Assignment #7

Due: March 3, 2011

As discussed in class, one of the security features provided by a TPM is the ability to encrypt and decrypt secrets using an RSA key pair contained within the TPM chip. But the TPM is a hardware device, and hardware devices can fail, so when securing secrets with a TPM we must take into account how we're going to migrate those secrets to other hardware in the event of a failure. Similarly, when using the TPM's "sealed storage" feature to secure secrets to a particular software configuration, we have to take into account how such secrets can be migrated if the software configuration changes. In this problem set we'll look at three types of secret migration issues that can arise with TPMs.

First, let's define some common terminology. Let A and B be two TPM-enabled computers. Assume each TPM has an RSA-2048 key pair for encryption (let $K_{EPUB,A}, K_{EPRIV,A}$ be A's public & private keys, and respectively $K_{EPUB,B}, K_{EPRIV,B}$ for B). Let S be a (small, <1024 bits) secret that A knows; S will be the secret that we want to migrate in our various scenarios. Our TPM performs encryption of $value$ using key with the function $Encrypt(key, value)$, where key may be any public key. Decryption is similarly performed as $Decrypt(key, value)$.

Additionally recall the sealed storage feature of TPMs that allow you encrypt a secret to one or more platform measurements stored in platform configuration registers (PCRs). A sealed storage call looks like this:

$$Seal(key, value, PCR_{A_0}, PCR_{A_1}, \dots, PCR_{A_i})$$

This call will seal $value$ using some public key key to the configuration specified by the set of PCR values $PCR_{A_0}, PCR_{A_1}, \dots, PCR_{A_i}$. To unseal a sealed value, the command is:

$$Unseal(key, sealedvalue, PCR_{A_0}, PCR_{A_1}, \dots, PCR_{A_i})$$

Finally, recall that TPMs also have RSA signing key pairs, which we'll label $K_{SSIG,A}, K_{SVER,A}$ for $A's$ private signature and public verification key (B has similar keys). A TPM is able to perform a digital signature with its key by calling this function:

$$Sign(key, value)$$

Similarly, a TPM can "attest" to a particular PCR configuration by calling

$$Quote(key, value, PCR_{A_0}, PCR_{A_1}, \dots, PCR_{A_i})$$

1. **Migrating secrets between platform configurations on A.** Assume that machine A uses secret S as a master key for a full-disk encryption system. To defend against tampering with the OS boot sequence needed to start up the full-disk encryption system, A stores S on disk (in an unencrypted disk sector) using the TPM's sealed storage feature. Assume that the entire configuration of the OS boot sequence is measured and stored in PCR's 0-5 (six total registers).
   a) What TPM command does A use to seal S and create the ciphertext that may be stored safely on disk? What TPM command does A use to unseal this value later?
   b) Assume that a security hole is found in the OS and it is necessary to patch a part of the OS boot sequence. When the patch is complete, PCRs 4 and 5 will change and their new values will be PCR_4' and PCR_5'. Describe what steps the OS needs to take *before patching the OS* in order to ensure that the patched version of the OS will have access to S, without ever letting S be written to disk in the clear.

2. **Migrating secrets between machines with TPMs.** Now assume machine A needs to be taken out of service for some reason and so its secret S needs to be migrated to backup machine B. Describe a protocol that A can use with B in order to securely transmit S to B. A and B don't know each other's public keys in advance, but you may assume the existence of a PKI that both A & B are part of and which has issued certificates to $A's$ and B's various public keys that chain to a common root.

3. **Migrating secrets between machines with TPMs with assurance of equivalent software stacks.** After implementing the protocol you designed for Question 2, your boss points out that while the secret S is indeed transmitted securely from A to B using your protocol, there's nothing in the protocol that ensures that the software on B that receives S is the same as the software running on A. Modify your protocol to address your boss's concern. Your new protocol should have the following additional features:
   a. Before sending S to B (in any form), A should receive from B some proof that $B's$ OS boot sequence (measured by PCRs 0-5) is the same as $A's$.
   b. When S is transmitted to B, A should ensure that S will only be accessible on B if the same software stack is running on B as on A at the time the migration begins.