CSci 5271: Introduction to Computer Security

Exercise Set 5 **due: December 7, 2012**

Ground Rules. You may choose to complete these exercises 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 handed in by December 7.

1. All Mixed Up. The file `rounds.txt`, posted on Moodle, contains the output of a "toy" example of a mix. In this example, there are 260 users, `a0` through `z9`. Each user has one "friend" chosen according to a power-law distribution (so that some friends are more popular than others). The mix itself has a batch size of 32. In each batch, a set of 32 users is picked uniformly at random, and each of these users picks one of its two friends to send a message through the mix. The output file lists the senders (on the lines beginning with "`S`") and recipients (on the lines beginning with "`R`") of each batch of 32 messages, but not the correspondence between them. For this problem, you should perform an "extended statistical disclosure attack" to discover the friends of three "target" users. You can read more about this attack at

`http://freehaven.net/doc/e2e-traffic/e2e-traffic.pdf`

Your target users' "names" will all end in the digit 0, and the letters you use should come from the beginnings of the first names of your group members, so Alice, Fred and Ginger would target `a0`, `f0` and `g0`. (If you have less than three group members, use last names as well; if you work alone, pick an arbitrary third letter). For your write-up explain briefly which "target" users you chose, whom you identified as their friends, and what procedures you took to identify those friends.

2. Anonymous Adversaries The "Sybil" attack is a general attack on security protocols that involve many computers or identities. The basic idea of the attack is to acquire as many identities as necessa2ry to violate whatever assumptions the protocol makes about parties working together. Clearly, anonymity schemes could potentially make such attacks easier, although many of the most popular schemes make it fairly easy to prevent this (for example, by just blocking all participants in the scheme). On the other hand, many of these schemes can themselves be vulnerable to Sybil attacks.

(a) "Remailers" are anonymous email servers that essentially implement a mix cascade. The "basic" mix cascade work as follows: each node assembles a "batch" of messages to decrypt and jumble together. If the batching works by waiting until N messages are received, the $N-1$ attack can be applied: the adversary sends $N-1$ messages to the mix, whose destinations he knows. Then when a sender sends the N^{th} message, its destination is obvious. One possible defense against this is for the mix to wait to mix a batch until it has seen messages from K different senders. Why doesn't this work?

(b) "Low-latency" networks like Tor try to mitigate the Sybil attack as follows: suppose an adversary controls f fraction of TOR nodes. Then the adversary should be able to trace at most f^2 fraction of connections through Tor. (Thus, an attacker who uses a Sybil attack to join even $f = 1/2$ of the nodes still should only be able to trace $1/4$ of the connections.)

The reason that Tor has this goal (rather than a stronger one, for example f^3 rather than f^2) is that an adversary who controls the first and last nodes of a Tor connection can trace it, even though the packets leaving the first node are decrypted before they go to the last node. Explain how this is possible.

3. Vote (often) by mail. The reason most cryptographers who work on voting dislike "vote-by-mail" and/or widespread use of absentee ballots is the possibility of *coercion*: it is easy to "sell" your vote (where the price could be such things as lack of physical or mental harm, continued employment, and so on) because the "buyer" can watch you fill out your ballot and mail it in. One commonly proposed countermeasure to this attack is to allow each voter to cast multiple ballots, with only the most recently submitted ballot being counted. Discuss any potential attacks that could arise as a result of this procedure.

4. Cut and Choose. Many cryptographic voting protocols use something called a "zero-knowledge proof scheme" at some point in the protocol to convince one party that another party has followed the protocol. For example, one party (the "prover") may need to prove she knows a certain secret without revealing the secret to the other party (the "verifier").

A core idea in many of these protocols is the "cut and choose" technique in which the prover produces two options for the verifier: if (and only if) the prover can guess which option the verifier will choose, she can "cheat" the other player. As long as the verifier follows the proof protocol, he can only be fooled with probability $\frac{1}{2}$. Repeating this process many times can make it exponentially difficult for the prover to cheat. [1]

In many paper-ballot systems, a state-level authority sends ballot boxes to each precinct in the state. These boxes are locked to prevent the precincts from tampering with them, e.g. by removing votes. Describe a cut and choose protocol to prevent the state-level authority from "pre-stuffing" the ballot boxes. Specifically, if there are k precincts, your protocol should allow the state authority to cheat with probability at most 2^{-k}.

[1] Since the prover can always cheat if she knows which option the verifier will choose, the verifier never finds out anything about the prover's secret; this is because (knowing his choices) he could have made up a sequence of messages from the prover – by himself – that would have looked exactly the same to him. This is what makes the protocol "zero-knowledge."