Problem 1-1. Message authentication codes Recall that in class we mentioned the hash-then-MAC paradigm, which uses a MAC for messages of fixed length n (such as AES, in which case $n = 128$), and a hash function with range $\{0,1\}^n$, to MAC messages of arbitrary length, by first hashing the message to an element in $\{0,1\}^n$ and then applying the underlying MAC to the hash value.

In class we mentioned that for the resulting MAC to be secure the hash function must be collision resistant, and thus this paradigm cannot be used with AES since its domain consists of messages of length 128, and there does not exist a collision resistant hash function with 128-bit security that maps to tags of bitlength 128, due to the Birthday Paradox.

In this problem we consider using the hash-then-MAC paradigm with AES but with a hash function with a *secret* seed. In particular, consider the following hash function H: It takes as input a (secret) seed $(a_1, \ldots, a_t)$ and a message $(M_1, \ldots, M_t)$, where each a_i and M_i is a block of 128 bits. The hash function outputs

$$H((a_1, \ldots, a_t), (M_1, \ldots, M_t)) = \sum_{i=1}^{t} a_i M_i.$$

The arithmetic is done over a finite field of size 2^{128} (known as the Galois Field and denoted by $\mathsf{GF}[2^{128}]$), though how the arithmetic is done is not important, the only important thing is that the output is in $\{0,1\}^{128}$, and that in every (finite) field (and in particular in $\mathsf{GF}[2^{128}]$) every non-zero element has an multiplicative inverse.

(a) Define the MAC scheme obtained by applying the hash-then-MAC paradigm to the hash function $H : (\{0,1\}^{128})^t \times (\{0,1\}^{128})^t \to \{0,1\}^{128}$ and AES as the underlying MAC.

(b)
1. Recall the attack for the hash-then-MAC using AES and a hash function without a secret seed (assuming the message space is $(\{0,1\}^{128})^t$ for $t \geq 2$). *Hint:* This attack takes time roughly 2^{64}.
2. Explain why this attack fails if we use the hash function above (with a secret seed), assuming the attacker sees many fewer than 2^{64} tags.
3. Is this hash-then-MAC scheme (with H as defined above and AES) secure if the attacker gets to see more than 2^{64} tags for messages of his choice? *Hint:* Use the fact that AES is injective (i.e., if $\mathsf{AES}(K, M) = \mathsf{AES}(K, M')$ then $M = M'$), and the fact that H is a linear function, together with the fact that one can efficiently solve t linear equations with t variables (using Gaussian elimination).

Problem 1-2. Weaknesses of CPA-secure cryptosystems

Let $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a secure pseudorandom function. In class, we saw the CPA-secure encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ with keyspace $\mathcal{K}$, where

$$\mathsf{Enc}(k,m) := \begin{cases} r \xleftarrow{R} \{0,1\}^n \\ c \leftarrow F(k,r) \oplus m \\ \text{output } (r,c) \end{cases} \qquad \mathsf{Dec}(k,(r,c)) := \begin{cases} m \leftarrow F(k,r) \oplus c \\ \text{output } m \end{cases} .$$

We will use a secure MAC scheme $\mathsf{MAC}\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$ over the same keyspace $\mathcal{K}$.

MIT uses encryption to protect communications between department offices and its central payroll system. In particular, the EECS department office and MIT's payroll office share a secret encryption key $k_{\mathsf{Enc}} \xleftarrow{R} \mathcal{K}$ and a secret MAC key $k_{\mathsf{MAC}} \xleftarrow{R} \mathcal{K}$.

On each pay date, a machine in the EECS department sends a sequence of messages to the payroll office. Each plaintext message is a 54-byte ASCII-encoded string with format:

$$m = \text{“}\texttt{date:NNNN-NN-NN, mit\_id:NNNNNNNNN, amount:NNNNNNNNN.NN}\text{”},$$

where each `N` represents an ASCII-encoded decimal digit (i.e., ASCII values `0x30`–`0x39`).

Unfortunately, the designers of the system make the wrong decision and use MAC-then-encrypt rather than encrypt-then-MAC. In particular, the EECS department encrypts each message m as:

$$\mathsf{ct} = \mathsf{Enc}(k_{\mathsf{Enc}}, m \,\|\, \mathsf{MAC}(k_{\mathsf{MAC}}, m)).$$

The machine at the payroll office operates on each ciphertext ct as follows:

- Compute $(m' \| t') \leftarrow \mathsf{Dec}(k_{\mathsf{Enc}}, \mathsf{ct})$.
- Check that the bytes of the string corresponding to the MIT ID number and the payment amount are valid ASCII decimal digits. If not, send a `FORMAT ERROR` message to the EECS machine.
- Check that the date corresponds to today's date. If not, send a `DATE ERROR` message to the EECS machine.
- Check that $t' = \mathsf{MAC}(k_{\mathsf{MAC}}, m')$. If not, send a `MAC ERROR` message to the EECS machine.
- Append the $(\texttt{ID}, \texttt{amount})$ pair to a log file for later processing and send an `OK` message to the EECS machine.

Explain how a network attacker that can intercept ciphertexts and interact with the payroll server can recover **all but one of the bits in each secret digit** in the plaintexts that the EECS department sends. Recovering these bits for a single plaintext using your attack should take no more than than 10,000 interactions with the payroll server.

Problem 1-3. Encryption and RSA

(a) Which of the following security goal(s) does encryption address :
(1) Confidentiality (2) Integrity (3) Sender authentication (4) Non-repudiation.

(b) Suppose you obtain two ciphertexts C,C' encrypted using one-time pad, with key K and its bitwise complement $\bar{K} = K \oplus 1$ respectively. What can you infer about the corresponding plaintext messages?

(c) Alice and Bob are using public keys $(e_1, N_1), (e_2, N_2)$ respectively. Suppose you are informed that their RSA moduli $N1, N2$ are not relatively prime. How would you break the security of their subsequent communication? It is sufficient to show that you can get $\phi(N_1)$ and $\phi(N_2)$.
Hint: Euclid's algorithm allows us to compute GCD(x,y) efficiently.

(d) Suppose that a system uses textbook RSA encryption. An attacker wants to decrypt a ciphertext c to obtain the corresponding confidential plaintext m. Assume that the victim system readily decrypts arbitrary ciphertexts that the attacker can choose, except for ciphertext c itself. Show that the attacker can obtain m from c even under this setting, i.e RSA is not CCA secure.