CS/ECE407 Cryptography – Homework 3
Block Ciphers, CPA/CCA Security, and MACs

Due: Tuesday, March 10, 3:30pm CT

Remember, you are free to collaborate with up to one classmate. See the course webpage for more details. You are expected to write out and submit your own solutions! Your collaboration is for discussing problems at a high level, not plagiarizing answers.

The non-coding portion of your homework should be typed or carefully handwritten. We provide a LaTeX template for this document, if you would like to use it as a starting point. **If we cannot read your handwritten answers, they will not receive credit.**

The coding portion of the homework should be submitted as a single C++ file named `hw3.cpp`. Simply fill in the provided function definitions. You should not need to `#include` additional files. Solve the coding problems by attacking the provided cryptographic interface. Solutions that exploit side-channels—such as by timing how long a function call takes to run—will be counted as wrong. It is up to our discretion to decide what counts as such a side-channel attack.

Your typed solutions and C++ code should be submitted through Gradescope (see course webpage). Your hand-written solutions should be scanned and turned in through Gradescope.

Problem 1 (CPA Insecurity)**.** Let F, F^{-1} be a secure block cipher on λ-bit blocks. Consider the following (incomplete) encryption scheme whose keys are uniform λ-bit strings and that handles 2λ-bit messages (parsed as two λ-bit halves (m_1, m_2)). The scheme has 3λ-bit ciphertexts:

$Enc(k, (m_1, m_2))$:
  $r \leftarrow \{0,1\}^\lambda$
  $c_0 = F(k, r)$
  $c_1 = F(k, r \oplus m_1 \oplus F(k, m_1) \oplus m_2)$
  return (r, c_0, c_1)

- **(3 points)** This scheme is not CPA secure (regardless of how we define *Dec*). Prove it.

Problem 2 (CCA Insecurity)**.** Let F, F^{-1} be a secure block cipher on n-bit inputs. Consider the following incomplete encryption scheme that handles n-bit messages (we give an explicit *KeyGen* procedure that describes how keys are sampled):

$KeyGen(\ )$:
  $k_0 \leftarrow \{0,1\}^\lambda$
  $k_1 \leftarrow \{0,1\}^\lambda$
  return (k_0, k_1)

$Enc((k_0, k_1), m)$:
  $r \leftarrow \{0,1\}^\lambda$
  $c_0 = F(k_0, r)$
  $c_1 = F(k_0, r \oplus m \oplus k_1)$
  return (c_0, c_1)

1. **(1 point)** The decryption algorithm *Dec* is missing. Define it!
2. **(2 points)** Prove that your *Dec* is correct.
3. **(3 points)** The completed scheme is not CCA secure. Prove it is not CCA secure by constructing a distinguisher and arguing that your distinguisher indeed wins the CCA distinguishing game.

Problem 3 (MACs)**.** Recall the definition of a secure MAC scheme (Definition 1).

1. **(3 points, C++ programming problem)** Let $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a PRF. Consider the following MAC scheme which tags messages of length a multiple of λ:

$tag(k, x_0, ..., x_{n-1} \in \{0,1\}^\lambda)$:
  $t := 0^\lambda$
  for $i \in [0..n-1]$:
    $t := t \oplus F(k, x_i)$
  return t

   Prove the above scheme is not secure by constructing a distinguisher.

Definition 1 (MAC Scheme)**.** A MAC scheme is an algorithm *tag* with keyspace K. The scheme is **secure** if the following indistinguishability holds:

$$\boxed{\begin{array}{l} k \leftarrow K \\ \\ get(m): \\ \quad \textsf{return } tag(k,m) \\ \\ check(m,t): \\ \quad \textsf{return } tag(k,m) = t \end{array}} \quad \approx \quad \boxed{\begin{array}{l} k \leftarrow K \\ S := EmptySet \\ \\ get(m): \\ \quad t = tag(k,m) \\ \quad S := S \cup \{(m,t)\} \\ \quad \textsf{return } t \\ \\ check(m,t): \\ \quad \textsf{return } (m,t) \in S \end{array}}$$

That is, the adversary is given oracle access to a procedure that produces MAC tags and to one that checks MAC tags are valid (but is not given access to the key k).

2. **(4 points, C++ programming problem)** In class, we saw both CBC-MAC and an adjusted version called ECBC-MAC. We said that ECBC-MAC works for variable-length messages, but that CBC-MAC does not. Construct a MAC distinguisher that proves CBC-MAC (defined next) is insecure when used with messages of length $n \cdot \lambda$ for variable n. In the following, let $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ denote a secure PRF family:

$$\boxed{\begin{array}{l} CBC.tag(k, x_0, ..., x_{n-1} \in \{0,1\}^\lambda): \\ \quad t := 0^\lambda \\ \quad \textsf{for } i \in [0..n-1]: \\ \quad\quad t := F(k, x_i \oplus t) \\ \quad \textsf{return } t \end{array}}$$