CSci 5271: Introduction to Computer Security

Exercise Set 3 **due: November 6th, 2014**

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, November 6th.

1. Vote (often) by mail. (15 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?

2. TCP-Unfriendly. (25 pts) TCP's "congestion control" mechanism relies on *end-hosts* (i.e., users) to respond appropriately to network congestion by backing off their sending rate. One potential problem with this mechanism is what's called by economists the "tragedy of the commons." Suppose Alice knows that everyone else obeys TCP's congestion control mechanism. Then if she continues sending at the same rate, everyone else will slow down a little bit more and she will get better service from the network. So Alice has no motivation to obey TCP congestion control (other than the fact that not doing so involves finding or writing her own TCP stack—details, details) and in fact neither does anyone else. But if no one obeys the mechanism, the network (commons) becomes useless, which is the tragedy.

(a) Bob the Network Builder has an idea about how to solve this problem. He reasons that congested routers can see the *exact* state of a TCP connection. So if a particular connection does not slow down in response to dropped packets, the router can send a `RST` packet to each end of the connection. This will cause both ends of the connection to drop the connection, much more painful than just dropping an odd packet or two. From a *security* standpoint, what's the problem with Bob's idea—that is, if I'm an unscrupulous user intent on communicating at a high rate, can I circumvent this mechanism?

(b) When Bob realizes that reset packets aren't sufficient, he proposes a more direct approach: *blacklisting*. Under this idea, routers that notice TCP senders that don't respond to dropped packets appropriately will just stop routing packets for that sender. List several ways in which this is both ineffective against adversaries and a generally bad idea if adversaries got wind of it.

3. (Mis-)using message authentication codes. (30 pts) Armed with a copy of Schneier's *Applied Cryptography* from a used bookstore, Sly can't wait to design his own encrypted thinga-madoodad protocol. He starts off with a super-secure key exchange protocol that ends with Alice and Bob sharing secret keys for encryption (K_e) and authentication (K_a). Now he wants to design a secure symmetric channel using these keys.

(a) Sly decides at first that he wants to use a CBC-MAC based on AES with 128 bit blocks for integrity. He looks carefully at his key exchange protocol and realizes that an adversary can interfere to make Alice and Bob end up deciding on different keys. So the first message sent over by Alice will be $\tau_0 = \mathsf{cbcMAC}_{K_a}(0^{128}) = \mathsf{aesEncrypt}_{K_a}(0^{128})$. If Bob's local value doesn't check out, he aborts, otherwise the channel is usable. Afterwards, whenever Alice wants to send the message M over the secure channel, she'll compute $\tau_M \leftarrow \mathsf{cbcMAC}_{K_a}(M)$ and send the pair (M, τ_M) over the channel; Bob will check whether $\tau_M = \mathsf{cbcMAC}_{K_a}(M)$ and if so will conclude that Alice said M.

This is a pretty bad idea. Show how to use the values τ_0, M and τ_M to compose a message to Bob that will convince him Alice meant to say the two-block message (M, τ_M) instead of just M. Explain why your message will convince Bob that Alice meant to say (M, τ_M) rather than just M. Hint: try writing a recursive definition of CBC-MAC, and use the facts that for any string A, $A \oplus A = 0^{|A|}$ and $A \oplus 0^{|A|} = A$.

Since τ_M is just 128 random-looking bits, why is this a big deal?

(b) Sly's friend Sally notices the same attack on his scheme. She proposes a different method of authenticating (and encrypting) messages: ignore the key K_a. Instead, to authenticate and encrypt the message M, first compute $H(M)$ using SHA-256; then encrypt $(M, H(M))$ together, using AES-CTR encryption. So the message sent on the insecure channel would be $\mathsf{CTR\text{-}Encrypt}_{K_e}(M, H(M))$; Bob would decrypt the message using K_e, check that the last 256 bits of the plaintext are the hash of all of the previous bits, and accept the message if they are.

Show that this is also a bad idea: if Alice ever sends a ciphertext corresponding to the message M, where Eve knows M, Eve can generate a ciphertext corresponding to any message M', (of the same length as M) that Bob will accept. (For example, if Alice sends the message "ATTACK AT TEN AM" Eve can drop it and make Bob accept the message "GO BACK HOME BOB" instead.)

4. Protocol (an)droids. (30 pts) Two robots Artoo and C3-2-0 often fly on different starships and need to alert each other to their presence when their ships come in contact—otherwise they might accidentally blow each other up! They agree on a shared key K and a MAC algorithm that outputs 256-bit tags to use in the following protocol.

1. $A \longrightarrow C$: a random 256-bit string N_A and $\mathsf{MAC}_K(N_A)$.
2. C: on message n, t checks that $\mathsf{MAC}_K(n) = t$, and if so, he accepts A, otherwise he blows up the other party.
3. $C \longrightarrow A$: $\mathsf{MAC}_K(t)$.
4. A: on message t' checks that $t' = \mathsf{MAC}_K(\mathsf{MAC}_K(N_A))$. If so, he accepts C, otherwise he blows him up.

The idea here is that A proves he is A by correctly MACing N_A (which, if the key is secret, only A or C could do) and C proves he is C by MACing the MAC. But...

(a) A and C use this protocol for a while and then discover, to their dismay, that sometimes the evil galactic emperor, E, has been successfully fooling C into believing he is A. Even supposing that robot-in-the-middle attacks are prevented by speed-of-light limitations or some other plot contrivance, what is a simple way for him to do this?

(b) A and C decide that one way to prevent the attack is for C to remember every value of N_A used in a previous challenge and reject if one is ever reused. Suppose E sees one authentication between A and C. How can he fool C into believing he is A as many times afterwards as he wants?