Study Sheet for Sample Midterm

CS 236

Spring 2011

This is not precisely an answer sheet for the sample midterm. The nature of these questions, like those that will appear on the real midterm, is rather open ended, so there are many ways they could be answered "correctly" and get a good grade. But I do point out here important elements of what I would expect in an answer to each question.

1. An important point here is that the brothers are signing parts of what they agree to, not the entire thing. They would leave fewer opportunities for problems if, instead of signing individual clauses, they created a new proposed contract and performed a digital signature on that entire new contract. For instance, if Groucho only wanted clauses 12, 587, and 772, rather than creating three signatures (one each for these three clauses), he should create a new contract containing only those clauses and produce one signature over the entire contract.

   What could happen if they signed individual clauses? Well, neither Groucho nor Chico would know for sure which clauses the other had signed. Groucho might not forward his signature on clause 587, or Chico might not forward his signature on clause 42. Or Groucho did forward his signature on clause 587, Chico produced a "complete" contract containing clause 587, but did not sign that clause and did not sign the entire revised contract. How could Groucho prove that 587 should be part of the contract? And what if they later agree on a totally different set of clauses, which both sign? Can Chicho later sign clause 12 himself, produce Groucho's signed version, and claim that the agreed-upon contract includes clause 12? Just because Groucho liked clause 12 when he also liked 587 and 772 doesn't mean he wants it inserted into an arbitrary contract.

   In terms of third party tampering, there are limited possibilities. Harpo cannot create valid signatures for his brothers. But he can cause messages to be dropped, and he can record and replay messages. As one example, what if Groucho proposed clauses 12, 587, and 772, signing all three, then Chico counterproposed clauses 42, 199, and 353 (signing them), and then Groucho agreed to Chico's counterproposal and also signed 42, 199, and 353? What if Harpo slipped in Groucho's signature on clause 12 into the contract? Would Chico be able to know it was Harpo's fault?

2. For cryptographic capabilties, the capability for A must not be exactly the same as the capability for B. If it were, B could steal a copy of A's capability off the

network and pretend it was his. So, in such a system, presumably the crypto ensures that a particular capability is tied to the identify of a particular user. This implies that creation of a new capability requires new cryptography. A may have the right to create a capability for B, but he doesn't necessarily have the ability to create the crypto for that purpose.

Most likely, for A to create a new capability allowing B to read, but not transfer the read privilege, A will need to consult a trusted authority that can create capabilities. This authority would create the new capability for B and either give it directly to B, or send it to A for subsequent delivery to B. If we assume the trusted authority is on another machine (very likely), then we need secure delivery of A's intention to the authority. Otherwise, user C could intercept the message and replace B's identity with C's, resulting in a read capability for C, not B. Similarly, B must not be able to tamper with the request to alter it to give him read right transfer abilities.

The reference monitor needs to check validity of the capabilities. This implies that when someone (A, B, C, or someone else) wants to read the file X, the reference monitor can determine, first, that it is a proper capability, and, second, that the capability is being offered by its owner. What if C intercepted the new read capability given to B, then requested to read the file and furnished the new B capability, saying, "sure, I'm B, give me the file?"

3. The code to be run is almost certainly application level code. What does TPM itself offer? The ability to boot securely and to make attestations about the identity of a piece of code. What it doesn't offer, that we need, is some kind of guarantee that a piece of code a remote TPM has attested to is actually run on that machine. The problem we're worried about here is Evil OS Z asks its TPM to attest to the virus scanner, and Z provides the TPM the real virus scanner. The TPM produces an attestation. Then Z sends the attestation to the remote machine, but never actually runs the virus scanner. The remote machine didn't want to know that Z has access to a copy of the virus scanner. It wanted to know that Z was actually running it.

   So, how can we know that? What if the TPM attested that it booted boot loader A, which is trusted? Since TPM provides secure boot loading, we no longer have any doubt about which boot loader was run, unlike the earlier case of the virus scanner. OK, now how about if A is carefully designed and will only boot an OS that has been attested to? A gets the TPM attestation for the OS the machine owner wants to boot, determines if it trusts that OS, and, if so, loads it. If not, the boot loader refuses to load the kernel. If we do this, not only does the local machine know it's running A, but it can provide a strong attestation to remote machines that it's running A. What if A is well studied and is known to never, under any circumstance, request an attestation for an application and then fail to run that application? If that's true, we can attest that we're running A, and, if A tells us an attestation for the virus checker, we can trust that A ran the virus checker, and not that it merely obtained the attestation.

This really only deals with half of the question, proving to the remote user that we're running his code. How about the other half, being sure that his code is limited in its functionality? Does TPM offer us anything to help with that problem?