CSci 5271: Introduction to Computer Security

Exercise Set 5 **due: Thursday, December 5th, 2013**

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. An electronic (plain text or PDF) copy of your solution should be submitted on the course Moodle by 11:55pm on Thursday, December 5th.

1. Remailer doppelgangers. (20 pts) 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 necessary to violate whatever assumptions the protocol makes about parties working together. Anonymity schemes could potentially make such attacks easier, although many of the most popular schemes make it fairly easy to prevent this (for example, it's easy to block Tor users from using your website at all, if you want). On the other hand, many anonymity schemes can themselves be vulnerable to Sybil attacks.

"Remailers" are anonymous email servers that essentially implement a cascade of mixes. 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. Explain why the Sybil attack makes this defense ineffective.

2. Ganging up on Tor. (30 pts) Continuing with the theme from the previous question, you can also see why we wouldn't expect attackers trying to compromise the security of Tor to just run a single malicious relay: they might try to run a large number of relays. It seems hard to avoid this problem completely in a low-latency network, but Tor aims to limit the damage to the following degree: suppose an adversary controls a fraction f of Tor nodes. Then the adversary should be able to trace at most an f^2 fraction of connections through Tor. Thus, an attacker who owns 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 in order to link the real world identity at the entry node to the browsing destination at the exit node. This is possible even though the cells are under layers of encryption the attacker doesn't control at the entry, while they're decrypted at the exit (thus the attacker can't just match up the contents of the cells). Suppose you were an attacker in this situation: how would you best match up the entry-node and exit-node flows?

3. Vote (often) by mail. (30 pts) The reason many security folks and cryptographers who work on voting object to "vote-by-mail" 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, or continued employment, instead of cash) 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 some of the trade-offs involved with this defense. If you were a vote buyer in an election with this defense deployed, what might you do? Can you think of any other negative side-effects?

4. Cut and Choose. (20 pts) 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.

We can also apply this principle to a paper-ballot system. Suppose that 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. However this leaves the possibility of a converse problem: how do the precinct authorities know that the locked boxes delivered to them are empty? A malicious state-level authority could "pre-stuff" the ballot boxes before they even get used. On the other hand, if we just gave the precincts keys, they could open the boxes to see that they were previously empty, but then they could stuff the boxes themselves.

Suppose the "acoustic side channel" of shaking the boxes to hear if anything rattles is out of bounds. Describe instead how to use 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}.