CS 5435:
Security and Privacy Concepts in the Wild
Homework #1

Due: Before class on 24 Sept. 2014
90 points (+ 20 bonus)

Instructor: Ari Juels TA: Sid Telang

Problem 1: Name three incentives an attacker might have for paying $325 (or more) for a high-profile Twitter account password. At least one of these should make money for the attacker. [15 points]

Problem 2: Recall that many password-reset questions are weak, in the sense of having a high guessing probability.

Question 2.1: For an online service whose users are U.S. inhabitants, give an estimate of the average number of randomly selected accounts an attacker would have to try to answer the question “What’s your mother’s maiden name?” correctly. Assume one guess per account. (Hint: Use statistics available at http://www.census.gov/genealogy/www/data/2000surnames/index.html.) [5 points]

Question 2.2: Find the weakest password-recovery question you can online that wasn’t discussed in class. Give an estimate of the guessing probability and explain how you arrived at it. [12 points]

Problem 3: The Ruritania National Bank (RNB) encrypts the four-digit PINs of its customers under a fixed, secret, randomly generated 256-bit key K. It may (or may not) be helpful to know how the RNB encrypts a PIN P for user U: The bank converts the PIN into a big-endian binary representation, pads it out by appending 0s to yield a 128-bit plaintext block, and then encrypts this block under key K using AES-256. The notorious Zenda League hacked the RNB to protest the bank's refusal to provide free checking accounts to its customers. They posted a spreadsheet with the encrypted PINs of 10,000 of RNB's customers, represented in hexidecimal. You'll find this spreadsheet (in the file RNBencryptedPINs.xlsx) on the course site.

Question 3.1: What adversarial model is the RNB assuming, i.e., what is the RNB assuming about the confidentiality provided by the encryption scheme? [4 points]

Question 3.2: Show an attack outside this model. What PIN belongs to the customer Ebenezer Scrooge? Note that a '*' in the customer name column denotes a customer other than Ebenezer Scrooge. (Hint: See http://www.datagenetics.com/blog/september32012/.) [10 points]

Problem 4: The NYC Taxi Authority recently released records of taxi trips in which medallion numbers were concealed via hashing with MD5. (See the Ars Technica article at http://bit.ly/1o0196n.) You can find a full list of current medallion numbers at https://data.cityofnewyork.us/Transportation/Current-Medallions/avwq-z233. (So can a potential attacker...)

Question 4.1: What is the maximum number of attempts needed to crack the hash of a medallion number (assuming no precomputation and no database of issued medallions)? (Hint: The number of possible medallion numbers reported in the Ars Technica article is wrong. Another hint: There are three authorized medallion formats, DLDD, LLDDD, and LLLDDD, where D stands for 'digit' and L for 'letter'.) [5 points]

Question 4.2: What is the security goal of the NYC Department of Transportation in this setting, i.e., why did they hash medallion numbers? [3 points]

Question 4.3: On the course site (file medallionhashes.txt), you'll find a list of SHA-256 hashes for some LLLDDD-format medallions that are active at the time of this exercise. Crack them! Also indicate who owns the corresponding taxis. (Hint: To verify the correctness of the SHA-256 code you use, here's the hash of medallion SBV381:

`95fa2dcf62a7ad26b29a08a03fe16c7a3aa341afb4e62999e12b6311d0d43f6a`.

The SHA-256 option at www.quickhash.com may also be helpful.) [15 points]

Question 4.4: Suppose that VIN numbers and licensee names were concatenated with medallion numbers prior to hashing active medallion numbers. Would this fix the problem and achieve the desired security goal? Why or why not? [5 points]

Question 4.5: Design a cryptographic scheme (a mechanism) using SHA-256 that would permit NYC to generate a unique pseudonym for each taxi medallion and meet the desired security goal. (Note: Given NYC's difficulties, maybe there's a startup or at least an interesting project somewhere in here. . .) [5 points]

Problem 5: Oklahoma Otis, the famous gambler, wants to start a gambling site for single-player roulette. (A standard American wheel has 38 slots. Otis thinks there's a business for an innovative, bigger wheel, one that instead has 39 slots.) He designed the following pseudocode client-server cryptographic protocol for his site.

```
C → S:  cslot                          % client submits bet cslot
S:      y = truerand();
        sslot = y mod 39;              % server chooses random slot
        if cslot = sslot, then
            win = true                 % client wins
        else
            win = false                % client loses
S → C:  sslot, win                     % send winning slot and outcome to client
```

(Some notation: C stands for client, S for server. The pseudocode $X \to Y$ denotes transmission of a message from entity X to Y, while X: denotes a local computation by entity X. The function SHA256 implements SHA-256; you may assume it acts as a random oracle. The function truerand() returns a 256-bit random string and the symbol $\|$, used below, denotes string concatenation.)

Question 5.1: Otis's friend, Mississippi Mabel, the famous sharpshooter, shot down his idea. She said his protocol won't work because the server can cheat without being detected. What is the security goal of the client? [2 points] How can the server defeat this security goal? [2 points] Is there any way in Otis's protocol for the client to detect server cheating? [2 points]

Question 5.2: Mabel came up with a clever protocol to enable the client to detect a cheating server immediately. Here is her protocol. (The 256-bit key K was generated at random.)

```
K =         7c49 70fe 78bd 92ea e9c2 22db f690 5ffe
            35ab 2c65 2a4d c6d4 bc52 9880 e133 8be2

$C$ :       a = SHA256(cslot || $K$)
$C \to S$ : a
$S$ :       y = truerand();
            sslot = y $\bmod 39$
$S \to C$ : sslot
$C \to S$ : cslot
$S$ :       if a = SHA256(cslot || $K$) AND cslot = sslot then
                win = true
            else
                win = false
$S \to C$ : win
$C$ :       if cslot = sslot AND win = false
                cheated = true
            else
                cheated = false
```

Why doesn't Mabel's protocol work, i.e., how can a server ensure that it never awards a win to a client but cheated = false? [5 points]

Question 5.3 (Bonus): Fix Mabel's protocol so that it does that does work, i.e., it allows a client to detect unfair play immediately. [10 points]

Problem 6 (Bonus): Recall that a MAC is a "tag" that may be appended to a message (plaintext or ciphertext) to ensure that tampering can be detected by the receiver. MACs are typically constructed using hash functions. A common example is HMAC.

The Chief Cryptographer of the RNB used SHA-256 to design a new, custom MAC called RuriMAC. Unfortunately, while RuriMAC cannot forged by an eavesdropper that doesn't intercept a valid (message, MAC) pair, and does make recovery of the secret key infeasible, it *can* be forged by an adversary that intercepts a single valid (message, MAC) pair. Give a possible design for RuriMAC. [10 points]