes written.

(b) [45 points] The scheme from Part (a) has the restriction that it is not possible to securely delete any part of the stored byte array without reading in the entire array in time $\Omega(n)$. So, each memory access costs n time instead of $O(1)$ time. In this part, you will remove that restriction.

As before, we want to outsource an array of n bytes. As before, you only have $O(\lambda)$ bytes of non-volatile internal storage. Now implement the following routines, **each of which must run in $O(\lambda \cdot \sqrt{n})$ time**:

- `Setup()` – Initialize the storage.

- `Write(i, byte)` – Set the value of byte i in non-volatile storage.
- `Read(i) -> byte` – Read the value of byte i from non-volatile storage.
- `Delete(i)` – Securely erase the value of byte i. That is, after `Delete` finishes running, an attacker who compromises the state of both the internal and external storage should learn no information, in a computational sense, about the value at byte i.

(c) [15 points] Explain how to extend your solution from Part (b) to allow all operations to run in time $\lambda \cdot O(\log n)$. As before, you only have $O(\lambda)$ bytes of non-volatile internal storage.

You **do not** need to write out the algorithms explicitly—just explain the high-level idea in words.

Hint: You will probably want to use some sort of tree.