EN.601.441/641: Assignment 1a

January 2026

This is an individual assignment. Submit to Gradescope by February 7 at 11:59pm.

Hash functions (5 points) Bob is trying to find collisions in some toy hash function that has a tiny digest size of just 32 bits:

$$H : \{0,1\}^* \to \{0,1\}^{32}$$

His goal is to find a collision, i.e., a pair of messages M_1, M_2 such that $M_1 \neq M_2$ and $H(M_1) = H(M_2)$. His strategy is to pick distinct input messages $M_1, M_2, \ldots, M_N$ and compute $H(M_1), H(M_2), \ldots, H(M_N)$ until he finds two distinct messages that hash to the same value. Three questions:

1. After Bob has chosen N distinct messages, how many different *pairs* of messages can he assemble from the N messages? (Expressed as a function of N.)
2. What is the concrete value when calculated for $N = 2^{16}$?
3. Let's imagine the hash function output is very "random", i.e., for any pair M_i, M_j the probability that $H(M_i) = H(M_j)$ is exactly $1/2^{32}$. Then after testing $N = 2^{16}$ messages, what is the probability that some collision pair has been found?

Hash functions II (5 points) Suppose Mallory is launching a new 'secure' messaging app. When Alice installs the app, it creates an account for her on the server using a hash of her phone number as the account ID. The app then queries the server by sending a hash of each phone number in Alice's contacts to learn which of Alice's friends are already on the platform. The goal is that users can discover their friends without the server learning the contents of every user's address book.

Assuming phone numbers are 10 digits, explain why this does not achieve the intended security goal. How can Mallory act maliciously to determine the phone numbers of every one of Alice's contacts?

Building Signatures (15 points) Ralph has an idea for a new signature scheme that can be built using *only* a secure cryptographic hash function. The downside of his signature scheme is that each signing key can only be used once, to sign a single message.

His idea is as follows. Let $H : \{0,1\}^* \to \{0,1\}^k$ be some hash function, where k is the digest size. To generate a signing and verification key, Ralph does the following:

1. First, he generates $2k$ distinct random strings, each string of length k bits. He organizes these $2k$ strings into two vectors (note, the superscripts are just labels, not exponents):

$$r_1^0, r_2^0, \ldots, r_k^0$$

$$r_1^1, r_2^1, \ldots, r_k^1$$

This collection of strings forms the secret signing key sk.

2. Next, he computes the hash of each string. This produces two vectors:

$$H(r_1^0), H(r_2^0), \ldots, H(r_k^0)$$

$$H(r_1^1), H(r_2^1), \ldots, H(r_k^1)$$

This set of $2k$ hashes becomes the public verification key vk.

To sign a message M, Ralph first computes $h = H(M)$. He then breaks h into a sequence of bits of the form $h = b_1 \| b_2 \| \ldots \| b_k$. Now for $i = 1$ to k, Ralph does as follows:

1. If $b_i = 0$, Ralph outputs r_i^0.
2. Otherwise if $b_i = 1$ Ralph outputs r_i^1.

Ralph outputs his signature as $\sigma = (r_1^{b_1}, r_2^{b_2}, \ldots, r_k^{b_k})$.

Please answer the following questions:

1. Given a message M, a verification key vk and a signature σ made by Ralph, explain how Bob would *verify* the signature is valid.
2. If Ralph signs a message M and gives out the public verification key vk and the signature σ, can an attacker Mallory "forge" a new signature σ' on a different message M' (without knowing sk)? Why or why not? Hint: explain why doing this would break some property of the hash function.
3. What happens if Ralph uses the same secret key sk to sign two different messages? Can Mallory now forge a signature?
4. Ralph has a clever idea: rather than outputting a string for each bit of h, he decides that the signature will contain a string $(r_i^{b_i})$ only if $b_i = 1$. If $b_i = 0$, he will not add any string to the signature. Describe the new verification algorithm, and determine whether this signature scheme secure against Mallory forging new messages.

PoW difficulty (10 Points) Let's define a hash function $H : P \times S \to \{0,1\}^n$ that takes in two inputs $p \in P, s \in S$ and outputs an n-bit digest. We'll call p the *puzzle*, and s the *solution*. For simplicity in the problem below, we'll treat the output as an unsigned integer in the range $0, \ldots, 2^n - 1$.

We say that H is *proof of work secure* with difficulty d (say, $d = 250$) if for a randomly chosen puzzle $p \in P$, it is difficult to find a solution $s \in S$ such that $H(p, s) < \lfloor \frac{2^n}{d} \rfloor$ in time significantly less than $\frac{2^n}{d}$. You can verify on your own that if we view a hash function as a random function (i.e., random oracle), then it indeed satisfies the proof of work security for any suitable choice of parameters. In this question we explore the relation between collision resistance and proof of work security.

Show that a collision resistant hash function may not be proof of work secure. Specifically, let $H : P \times S \to \{0,1\}^n$ be a collision resistant hash function. Construct a new hash function $H' : P \times S \to \{0,1\}^{n'}$ (where n' may be greater than n) that is also collision resistant, but for a fixed difficulty d is not proof of work secure with difficulty d. This is despite H' being collision resistant. Also, explain why H' is collision resistant, that is, why a collision on H' would yield a collision on H.

Bitcoin (5 points) Explain qualitatively why the verification of a transaction in the most recently broadcast block is less reliable than one in a block a few prior to the most recent block.