CSE 409, Fall 2011, Rob Johnson, http://www.cs.stonybrook.edu/~rob/teaching/cse409-fa11/
Alin Tomescu, December 2nd, 2011

Sealed storage

Overview of sealed storage

Sealed storage protects private information by **binding it to platform configuration information** including the software and hardware being used. This means the data can be released only to a particular combination of software and hardware. Sealed storage can be used for DRM enforcing. For example, users who keep a song on their computer that has not been licensed to be listened will not be able to play it.

```
$\boldsymbol{Seal(PCR_{value}, x)}$:

    $c = Enc_{S'_{TPM}}(x)$
    $t = Sig_{S_{TPM}}(PCR_{value}, c)$
    return $(C, PCR_{value}, t)$
```

PCR_{value} is a value stored in a PCR and not a PCR register #.

```
$\boldsymbol{Unseal(c, PCR'_{value}, t)}$:

    if $!Vrfy_{P_{TPM}}((PCR'_{value}, c), t)$
        return $error$
    if $PCR \neq PCR'$
        return $error$
    return $Dec_{S'_{TPM}}(c)$
```

Purpose: Allow the computer when it's in a certain state to store a message that can only be accessed when the computer is back in that state.

Quick recap: TPM, we had a PCR, a (S_{TPM}, P_{TPM}) keypair, a secret key S' used for encrypting sealed storage (never leaves the TPM).

If you want to seal a message x, you specify the value saved in the PCR as a parameter, since the PCR will contain a hash (more or less) of the software that is running on the computer.

Secure movie playing application example

You've got a **movie playing service** (MPS) and a client, who is untrusted.

Flickr allows you to **jump into a small secure environment** where you are guaranteed what's running is one secure trusted piece of code called `play1frame`.

When `play1frame` is running the PCR on the client will hold a hash of the `play1frame` code: $hash(17 \mid\mid \text{play1frame})$.

We would like to have the frame sealed for `play1frame` and delivered to the `play1frame` chunk of code.

The **problem** is that if the movie playing service will take the plaintext of the frame and seal it under its TPM's public key, then the client's TPM chip will not be able to decrypt it, since it will not know the server's TPM's secret key.

Solution: We'll have a Flickr module on the client called the **play setup module**.

The play setup module will be given the public key P_{MPS} of the movie playing service and it will **generate a special key k that will be used to encrypt the movie frames**.

The play setup module will ask the TPM to seal k such that only `play1frame` can unseal it:

$$a = seal\left(PCR_{p1f}, k\right)$$

The play setup module will also send the key k, encrypted, to the movie playing service. The MPS will be able to decrypt b using its secret key.

$$b = Enc_{P_{MPS}}(k)$$

At this point the MPS and the client agreed on the frame encryption key.

How it works:

- The movie playing service will encrypt the frame under k and send $Enc_k(frame)$
- The client will invoke the Flickr module to go into the secure environment, it will unseal a and get k
- The client will use k to decrypt the frame, play the frame and erase the memory

Benefits: Instead of having to verify the entire software stack, Flicker allows you to go into a small trusted computing base where the only thing you have to trust is the small piece of code that is executed in it.

Trusted computing applications:

- Antivirus
- Integrity protection system

Security usability

How do you **make security decisions easy to understand** for users?

- Phishing
- Sitekey
- Condition safe ceremonies
- Phorcefields
- Password managers
- Device pairing

Phishing

You can **impersonate a secure browser** session using JavaScript by popping up a browser window without a location bar which looks like and implements a secure Firefox browser window, in order to trick the user into believing he's in a secure session.

Security indicators:

- Lock (ignored and spoofable)
- Green location bar (ignored and spoofable)
- Sitekey

CSE 409, Fall 2011, Rob Johnson, http://www.cs.stonybrook.edu/~rob/teaching/cse409-fa11/
Alin Tomescu, December 2nd, 2011

Sitekey

With Sitekey, when you type in your username, the webserver takes you to a login page where there's **a picture and a word that you chose** and then there's a password box.

This image and this word are **stored on your computer in secure cookies** and a smart user is supposed to look for these before logging in.

A **phishing website will not be able to present the image and the word** since it will not be in possession of the secure cookies which are stored on the client side after the **secure machine registration** is done.

Secure machine registration (using secret question)	Send username and secret question answer	
		Set cookie for image and text
Logging in		Send login.html which displays image and text
	Send password	

A man in the middle attack is possible if the secure machine registration is not done.

Problems:

- Phisher can display a misspelled "Our award-winning Sitekey security feature is down for maintenance. Call in 24 hours if your image is still not displayed." and a gullible user will login using the phisher's interface.
- Phisher can try and phish the secret question answer as well and then obtain the image and text.

Condition safe ceremonies

Conditions safe ceremonies are very **simple** and are also the **best secure machine registration protocol.**

Bank probably has your email address anyway. Here's what happens when you want to register your computer:

Secure machine registration	User clicks on register my computer on the bank's website.	
		Bank sends an email to the user with a link containing a big random number in it.
	User clicks on the link (he really can't help clicking on it)	
		Bank sets the secure cookie on user's browser