**Applications of Cryptography CSCE 4050/5050 (Spring 2025)**

**Homework 6**

1. [**Hashing files: implementation**] Download the slides of Lecture 6-1 as “L6-1-CSCE4050-5050.pdf”. Compute its SHA256 hash value and compare it with the value provided on the Canvas page of Lecture 6; attach the screenshots of both hash values. Note that any popular OS has a built-in utility for com**A screenshot of a computer

   Description automatically generated**puting SHA256 hash values of files: find out what it is and use it. Next, let us confirm that changing the contents of a file results in changing the hash value. Open “L6-1-CSCE4050-5050.pdf” in any text editor (e.g., Notepad++), replace “1.7” in the first line with “1.0” and save the file. You may confirm that the modified PDF opens correctly and displays the same contents as before. Now, compute the SHA 256 value of the modified file and print it to the screen. Attach the respective screenshot.

**Answer:**

**SHA256 hash value given provided on canvas:**

A black text on a white background

Description automatically generated

**Computed SHA256 hash value of file “L6-1-CSCE4050-5050.pdf” :”:**

**A screenshot of a computer

Description automatically generated**

**SHA256 hash value of modified file:**

**A screenshot of a computer

Description automatically generated**

1. [**Hash function: implementation**] Write a program that implements a (weak) cryptographic hash function (we will call it WEAK24) as follows: WEAK24:= SHA256[1..24]. In other words, WEAK24 is based on SHA256, and it outputs the first 24 bits of the output of SHA256.   
   For the input “Hello, World!”, print its WEAK24 output to the screen and attach the respective screenshot.

**Answer:**

**The output of the program:**

**A black background with white text

Description automatically generated**

1. [**Collision search: implementation**] Write a program that finds a collision for WEAK24. In other words, you need to find two distinct inputs with the same WEAK24 hash. Use the birthday attack for your collision search. See Lecture 6-1 for details.

**Note:** Full credit will only be given to the implementations that use birthday attacks.

**Answer:**

**The output of the program:**

**A black screen with white text

Description automatically generated**

1. [**Secure** **combiner for collision-resistant hash functions**] Suppose that H1 and H2 are collision-resistant hash functions (CRHF). Now, we want to construct a new CRHF H using H1 and H2 as building blocks, so that if at some point in the future one of them is broken (but not both) then the overall construction H is still secure. Suppose H1 and H2 are defined over the same message space M (but their digest spaces T1 and T2 may be different). Let us define a new function as follows: H(m) := ( H1(m) || H2(m) ), where “||” denotes a concatenation of binary strings.

Task: Show that H is a secure CRHF, if either H1 or H2 is secure.

Hint: A simple proof by contradiction is possible. For that, first assume that one can find a collision on H and then show that it is impossible given the assumptions on H1 and ­H­2.

**Answer:**

Let us assume by contradiction, H is not Collison-resistant which means there is a polynomial time adversary A that has a non-negligible probability of finding a collision for H.

In this case, if m and m’ are two distinct messages, A outputs two distinct messages such that m != m’.

H(m) = H(m’) ⇒ H1 (m) || H2 (m) = H1(m’) || H2(m’)

Which tells us that: H1(m)=H1(m’) AND H2(m)=H2(m’)

Thus, both H1​ and H2 ​ produce the same hash values for two different messages m and m′ , meaning that both have collisions.

We can use adversary A to create two new adversaries, A1 and A2, as follows:

Since the adversary A discovered a collision (m, m′) for H

* As a collision for H1, Adversary 𝐴1 runs A and produces (m, m′).
* As a collision for H2, adversary A2 runs A and outputs (m, m′).

H is trivially insecure if both H1 and H2 are broken. We did, however, assume that at least one of them is Collison-resistant and secure.

However, according to the CHRF Security, if H1 is secure, then no effective adversary such as A1 should be able find a collision in H1.  
Likewise, if H2 is secure, then A2 is likewise impossible.

Which contradicts our assumption that H is not collision resistant.

Since no adversary A could exist, its certain that:

The function H(m)=H1(m) || H2 (m) is secure CRHF if at least one of H1 or H2 is collision-resistant (Secure).

1. [**Digital signature: ECDSA: implementation**] Write a program that computes and verifies the digital signature according to the ECDSA algorithm. Specifically, the program should perform the following steps: (1) Generate a key pair and store the public key in the file “pk.key” and the secret key in “sk.key”. (2) Create a file “message.txt” containing the text “Hello, World!”, compute the digital signature of this file (using the secret key) and store the signature in the file “sig.bin”. (3) Verify the signature on “message.txt” (using the public key) and confirm that the signature is valid. (4) In “message.txt”, replace the first “H” with “J”, save the file (but keep “sig.bin” unchanged), and then try to verify the signature again. Confirm that the verification now fails.

**Answer:**

**Output of the program:**

**A computer screen with white text

Description automatically generated**

**File: pk.key**

**A black screen with white text

Description automatically generated**

**File: sk.key**

**A black background with white text

Description automatically generated**

1. [**VirtualBox and SEED VM**] Later in this course, some assignments will include labs. They are based on the SEED labs (https://www.handsonsecurity.net/resources.html) which use VirtualBox (https://www.virtualbox.org/). In this question, we will perform some basic setup of the environment for these labs. Download the latest version of VirtualBox on your computer. Then, import a pre-built SEED Ubuntu 20.04 VM using the VM Manual in Section “Ubuntu 20.04 VM (for Intel/AMD Machines)” from here: <https://seedsecuritylabs.org/labsetup.html>

Note that the newest version of VirtualBox does support Apple Silicon machines (M1/M2 chips), and hence their users may also follow the above-mentioned VM Manual. You must attach the following two screenshots: (1) a VirtualBox window showing an imported SEED Ubuntu VM.   
(2) Start the SEED Ubuntu VM and provide a screenshot of the Ubuntu (guest OS) desktop.

**Answer:**

**Screenshot for 1: An Imported SEED Ubuntu VM (Virtual Box)**

**A screenshot of a computer

Description automatically generated**

**Screenshot for 2:**

**A screenshot of a computer

Description automatically generated**