

Recipe for Mobile Data Security: TPM, Bitlocker, Windows Vista and Active Directory

Tom Olzak
October 2007

If your business is like mine, laptops regularly disappear. Until recently, centrally managed mobile storage encryption solutions for Windows environments were either too costly, required users to carry a key-resident device, or relied on keys residing on local disk. Sometimes the best solution under these circumstances was the presence of administrative controls (i.e. policies) prohibiting users from storing sensitive information on local laptop drives. With the proliferation of TPM 1.2 across most laptop platforms and the release of Microsoft Windows Vista, most roadblocks to laptop data encryption have been removed.

In this paper, I explore the challenges facing security managers responsible for laptop data security, TPM technology, and how the features of a TPM can integrate with Microsoft's Bitlocker and Active Directory technologies to provide for more secure data on the road.

The Challenges

Vulnerable Laptop Data

When a mobile user's laptop is stolen, a variety of data might be potentially exposed. The following is a short list of some of the most common types of sensitive information that might be found on a laptop drive:

- **ePHI** – Electronic Protected Health Information, protected under the [HIPAA](#), is any information that can be used to uniquely identify a health care recipient.
- **PII** – Essentially, Personally Identifiable Information is any data that can be used to steal a person's identity. Most states have passed laws requiring businesses to take steps to protect this information as well as to notify and assist victims if this information is compromised.
- **Intellectual property (IP)** – Information on which an organization depends for competitive advantage is often located where it is easily accessible to mobile workers—on laptop drives.
- **Keys, passwords, and PINs** – Information used by the user or the system to protect local data or to access corporate networks is often stored on local disk and processed in the general memory space. Any information stored on local disk or accessible via memory is vulnerable to compromise through various methods, including those described in the following section.

Compromise of Encrypted Data

Even when sensitive information is encrypted, it is often susceptible to theft. When sensitive information—including the data types described in the previous section—is unencrypted during normal use, it can potentially be stored in areas on disk that are not protected, including (Olzak, 2007):

- **Windows pagefile.sys.** This file is used to swap data to and from RAM. Data that might be encrypted in its normally stored state may be stored unencrypted in pagefile.sys. By extension, information stored in RAM might also be susceptible to compromise.
- **Windows hibernation/sleep file.** When a Windows device goes into hibernation or sleep mode, it writes the contents of memory to disk. As with the pagefile.sys issue, information normally encrypted when not in use might be stored in plain text in these files.
- **User specified locations.** Users might decide to copy information from one disk folder to another. In situations where the source folder was configured for encryption and the target folder was not, sensitive information might end up unencrypted in violation of company policies or government regulations.

Finally, encrypted keys and other authentication and identification information stored on disk are subject to capture and cracking attempts. The ideal situation is to store these types of sensitive information in protected storage where no user or application has access. This is one of the functions of the Trusted Platform Module.

The Trusted Platform Module (TPM)

The TPM is an instantiation of the Trusted Computing Group's (TCG) specification for trusted computing (Bajikar, 2002). Among other things, it establishes a "chain of trust", beginning with a "root of trust, as depicted in Figure 1.

The root of trust is "The TPM hardware along with its supporting software and firmware..." validated through attestation of the TPM module (Bajikar). An endorsement key (EK) is used for this purpose. The EK is a public/private key pair generated by the TPM itself. Although the public key is made available to applications and services, the private key is never revealed outside the TPM. Attestation of the TPM is necessary to create a trusted platform.

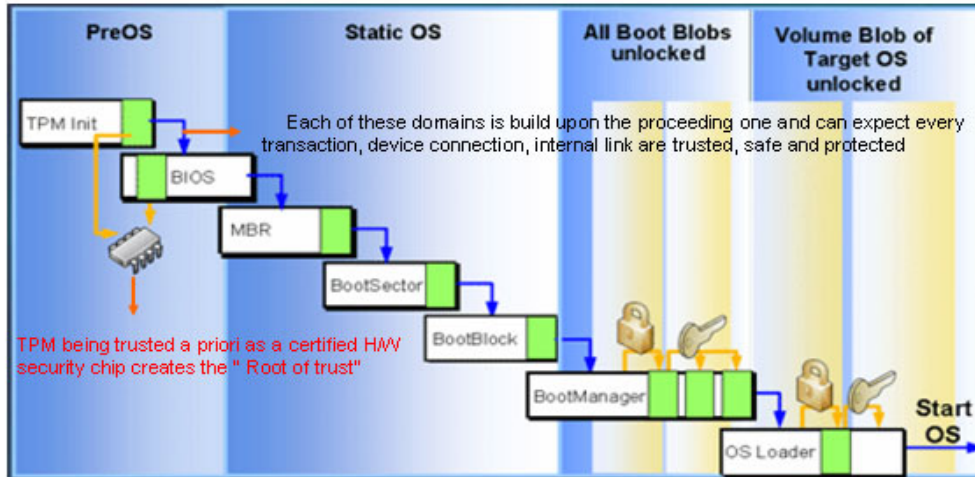


Figure 1: TPM Chain of Trust
(IEI Technology Corp., 2006)

On a trusted platform, every hardware and software component must be known and identifiable to enable the creation of the chain of trust. This is accomplished by validating the following characteristics of the computing platform's system components (Hardjono & Smith, 2007):

- Expected operation of each component must be measured and baselined
- Consistent behavior of each component must be verified or attested to at all levels in the chain

The expected configuration state of each component in the chain of trust is represented by a hash value stored in the TPM's Platform Configuration Register (PCR). Hash values are recomputed upon boot up and compared to those in the PCR to ensure no unwanted changes occurred.

Data encryption and attestation capabilities of the TPM

The TPM is typically a small microcontroller with cryptographic capabilities. As indicated earlier in this paper, software and hardware components outside the TPM have no access to the execution of the cryptographic function. The components that make up the cryptographic capabilities of the TPM include the following (Bajikar):

- **RSA Accelerator** – Performs up to 2048 bit RSA encryption and decryption.
- **SHA-1** – Cryptographic engine used to compute hash values for small amounts of data.
- **True Random Number Generator** – The TPM contains a true, hardware-based, random number generator. This is in contrast to software-based, pseudo-random number generators that produce results with lower entropy.
- **Protected storage for**
 - **Endorsement Key**
 - **Attestation Identity Key** – The AIK is used to enable platform attestation to a service provider.

- **Certificates**
 - **Platform certificate** – This security component is provided by the vendor of the platform. It provides attestation that the security components of the platform are genuine.
 - **Endorsement certificate** – The endorsement certificate contains the public key in the EK key pair. Used with the private key, it provides attestation that the TPM is genuine and the EK is protected.
 - **Conformance certificate** – This certificate is provided by the platform vendor or an evaluation lab. It provides accredited party attestation as to the security properties of the platform.
- **User's sensitive security data** (e.g. file or volume encryption keys)

Detailed chain of trust and TPM capabilities that allow trusted platforms to verify themselves to other devices and services as well as to verify secure configuration of platform components are outside the scope of this paper. However, additional information is available in [Trusted Platform Modules Strengthen User and Platform Authenticity](#) published by the TCG.

TPM key hierarchy

Keys created and managed by the TPM are linked in a hierarchy of parent and child (or leaf). Please refer to Figure 2.

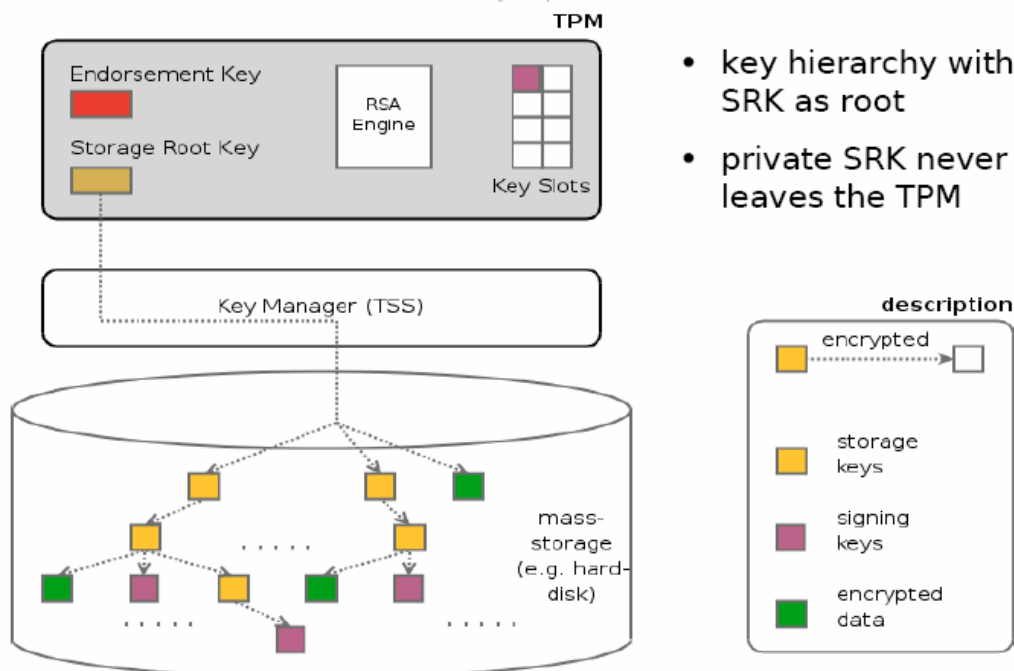


Figure 2: TPM Key Hierarchy
(Winkler, 2007)

The Storage Root Key (SRK) is a 2048 bit RSA key that is at the top of the TPM key hierarchy. Other keys include:

- Storage keys, used to encrypt other elements in the hierarchy.
- Signature keys are used for signing and must be a ‘leaf’ in the TPM key hierarchy.
- Binding keys
- Legacy keys, used for both signing and binding

The number of key slots is limited, so not all keys can be kept in TPM protected storage. Typically, the EK and the SRK are the only keys stored in the TPM itself. They are never exported. Some keys are exported for storage on disk or other medium. However, the private portion of the SRK key pair never leaves the TPM and all subordinate keys stored on disk are encrypted with the SRK’s public key. Since the private key never leaves the TPM, decryption of an exported subordinate key can only be performed within the TPM.

Using the TPM to create keys

One of the principle uses of a TPM is key creation. Using the onboard random number generator, the module is able to create a private/public key pair with high entropy. Figure 3 depicts this process.

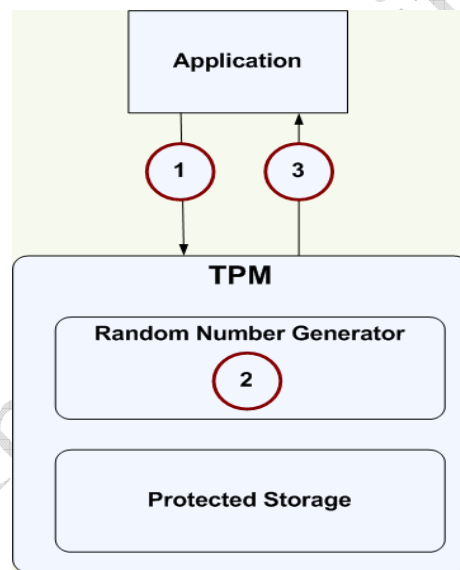


Figure 3: TPM Key Creation Process

Step 1: An application sends a request for a key pair to the TPM. The request must include a key password, the parent for the key, and the password for the parent.

Step 2: The TPM creates the key pair internally using its random number generator. The TPM concatenates the private key and the key password and encrypts the resulting string with the parent’s public key. This is known as “wrapping.”

Step 3: The wrapped key and the public key are returned to the requesting application. At this time, the key has not been “loaded” into the TPM.

An alternative to this process is to have an application create its own key pair. However, actual randomness is very hard to attain with the use of software-based key generators.

To leverage the TPM for key management, the key must be loaded. The loading process is shown in Figure 4.

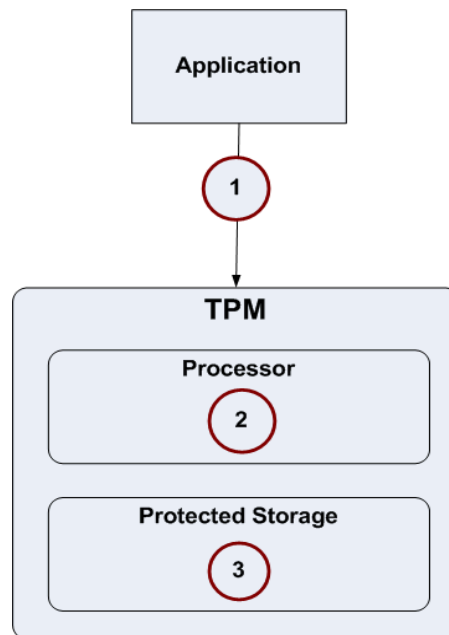


Figure 4: Loading a Key

Step 1: The application submits a wrapped key to the TPM along with the identity of the parent.

Step 2: The TPM decrypts the wrapped key using the parent key.

Step 3: The unwrapped key and key password are stored in protected storage.

Once stored in protected storage, the encryption key can only be accessed by an application that knows the key’s password. Also, since the key is not stored on disk, attackers cannot access it in a cracking attempt. In the next two sections, we’ll look at how TPM works with Microsoft Windows Vista, Bitlocker, and Active Directory. As we step through how these various components can be used to protect laptop data, I’ll demonstrate how keys stored in the TPM are accessed and used.

Protecting the Local Laptop Drive

For the purposes of this paper, I'm taking the position that the best way to protect sensitive information on a laptop drive is to encrypt it and that the laptop runs Windows Vista Enterprise in an Active Directory environment. Finally, we're going to assume that encryption must take place without the purchase and installation of additional laptop or enterprise products.

Two approaches to stored data encryption

There are two basic methods of encrypting data on our laptop. First, we can use Encrypting File System (EFS) to encrypt specific files or folders. Refer to Figure 5.

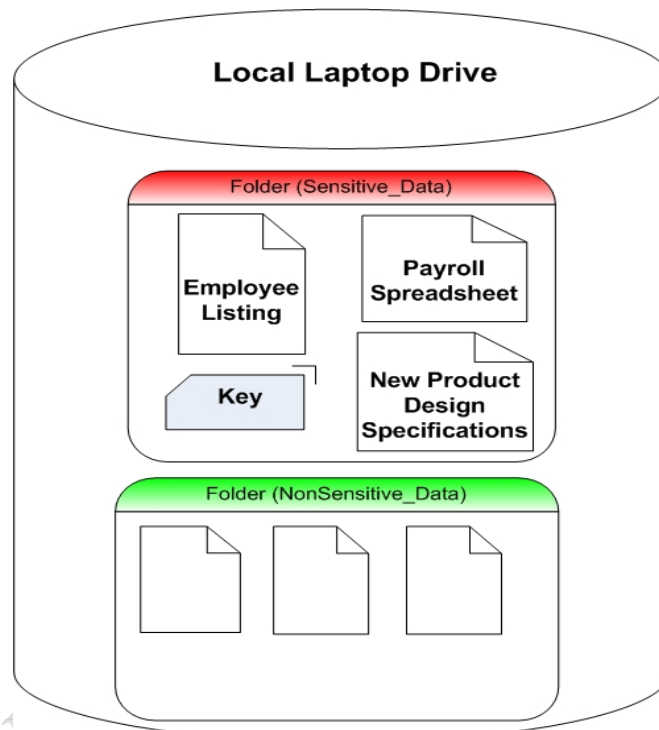


Figure 5: EFS Encryption

EFS is not a full disk volume encryption solution. In this example, it's used to encrypt a single folder called *Sensitive Data*. All files stored there are encrypted and safe from prying eyes. However, files in the other folder, *NonSensitive_Data*, are not encrypted. This is a good, basic approach to laptop data encryption, but it has some serious flaws. First, what happens if a user modifies *New Product Design Specifications* and accidentally saves it to *NonSensitive_Data*? This is the situation depicted in Figure 6.

Note that the original copy of the design specifications is still safely encrypted in its original location. However, an updated copy now exists in an unprotected folder. This is possible because files copied or written to locations outside an encrypted folder are not encrypted—even if they were encrypted in their original location.

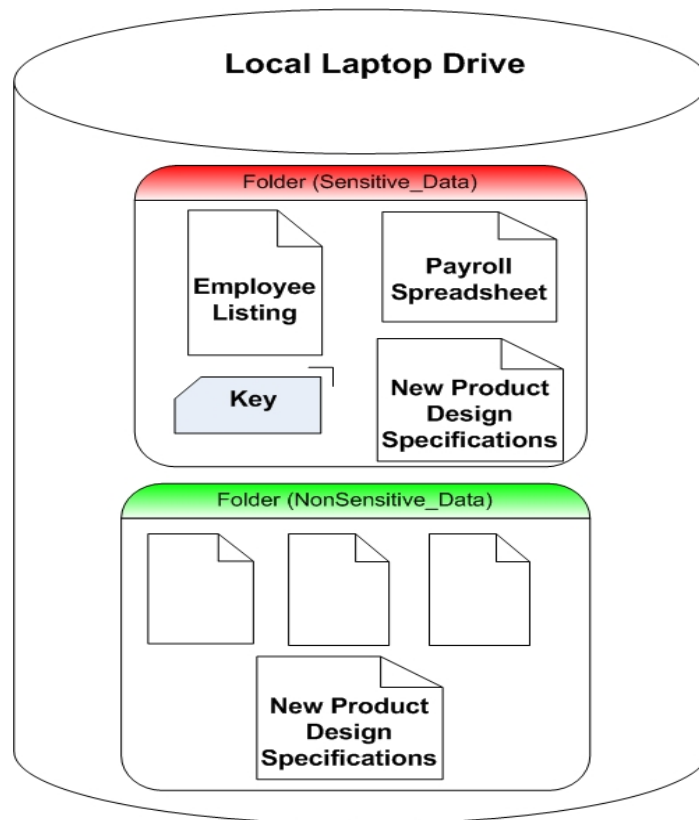


Figure 6: EFS Encryption File Copy Issue

Another potential vulnerability occurs when sensitive information is written to *pagefile.sys* or to the hibernation file. This is depicted in Figure 7. For example, when the user opens the product design file, it is decrypted and loaded into RAM. Any information in RAM is subject to being moved to the paging file as part of Windows' virtual memory management process (Microsoft, 2007). So any part or all of the sensitive information contained in the file could be written to *pagefile.sys*. Since *pagefile.sys* is not encrypted in our EFS example, an attacker can potentially extract confidential information by using one of the many available disk management utilities.

Similarly, when a laptop is placed into hibernation mode, an unencrypted file is placed on the local disk. This file contains the contents of memory at the time hibernation occurred.

The second method of encrypting a laptop drive is through implementation of full volume encryption. Instead of encrypting specific files or folders, full volume encryption can protect the entire local disk, including paging files, hibernation files, and sensitive content written into any folder. This type of protection is provided in a Vista Enterprise or Ultimate environment by using Bitlocker.

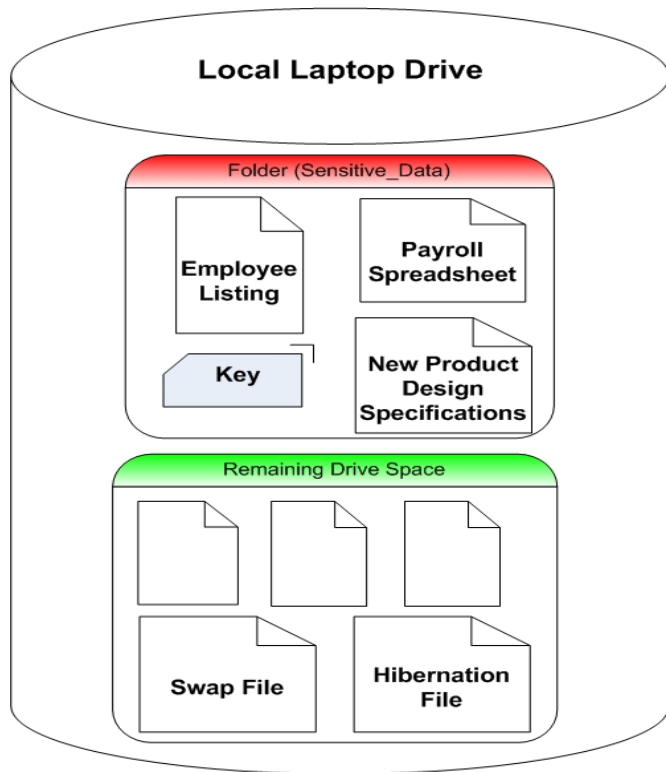


Figure 7: EFS Swap and Hibernation File Challenge

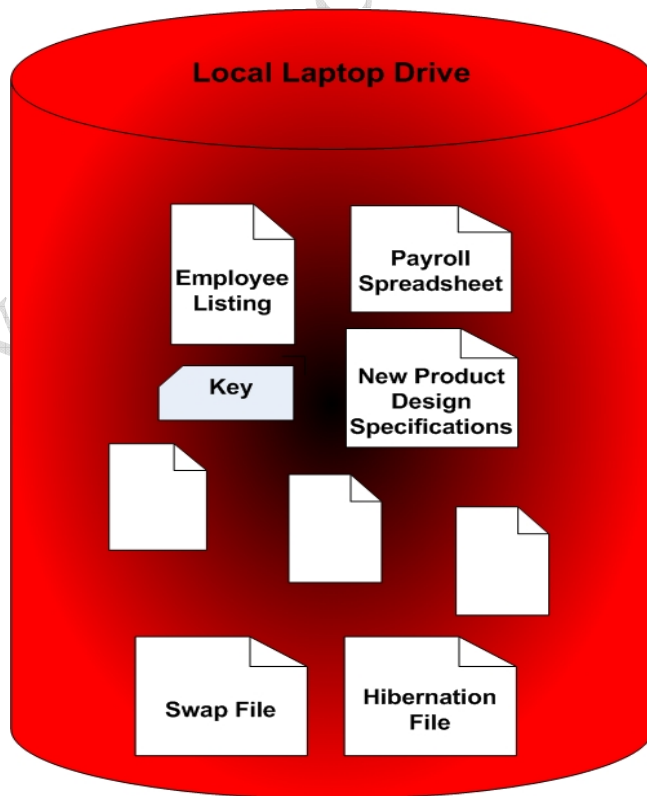


Figure 8: Full Volume Encryption

Figure 8 depicts a drive with Bitlocker full volume encryption enabled. No part of the disk volume containing the operating system is unprotected. In addition to protecting hibernation and paging files, this approach also makes it very difficult to obtain unauthorized access to key files, which are typically stored in the OS volume by default.

Depending on hardware compatibility, Bitlocker can also perform a validation of certain system components prior to loading the operating system. It accomplishes this using the hashed contents of the TPM's [PCR](#). The verification process is outside the scope of this paper.

Once the components are verified, Bitlocker then commits itself to encrypting and decrypting data on the OS volume and the OS volume only. If other volumes exist, and data on those volumes must be encrypted, [EFS](#) will have to be used.

Now let's take a look at how TPM and Active Directory functionality can be leveraged to facilitate effective full volume encryption with centrally managed data recovery methods.

Managing TPM and Bitlocker Encryption with Active Directory

As we saw in [TPM key hierarchy](#), the SRK is the top level key within the TPM. The public key in this 2048 bit asymmetric key pair is used in database encryption solutions to encrypt the Volume Master Key (VMK). The VMK, a 256 bit key, is then used to encrypt and decrypt the Full Volume Encryption Key (FVEK). A 128 or 256 bit AES key, the FVEK is used to actually encrypt or decrypt sectors as they're written to or read from the operating system volume. Let's take a look at how this works in reverse when the user accesses the encrypted volume.

If the system is using the complete functionality of Bitlocker, the first step after power up is to verify system components. Once verification is successful, the process depicted in Figure 9 is used to access the FVEK to allow access to protected data.

Step 1: The TPM uses the SRK's private key to decrypt the VMK.

Step 2: The unencrypted VMK then decrypts the FVEK

Step 3: The FVEK employs the kernel driver FVEVOL.SYS to read data from the encrypted volume. No unencrypted data are ever written to the protected OS volume.

One of the advantages of this process is that re-keying the drive is a simple matter of changing the VMK. No data on the drive has to be re-encrypted. Another advantage is the inaccessibility of the private portion of the SRK key pair. This key never leaves the TPM and is therefore not subject to cracking attacks.

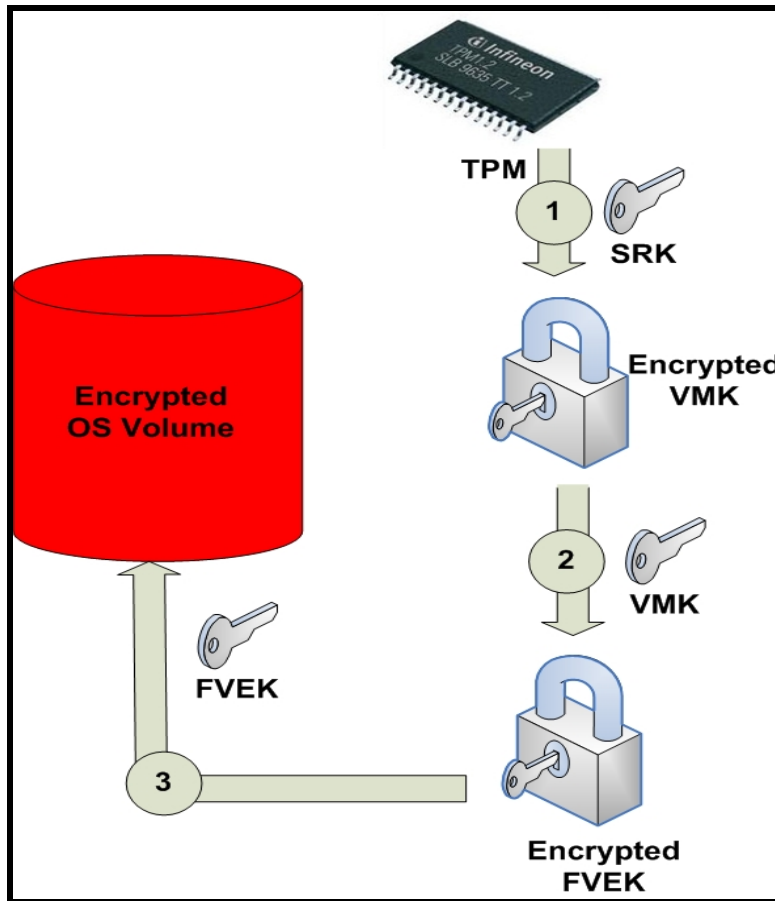


Figure 9: TPM/Bitlocker key access process

Figure 10 is a bar graph depicting the four primary methods of authenticating to a system protected by a TPM and Bitlocker. Note that security increases as you travel along the X axis and usability increases as you travel up the Y axis.

As depicted in the first bar, initiating this process with just TPM and Bitlocker can be as easy as simply entering the normal system password. However, Microsoft recommends against this single factor authentication method.

TPM-only mode offers the least amount of data protection. This mode protects against some attacks that modify early boot components, but the level of protection can be affected by potential weaknesses in hardware or in the early boot components (Microsoft, April 2007, p. 26)

Using a dongle, such as a USB device, is better. However, loss of the device can put the protected system at risk. It is also something else for the user to remember to carry along when out of the office.

Microsoft's recommended method—and one that balances ease of use with security—is the use of a PIN in conjunction with the TPM (Microsoft, April 2007, p. 32). When using this approach, the user must enter a password and a PIN. The PIN is restricted to a

numeric value entered via the function keys at boot up. Combined with the SRK private key, the PIN is required to decrypt the VMK.

The use of a token with a PIN is the most secure approach, but there is still the problem of the dongle.

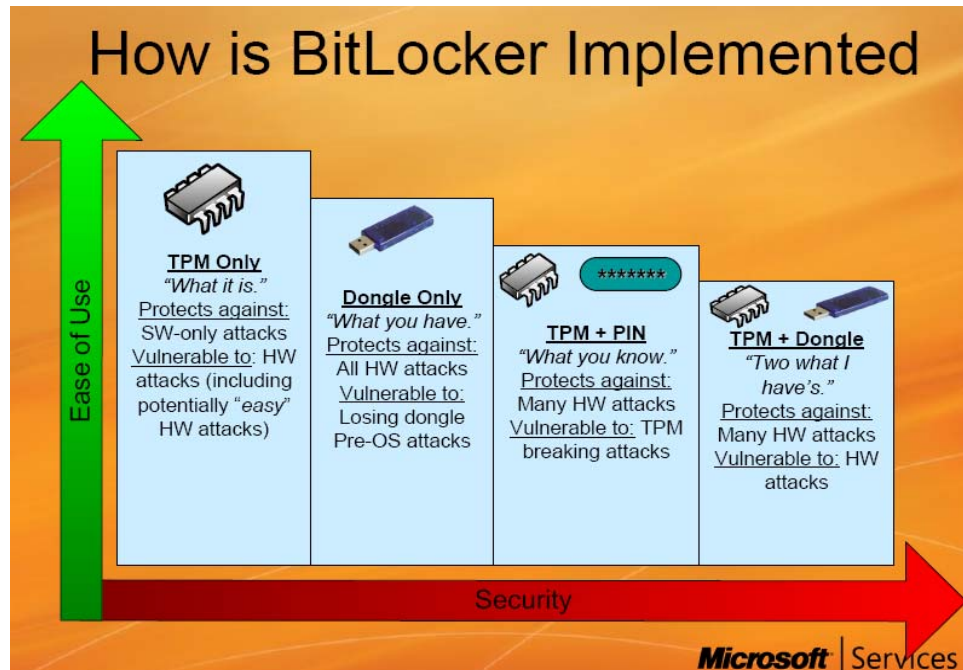


Figure 10: TPM/Bitlocker Access Methods
(Microsoft, June 2007)

Key management and data recovery are not an issue with Vista and Bitlocker. By extending the Active Directory Schema, recovery information can be stored within the machine object. This provides recovery information for protected systems when employees are no longer with the company, when PINs are lost, or when boot up components are modified.

Conclusion

The solution presented in this paper is just one way to economically protect sensitive data on distributed systems. It can be rolled out incrementally as part of a laptop end-of-life process or as a component in a network access control solution. To some degree, it addresses all the challenges defined at the beginning of this paper.

No, it isn't a perfect solution. Like most other security controls, a TPM/Bitlocker solution has vulnerabilities, including not securing non-OS volumes. However, it meets the best practice of applying reasonable and appropriate safeguards to prevent breaches of PII or ePHI.

© 2007 Thomas W. Olzak.

Tom Olzak, MBA, CISSP, MCSE, is President and CEO of Erudio Security, LLC.

He can be reached at <mailto:tom.olzak@erudiosecurity.com>.

Check out Tom's book, [Just Enough Security](#)

Additional security management resources are available at <http://adventuresinsecurity.com>

Free security training available at <http://adventuresinsecurity.com/SCourses>

www.InfoSecWriters.com

Works Cited

- Bajikar, S. (2002, June). *Trusted platform module (TPM) based security on notebook PCs – white paper*. Retrieved September 27, 2007 from http://www.intel.com/design/mobile/platform/downloads/Trusted_Platform_Module_White_Paper.pdf
- Hardjono, T. and Smith, N. (2007, June). *Integrity management for trusted platforms*. Retrieved September 28, 2007 from <http://www.naspa.com/integrity-management-trusted-platforms>
- IEI Technology Corp. (2006). *TPM function and application*. Retrieved September 27, 2007 from http://www.ieiworld.com/tw/news_content.asp?id=erbium/projectOBJ00244201&news_cate=news&news_sub_cate=all%20news
- Microsoft (2007). *What is virtual memory?* Retrieved October 4, 2007 from <http://windowshelp.microsoft.com/Windows/en-US/Help/fd3e9104-1076-4600-9f9d-8739849387ec1033.mspx>
- Microsoft (2007, April). *Data encryption toolkit for mobile PCs: security analysis*. Retrieved October 4, 2007 from <http://www.microsoft.com/technet/security/guidance/clientsecurity/dataencryption/analysis/default.mspx>
- Microsoft (2007, June). *Microsoft Windows Vista forensic jumpstart*. Retrieved October 10, 2007 from <http://www.techsec.com/TS-2007-PDF/Mo%20T6-1.pdf>
- Olzak, T. (2007, August). Protect endpoint devices from swap and hibernation data leaks. *CNET*. Retrieved September 27, 2007 from <http://blogs.techrepublic.com.com/security/?p=225>
- Winkler, T. (2007, March). *TPM – Trusted Platform Module*. Retrieved October 1, 2007 from http://www.iaik.tugraz.at/teaching/04_trustedcomputing/slides/tpm_vo.pdf