2. Broken Hashes.

Prove that the following MAC constructions are insecure, even if $F$ must not be explicitly mentioned and you may not use any source besides the lecture notes or textbook. 

Each group should turn in one copy with the names of all group members on it. You may make any choice to complete these homeworks in a group of up to three students.

### Ground Rules.

You may choose to complete these homeworks in a group of up to three students. Each group should turn in one copy with the names of all group members on it. You may use any source you can find to help with this assignment but you must explicitly reference any source you use besides the lecture notes or textbook. Electronically typeset copies of your solution should be submitted on the course moodle by 11:59 PM on the date above.

### 1. Broken MACs. [20 points]

Let $F : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}^k$ be a pseudorandom function. Prove that the following MAC constructions are insecure, even if $F$ is secure.

(a) Define $aMAC$ on messages $m = m_1||m_2 \in \{0, 1\}^{2k}$ by

$$aMAC_K(m_1||m_2) = F_K(m_1)||F_K(F_K(m_2)).$$

(b) Define $bMac$ on messages $m = m_1||\cdots||m_\ell \in \{0, 1\}^{\frac{k}{2} \times \ell}$ by

$$bMac_K(m_1||\cdots||m_\ell) = r||(F_K(r) \oplus F_K(1)||m_1) \oplus \cdots \oplus F_K(\ell)||m_\ell),$$

where $r \in_R \{0, 1\}^k$ and $\langle i \rangle$ is the $\ell/2$-bit representation of integer $i$.

### 2. Broken Hashes. [20 points]

(a) [10 points] Suppose we design a hash function (family) $H_K : \{0, 1\}^\ast \rightarrow \{0, 1\}^\ell$ from the block cipher $E : \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ by fixing a key $K \in \{0, 1\}^k$ and setting $H(m) = CBC-MAC_E^K(m)$, that is, we divide $m$ into $\ell$-bit blocks $m_1, m_2, \ldots, m_\ell$, set $c_0 = 0^\ell$, $c_i = E_K(m_i \oplus c_{i-1})$, and let $H(m) = c_\ell$. Show that this hash function (for known $K$) is not collision resistant.

(b) [5 points] Let $H_1, H_2 : \{0, 1\}^{\ell+t} \rightarrow \{0, 1\}^\ell$, where $t < \ell$, and define $H'(x) = H_1(x)||H_2(x)$. Show that if $H_1$ or $H_2$ is collision-resistant, then so is $H'$.

(c) [5 points] However, show that even if $H_1$ is a truly random function, there exists a function $H_2$ such that $H'$ is not preimage-resistant: there is an algorithm that finds preimages to $H'$ in expected time $2^\ell$.

### 3. MACs and Hashes together. [15 points]

(a) [5 points] Let $H : \{0, 1\}^\ast \rightarrow \{0, 1\}^\ell$ be a collision resistant hash function (family). Let $LB_t(x)$ return the last $t$ bits of $x$. Prove that the hash function (family) $H'(x) = H(x)||LB_t(x)$ is collision resistant.

(b) [10 points] Let $H : \{0, 1\}^\ast \rightarrow \{0, 1\}^\ell$ be a collision-resistant hash function (family) and $M_K : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ be a (EUF-CMA) secure fixed-length MAC. Let $F_K : \{0, 1\}^\ast \rightarrow \{0, 1\}^\ell$ compute a tag $F_K(m)$ by dividing $m$ into $n$-bit blocks $m_1, \ldots, m_\lambda$ and computing $H(M_K(m_1), \ldots, M_K(m_\lambda))$. Prove that this composition is not generically secure, i.e. that there exist secure $H$ and $M$ such that $F_K$ is not EUF-CMA secure.
(c) [Extra credit: 10 points] Prove, however, that if the hash function $H$ in part (c) is modelled as a random oracle, then the resulting MAC $F_K$ is EUF-CMA secure.

4. MACs and Encryption. [20 points]

(a) [10 points] Let $\text{Enc}$ be an IND-CPA secure encryption scheme and $M$ be a EUF-CMA secure MAC. Define the composed encryption function $\text{Enc}^M_{K_1, K_2}(x) = \text{Enc}_{K_1}(x) || M_{K_2}(x)$ ("encrypt AND mac"). Prove that there exists an IND-CPA secure encryption scheme $\text{Enc}$ and EUF-CMA secure mac $M$ such that $\text{Enc}^M$ is not even IND-CPA secure.

(b) [10 points] Since a block cipher in CBC mode can be used to build both an IND-CPA secure encryption scheme and a EUF-CMA secure block, a common mistake made in “roll-your-own” cryptosystems is to try to use the last ciphertext block to compute a MAC on the plaintext, e.g., to encrypt the message $m$, we compute the ciphertext $c = \text{CBC-Encrypt}^E_K(m)$, let $c_*$ be the last block, compute tag $\tau = E_K(c_*)$. and use $\langle c, \tau \rangle$ as an “authenticated encryption” scheme. Prove that this scheme fails to provide chosen-ciphertext security.

(c) [Extra Credit: 10 points] In several encryption standards, ciphertexts may optionally be protected by a MAC. The entire ciphertext is accompanied by metadata specifying information such as which keys and encryption algorithms to use; if a MAC is used the tag is computed over this “associated data” as well. Suppose that a ciphertext is encrypted using an implementation that is vulnerable to chosen ciphertext attack (such as the CBC padding attack), and a MAC is used to protect against this attack. (i) Show how the ciphertext can still be attacked. (ii) Assuming that the unauthenticated encryption option must still be supported, how would you design the authenticated encryption scheme to avoid this kind of attack? Prove that your design is secure.

5. Hash cycles. [25 points] For a given hash function $h$, a hash chain starting from $x$ is recursively defined as follows:

$$H_0 = x$$
$$H_i = h(H_{i-1}) \text{ for } i \geq 1.$$

For the purpose of this homework, each student’s hash function is the last $k$ bits of SHA256 seeded with his or her CSE Labs username. In other words, $h|_k(x) = \text{SHA256}($userid||x$) \mod 2^k$. (If you want to test your implementation of $h|_k$, you can use the program hashtest.py. hashtest.py k x takes a number of bits $k$ and a string $x$ to hash on the command line and writes $h|_k(x)$ to the standard output; if $k$ is not a multiple of 8, the leading byte is zero-padded.

After reading Section 2.1.6 of HAC (available from http://www.cacr.math.uwaterloo.ca/hac/about/chap2.pdf), answer the following questions.

(a) [10 points] Write a computer program to compute the number of components, average/max tail length, and min/average/max cycle length in your $h|_{16}$. Your output should print out these 6 numbers. Tail and cycle are defined in 2.35. To avoid the confusion, tail length is defined as the number edges of the path to a cycle from a point. In the following Figure, the number of components is 2, tail length of node 13 is 3, tail length of node 12 is 1, and the
5. Hash chain (80 points) This problem requires programming. If possible, please restrict the programming language to one of C, C++, java, and Python. You need to let us know how to compile and run your program.

5. Hash chain is recursively defined as follows: $h(x)$ for the largest $k$ you can. To show that you found a cycle, present the initial value, and the number of times it needs to be hashed before it repeats. Under the assumption that your program runs correctly, here are the grading criteria.

(b) [5 points] Prove that the functional graph of any hash function with a fixed output length must have cycles.

(c) [10 points] How would you design a hash function to have maximal cycle length while still resisting preimages and free collisions? You may assume you have access to an ideal block cipher. Prove that your construction has these properties.

(d) [Extra credit: 15 points] Find a cycle of $h|_k$ for the largest $k$ you can. To show that you found a cycle, present the initial value, and the number of times it needs to be hashed before it repeats. Under the assumption that your program runs correctly, here are the grading criteria. If $k > 80$, you will get 15 points. If $72 < k \leq 80$, you will get 12 points. If $64 < k \leq 72$, you will get 9 points. If $56 < k \leq 64$, you will get 6 points. If $32 < k \leq 56$, you will get 3 points. For the purpose of verification, use your student ID as an initial hash value and provide the cycle length you found. (Your main submission should include the cycle length, a brief description of your algorithm, and how to compile/run your source code on a CSELabs Linux machine. Your moodle submission should include a separate source file for this question)

(e) [Extra credit: 10 points] Find a (free) collision in $h|_k$: two distinct messages $m_1$, $m_2$ such that $h|_k(m_1) = h|_k(m_2)$. For verification purposes, use your student ID as the initial hash value and provide the following values: the colliding hash ($h_4$ in the figure), and its preimages in the tail and the cycle ($h_3$ and $h_{100}$ in figure (b), respectively). For $k > 80$ this is worth 10 points, $72 < k \leq 80$ is worth 8 points, $64 < k \leq 72$ is worth 6 points, $56 < k \leq 64$ is worth 4 points, and $32 < k \leq 56$ is worth 2 points. Your main submission should include the collision values, a brief description of your algorithm, and how to compile/run your source code on a
CSELabs Linux machine. Your moodle submission should include a separate source file for this question.

(Note: for (d) and (e), be careful not to use too much memory: a straightforward algorithm could use several terabytes for large enough $k$. You need to use a more clever, space-efficient algorithm. Either design one yourself, or do some research for such an algorithm.)