CS 161 Spring 2010 — Computer Security, Paxson/Wagner — HW 4

Due Thursday April 15, 5:00pm

Instructions: Submit your solution by Thursday, April 15, 5:00pm *electronically*. Write up your answers in either PDF format or plain text (7-bit ASCII), in a separate file for each question. Your files should be named `q1.txt`, `q2.txt`, ..., `q4.txt` (for plain text), or `q1.pdf`, ..., `q4.pdf` (for PDF). You may mix-and-match text and PDF if you like. Also, include a file called `collaborators.txt`, containing a list of everyone you collaborated with (see below). Then run `submit hw4`.

You may collaborate with at most 3 other CS161 students on this homework. Collaboration may involve discussing the problems with other students, but must *not* involve sharing solutions or having access to anyone else's solutions *at any time*. You must develop and write up your solutions *entirely on your own*. You must never read or copy the solutions of other students, and you must not share your own solutions with other students.

At the beginning of each file, include your name, class account (e.g., cs161-xy), and the *number* of the discussion section where you want to pick up your graded homework (e.g., 102; not your TA's name or the time). We will deduct points if this is missing for any of the files for your answers to Q1–Q3 or if you list the wrong class account. (You don't need to include this for `collaborators.txt`.)

1. (40 pts.) Blind Signatures and Digital Cash

(a) For your answers to this question, please use the following notation: use _ for subscripts (e.g., x_i) and ^ for superscripts (e.g., m^3).

It would sometimes be useful to have electronic forms of payment that provide anonymity the way that paper cash does. Blind signatures provide a way to do that. In a blind signature, the contents of the document being signed are never revealed to the signer.

Here is one blind signature scheme that uses RSA signatures. Bob has a public key n (his RSA modulus) and a private key d. Alice wants Bob to sign the message m blindly.

1. Alice chooses a random value r uniformly at random from the set $\{r \in \mathbb{Z} : 1 \leq r < n \text{ and } \gcd(r,n) = 1\}$. Alice blinds the message by computing $t = mr^3 \bmod n$.
2. Bob signs t by computing $u = t^d \bmod n$ and returning u to Alice.

Show how Alice can unblind u to obtain a valid signature $s = m^d \bmod n$ on m. In other words, show how Alice can compute s, based upon the information she knows and the information she has received from Bob. (Of course, Alice does not know Bob's private key d, so your solution must not assume knowledge of d.)

(b) In the protocol above, suppose Bob does not know the message m that Alice wants signed, but Bob somehow knows it must be either m_0 or m_1. Can Bob distinguish which is the case, more effectively than random guessing (e.g., is there some strategy he can use to guess correctly with probability $3/4$ or greater)? Justify your answer carefully.

HINT: The function $f(r) = r^3 \bmod n$ is one-to-one: if $r \neq s \bmod n$, then $r^3 \neq s^3 \bmod n$.

(c) Going back to our desire for digital cash, our requirements for a digital cash protocol are:

1. The bank cannot tell where the customer is spending his/her digital cash. In particular, the bank does not learn which merchant the customer spends his/her money at.
2. The merchant cannot identify the customer from the digital cash he/she provides to the merchant.
3. The merchant has some way of checking that the digital cash is not counterfeit.
4. The digital cash cannot be copied and spent twice.

Below we lay out one digital cash protocol. In our protocol there are three players: Carol, the customer; BankOBits, the bank; and Maxine, the merchant. Let coin $= (c,s)$ where $s = \mathrm{Sign}_{\mathrm{BOB}}(c)$ and $c =$ the value of the coin in dollars.

Imagine the following protocol:

a. Carol wants some digital cash in the amount of c dollars. She blinds c, obtaining $t = cr^3 \bmod n$.
b. Carol opens a secure channel to BankOBits and sends BankOBits the following: c,t.
c. BankOBits authenticates Carol (you can assume this is done correctly), decrements Carol's account balance by c dollars, and returns the signed, blinded coin: $u = t^d \bmod n$.
d. Carol can now unblind u to recover the signed coin, coin $= (c, c^d \bmod n)$.
e. When Carol presents the coin to Maxine some time later to buy valuable goods from Maxine, Maxine can verify the bank's signature and feel confident the coin is valid.
f. Finally, Maxine can present the coin $(c, c^d \bmod n)$ to BankOBits to get the money deposited into her account. BankOBits will first verify the signature before honoring the coin and increasing Maxine's account balance by c dollars.

With this protocol, one way that Carol can cheat is to double-spend. That is, she can make a digital copy of c^d and present it to two different merchants. Name one other way that Carol can cheat the bank to get free money.

(d) Dr. Alizarin suggests that we fix the protocol by making the denomination c the same for all coins: the bank will announce in advance that every coin signed under the modulus n will be worth exactly one dollar. Now, the coin will be derived from a random 128-bit string z. In particular, the coin will be given by coin $= (z,s)$ where $s = H(z)^d \bmod n$.

Whenever a merchant receives a putative coin from a customer, the merchant immediately deposits it with the bank. When a merchant presents a signed coin to the bank for deposit, BankOBits checks its database to make sure a coin with the same string z hasn't been previously deposited. If the coin has not been previously deposited and is validly signed, BankOBits increments the merchant's balance, adds z to the database, and returns a success message to the merchant; otherwise, BankOBits returns an error message to the merchant. The merchant ships the goods to the customer only after receiving a success message from the bank.

Will Dr. Alizarin's scheme prevent counterfeit/copied coins? In other words, does it satisfy requirements 3 and 4 above?

(e) Does Dr. Alizarin's scheme satisfy requirement 2 above?

(f) Suppose Carol is one of 10,000 customers withdrawing coins on Monday, and 500 merchants deposit coins on Tuesday. Assuming Alice spent some of her coins at some merchant on Tuesday, can the bank identify exactly which merchant Alice spent her digital cash at?

Based on this, do you think that Dr. Alizarin's scheme satisfies requirement 1 above? You can assume BankOBits has been wildly successful and has many customers, and that customers tend to withdraw coins from the bank well in advance and spend them only much later.

(g) Prof. Jasper is unhappy that Dr. Alizarin's scheme requires the merchant to contact the bank immediately. Prof. Jasper proposes an alternate scheme that allows the merchant to accept digital cash, even if the merchant is not Internet-connected at the time of purchase. In particular, Prof. Jasper proposes that a coin should have the form coin = $(c||z,s)$ where $s = \text{Sign}_{\text{BOB}}(c||z)$, i.e., $s = m^d \bmod n$ where $m = H(c||z)$. (Here $||$ denotes concatenation of bit strings.) Whenever a merchant presents a signed coin to the bank for reimbursement, BankOBits checks its database to make sure a coin with the same string z hasn't been previously deposited.

In addition, if the bank does detect double-spending it would like to know if it was the merchant or a customer who tried to cheat. To aid in this, BankOBits requires that after the merchant has verified the signature on $(c||z,s)$, the merchant asks the customer to provide a random 128-bit string *id*, and the merchant provides $(c||z,s,id)$ to BankOBits. If, when the merchant presents $(c||z,s,id)$ to the bank, the bank detects double-spending (i.e., some other coin with the same z was previously deposited), the bank will then compare the value of id on the coin provided by Maxine to the value of *id* stored in the database with the doubly-spent coin. If the values of *id* are the same then the bank knows the merchant cheated. If they are different then the bank knows that the customer cheated.

Will Prof. Jasper's scheme work? That is, will the bank be able to tell whether it was the merchant or Carol who cheated in the case of doubly-spent money?

2. (30 pts.) Onion Routing

(a) Give two reasons why usability might be considered a security *requirement* for using onion routing for anonymity (such as by systems like Tor).

(b) Bob is concerned about his privacy online so he always uses an onion router (the same as a "mix" discussed in lecture). He makes sure that all his outgoing traffic (e.g., HTTP requests, DNS requests) goes via the onion router. However, Bob does not take any steps to scrub the contents of his requests of identifying information. For each of the following, briefly describe whether it could be used by a web server Bob visits to learn Bob's identity, and explain why or why not.

i. Cookies

ii. Web bugs

iii. Referer field

iv. Google Analytics

(c) Suppose Alice sends a message to Bob using the onion routing protocol described in class. Neither Bob nor any of the onion router mixes (other than the first) along the circuit know the identity of Alice. How would you augment the protocol to allow Bob to send a reply to Alice without learning the identity of Alice? What could Alice include in her initial message that Bob could then use to get a message back without learning her identity? You may add new functionality to all onion routers, but you must explain how your new functionality would work.

3. (30 pts.) Reasoning about memory-safety

(This problem looks back to a topic addressed earlier in the class. Keep in mind that the final exam will be comprehensive across all of the course topics.)

Consider the following C code:

```
/* (a) requires:________ */
int sanitize(char s[], size_t n) {
  size_t i=0, j=0;
```

```
    while (j<n) {
      /* (b) invariant:_______ */
      if (issafe(s[j])) {
        s[i] = s[j];
        i++; j++;
        /* (c) invariant:_______ */
      } else {
        j++;

        /* (d) invariant:_______ */
      }

      /* (e) invariant:_______ */
    }
    return i;
  }
  int issafe(char c) {
    return ('a' <= c && c <= 'z') || ('0' <= c && c <= '9') || (c == '_');
  }
```

We'd like to know the conditions under which `sanitize()` is memory-safe, and then prove it.

Find the blank space labelled `(a) requires:` and fill it in with the precondition that's required for `sanitize()` to be memory-safe. (If several preconditions are valid, you should list the most general precondition under which it is memory-safe.)

Then, find the four blanks (b)–(e) inside `sanitize()` and fill each of these with invariants so that (1) each invariant is guaranteed to be true whenever that point in the code is reached, assuming that all of `sanitize()`'s callers respect the precondition that you identified, and (2) your invariants suffice to prove that `sanitize()` is memory-safe, assuming that it is called with an argument that satisfies the precondition that you identified.

Submit the final version, including the code and with all blanks filled in with the proper invariants. You can find a blank template in the file `/home/ff/cs161/hw4-files/q3.txt`.

4. (0 pts.) Optional: any feedback?

Optionally, feel free to include feedback in a file called `q4.txt`. What's the single thing we could to make the class better? Or, what did you find most difficult or confusing from lectures or the rest of class, and what would you like to see explained better?

Your answers will not affect your grade. Feel free to be frank: we appreciate all feedback, even (especially) critical feedback.