Homework 3.4

Due Date: See Canvas

Recall: Merkle-Damgård Transform

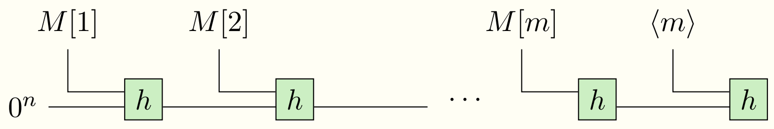

Figure 1: A visualization of the general Merkle-Damgård transform, where h is a function compression an $(n+b)$-bit input into an n-bit output, $M[i]$ is the i^{th} b-bit block of the message M, and $\langle m\rangle$ is the bit encoding of the message's block count, m.

I. Introduction

The goal of this assignment is to break security of a deterministic hash-based MAC. Namely, let H be the hash function constructed from the compression hash function h via the Merkle-Damgård transform as per Figure 1. Then, to tag a message M consisting of b-bit blocks under a b-bit key K we compute $tag \leftarrow H(K\|M)$.

We have provided you with several Python files that you will work with in this assignment. In particular, **you will be demonstrating that the MAC described above is not UF-CMA secure by modifying student.py to create a forged message and a corresponding tag**. Detailed explanations can be found in the Files section, below.

You will be submitting your deliverables via Gradescope.

1.1 Files

You've been provided the following library:

- student.py contains your exploit code that will successfully create a forged message that passes the MAC without knowing the secret key.

  This is the only file you will modify for submission.

- crypto.py contains a custom SHA1 implementation modeled after Python's `hashlib` module, but also exposes some extra parameters that might help with a length extension attack.

  This module also includes convenience functions for converting between strings and bytes. **We recommend that you do NOT use Python's `str.encode` and `bytes.decode`**, as these often result in confusing errors with character encodings and Unicode. Prefer `crypto.s2b` and `crypto.b2s` (for converting from strings to bytes and vice-versa, respectively), instead.

- oracle.py simulates an UF-CMA adversarial scenario by including the necessary oracles.

- grader.py runs your exploit and lets you know whether or not you've successfully forged a message.

- secret.txt holds the secret key used by the oracle to create tags internally. Though you can view and edit it if you wish, it will be of no use to you. The auto-grader will use different keys, and **your solution should work for any key**.

The docstrings in each file provide further details about these modules.

Objective: Make `student.main()` return a (message, tag) pair that (at least) includes the **original message** and your **GT username**, hasn't been submitted to the oracle, and passes the MAC.

1.2 Running the Code

You can install the latest version of Python 3 for your system from here; there are no extra dependencies to install. Older versions of Python 3 may work, but we cannot make guarantees. To run the local auto-grader, simply execute:

```
python grader.py [your GT username]
```

II. Submission

You will need to submit the following deliverables via Gradescope. **There are different assignments for each**, so please be careful to submit to the right one!

- student.py *(15 points)*: Complete `main()` to forge messages via a length extension attack. Remember that the messages you forge should contain at least your GT username and the original message.

  You **must** keep the existing structure: nothing should run if you execute `python student.py` on its own, the input parameters should stay the same, and the return value(s) should match the expected format.

  Submit this to **Homework 3.4 (Coding section)** on Gradescope.

  The autograder will run a suite of tests to determine your score, offering small suggestions for common mistakes if it encounters them or exception logs if your code doesn't run.

- report.pdf *(5 points)*: **Briefly** discuss the exploit details. You should touch on, at minimum: how reliable your attack is, its (rough) run-time complexity, the root causes of the vulnerability in the integrity scheme at a high level, and how it could be alleviated (be sure to explain *why* these fixes will remedy the problem.).

  Submit this to **Homework 3.4 (Report)** on Gradescope.

  All of these can be answered in less than a page; take care to be succinct *and* precise. Submissions over two pages long may be penalized or their excess content ignored.