CS 427/519: Homework 5

Due: Monday February 26, 10pm; **typed** and **submitted electronically**.

> **Important notes** for this homework and all future ones:
>
> - When asked to show that something is **insecure**, please clarify what libraries you are going to distinguish. Explicitly write the code of the distinguisher / calling program. Explicitly derive the output probability of the distinguisher in the presence of each library.

1. Let F be a secure PRP with block length λ. Consider the encryption algorithm below:

```
Enc(k, m):
  r ← {0,1}^λ
  x := F(k, r ⊕ m)
  return (r, x)
```

Write the decryption algorithm. Then show that the scheme does **not** have CCA security. (Describe a distinguisher and compute its advantage.)

2. Let E be a CPA-secure encryption scheme and M be a secure MAC. Show that the following encryption scheme (called encrypt & MAC) is **not** CCA-secure:

```
E&M.KeyGen:
  k_e ← E.KeyGen
  k_m ← M.KeyGen
  return (k_e, k_m)

E&M.Enc((k_e, k_m), m):
  c ← E.Enc(k_e, m)
  t := M.MAC(k_m, m)
  return (c, t)

E&M.Dec((k_e, k_m), (c, t)):
  m := E.Dec(k_e, c)
  if t ≠ M.MAC(k_m, m):
    return err
  return m
```

Describe a distinguisher and compute its advantage.

3. When a user creates a new account at a website, they receive a browser cookie containing $(d, \mathsf{MAC}(k, H(d)))$, where: d is a string of the form "`username|timestamp`", H is a hash function, and k is the website's secret key. The timestamp reflects the time that the cookie/account was created, encoded in format `yyyymmdd`; usernames must consist of entirely lowercase characters `a-z`. When a user connects to the website, the website checks that the MAC is valid and that the timestamp is in the past. A new user can request an account for any (available) username.

In general $\mathsf{MAC}(k, H(d))$ is a secure MAC of d, but suppose the website inexplicably uses $H(d)$ = first 5 bytes (40 bits) of $\mathrm{MD5}(d)$. I already have an account on the server with username `mikero`, and I won't let you see my authentication cookie.

Find and show me a username (other than `mikero`) that you can register for on this homework's due date (Feb 26) which will allow you to authenticate as `mikero`. **Describe** the MAC forgery you have in mind and show **how** you found it, and **why** it takes the amount of time that it does.

Note: I chose the parameters for this problem so that you will not be able to solve this problem if you take the wrong approach. You can easily do 2^{20} work before the due date but probably not 2^{40}. My solution typically takes between 5 and 10 seconds to find a forgery. If yours hasn't stopped after a few minutes, then you are probably doing it wrong.

Tip: You should be able to find an MD5 library implementation in your language of choice. In lieu of that, you can use the Linux command line as follows:

```
$ echo -n "mikero|20180219" | md5sum
014faa2a897a42e9af9d0b8f25087e8f  -
$ echo -n "mikero|20180219" | md5sum | cut -c1-10
014faa2a89
```

`echo -n` sends its argument to md5sum without adding a trailing newline character. `cut -c1-10` returns the first 10 characters of md5sum's output. Since md5sum returns the hex-encoded hash, the first 10 hex characters correspond to the first 5 bytes.

grad. Show that a scheme has CCA$ security **if and only if** the following two libraries are interchangeable. That means a security proof in both directions.

$\mathcal{L}^{\Sigma}_{\text{left}}$

$k \leftarrow \Sigma.\text{KeyGen}$

$\underline{\text{CHALLENGE}(m \in \Sigma.\mathcal{M})}$:
  return $\Sigma.\text{Enc}(k, m)$

$\underline{\text{DEC}(c \in \Sigma.\mathcal{C})}$:
  return $\Sigma.\text{Dec}(k, c)$

$\mathcal{L}^{\Sigma}_{\text{right}}$

$k \leftarrow \Sigma.\text{KeyGen}$
$D :=$ empty assoc. array

$\underline{\text{CHALLENGE}(m \in \Sigma.\mathcal{M})}$:
  $c \leftarrow \Sigma.\mathcal{C}(|m|)$
  $D[c] := m$
  return c

$\underline{\text{DEC}(c \in \Sigma.\mathcal{C})}$:
  if $D[c]$ exists: return $D[c]$
  else: return $\Sigma.\text{Dec}(k, c)$

Note: In $\mathcal{L}_{\text{left}}$, the adversary can obtain the decryption of *any* ciphertext via DEC. In $\mathcal{L}_{\text{right}}$, the DEC subroutine is "patched" (via D) to give reasonable answers to ciphertexts generated in CHALLENGE.