Attacking the BitLocker Boot Process*

Sven Türpe, Andreas Poller, Jan Steffan,
Jan-Peter Stotz, and Jan Trukenmüller

Fraunhofer Institute for Secure Information Technology (SIT),
Rheinstrasse 75, 64295 Darmstadt, Germany
{sven.tuerpe,andreas.poller,jan.steffan,jan-peter.stotz,
jan.trukenmueller}@sit.fraunhofer.de
http://testlab.sit.fraunhofer.de

Abstract. We discuss five attack strategies against BitLocker, which target the way BitLocker is using the TPM sealing mechanism. BitLocker is a disk encryption feature included in some versions of Microsoft Windows. It represents a state-of-the-art design, enhanced with TPM support for improved security. We show that, under certain assumptions, a dedicated attacker can circumvent the protection and break confidentiality with limited effort. Our attacks neither exploit vulnerabilities in the encryption itself nor do they directly attack the TPM. They rather exploit sequences of actions that Trusted Computing fails to prevent, demonstrating limitations of the technology.

1   Introduction

One promise of Trusted Computing is better protection of system integrity. Various applications can profit from mechanisms that protect software on a computer from being tampered with. To this end, a v1.2 TPM supports authenticated boot, keeping track of the boot process and eventually basing operations such as sealing and attestation upon the result. This is one step short of what theory suggests for best security: stopping the boot process of fixing the issue as soon as a manipulation has been detected [1, 2].

This leads to the question what the implications of this difference are in practice. We explore this question for one particular software design and application: BitLocker. Included with some editions of Microsoft Windows Vista and Windows Server, BitLocker encrypts volumes on disk and uses the sealing function of a v1.2 TPM for part of its key management. We devise several scenarios for targeted attacks that break the confidentiality BitLocker is supposed to protect.

Note that there are two distinct attack strategies against which BitLocker should ideally protect. *Opportunistic* attacks use only what is easily obtained under common real-world conditions. An example is recovering data on a disk or computer that has been bought in used condition from somebody else, or stolen somewhere. A *targeted* attack is different in that the attacker attempts to get access to data on a specific, predetermined disk or machine, usually within some

---

time and resource constraints. According to Microsoft, BitLocker is designed to withstand at least opportunistic attacks. Considering targeted attacks as we are doing here may be beyond its specification. However, disk encryption along with TPM-based key management might be expected and perceived to be more powerful than what the manufacturer is willing to promise, and we deem it useful to explore the actual security properties and limitations regardless of claims and cautionary notes.

The remainder of this paper is organized as follows. Section 2 briefly describes the design of BitLocker, focusing on its key management and how it is using the TPM. Our adversary model and security considerations are outlined in section 3. Section 4 describes attack scenarios that seem feasible and either yield secret key or data or achieve some important steps towards a successful attack. Causes and contributing factors are discussed in section 5. Section 6 outlines our practical implementation of the attack, followed by the conclusions in section 7. Related literature is referenced where appropriate but not specifically discussed.

2   An Overview of BitLocker

This section only mentions facts about BitLocker that we will use in this paper. For further details refer to the documentation available from the manufacturer [3, 4] and from unofficial sources [5].

2.1   Integrity Model and Design Constraints

BitLocker works, at boot time, as a component of the boot loader and later as a driver of the operating system kernel. Its design assumes that the kernel boots from a BitLocker-protected volume, that BitLocker sufficiently protects the integrity of data on this volume, and that anything that happens after initiating the OS boot process is sufficiently controlled by other security mechanisms. We do not challenge these assumptions here; see [6–8] for two known attacks against the running system.

According to these assumptions, BitLocker has to protect the integrity of the boot loader and its execution environment up to the point where the kernel can be read from the locked volume. This code is read from an unencrypted part of the disk and needs to be supplied with a secret key for the AES algorithm. This is where the TPM is being used in. BitLocker uses the *sealing* function to store all or part of its key material in such a way that it becomes accessible only if the platform configuration as represented by the PCR values is in line with the reference configuration. The reference configuration is determined by the administrator accepting the current system configuration at some point in time. This adoption of a reference configuration is initially done during BitLocker activation but can be repeated at any time from the running Windows system.

2.2   Key Management and Recovery Mechanisms

Apart from special cases—BitLocker can also be operated without a TPM or with all key material being managed by the TPM—key material is divided. One

part is managed by the TPM and released only if the platform is in the trusted state, the other is supplied by the user as a password and/or key file on a USB memory stick.

If the TPM works as desired, there is no way according to the design to gain access to all required key material if the platform state measured is different from the reference state. This is intended if the platform state is modified by an attack, it is not, if state is modified for a legitimate reason and the change can not be reverted easily, e.g. after BIOS update or hardware repair. BitLocker therefore offers two recovery mechanisms, the recovery password and the recovery key. Both are designed to circumvent the TPM and supply BitLocker with its secret key independent of the current platform state. The recovery mechanisms don't correct the problem, though. This is left to the administrator who, after the recovery boot, may set a new reference state from the running system.

The actual encryption key does not change during the recovery process.

2.3   User Experience

The user experience hides most of the details. When switching on their PC, users will see a text-mode prompt for their PIN and/or USB stick. If the platform is not in reference state they will next be prompted for their recovery key or password. Otherwise the boot process will resume after the PIN or USB key have been provided. Depending on how the computer is being used, users may experience a recovery prompt from time to time, e.g. after accidentally leaving a bootable CD or DVD in the drive or when a bootable USB stick is plugged into their computer.

3   Security Considerations

3.1   Security Objectives

The primary security objective is confidentiality of any data stored on the encrypted volume. Encryption alone, however, cannot guarantee confidentiality as a system security property. Its scope has—at least—two intrinsic limitations. First, disk encryption is not expected to protect cleartext data before en- or after decryption. The system must provide further security mechanisms to provide such protection. Second, encryption does not solve confidentiality problems but rather shifts them: from the data to the key(s). Encryption therefore cannot be more secure than its key management allows it to be.

A secondary objective is integrity of the data stored on disk. We consider integrity here only insofar as it is a prerequisite for protecting cleartext data and keys.

3.2   Attack Success Conditions

There are several distinct conditions that, if achieved by an attacker, would violate the primary security objective:

- The attacker obtains all or some ciphertext and breaks the encryption.
- The attacker obtains the cleartext from place or situation where cleartext is normally handled. The attacker works outside the scope of the encryption in this case and exploits a vulnerability elsewhere.
- The attacker obtains all or some ciphertext as well as sufficient key material, and decrypts the ciphertext.

An attack may achieve everything at once, or it may comprise a sequence of steps, each of which brings the attacker closer to success. We assume that steps can be arranged in arbitrary sequence. But we require that no step except the last one may spoil continuation, e.g. by preventing necessary subsequent steps or by clearly alerting the user to the ongoing attack before it is finished.

3.3 Attack Situations

When accessing the system, the attacker may encounter one of several different situations. Situations can be thought of as a set of parameters that the attacker does not control. The situation found, together with capabilities, determine what the attacker can achieve during the visit. Though not being able to control the situation, in a targeted attack the attacker can wait for the right moment. Some parameters, however, may have a very low probability of changing. An example is the configuration of a disk encryption scheme once it has been installed. Situational parameters are:

- The time and channel available for undetected interaction with the target system. This is really a continuum but we can roughly distinguish three classes of physical access: brief visits (up to few minutes), temporary control of the device (up to a few days), and permanent possession. Another channel is remote communication. We assume that the attacker must successfully install software on the target machine to gain such a channel, and the channel is available only while this software is running.
- The boot state of the target computer. The system may be powered on and fully booted, or it may be powered off. When it is running, encryption keys are present in RAM. The attacker can power it down at the risk of the change being noticed by the user. If the system is powered off, the attacker is not able to fully boot the system without the user-managed secrets. He may boot his own software, however; this is indeed one of the actions that using a TPM should protect against.
- System configuration. There is a vast amount of system configurations that an unprepared attacker may encounter. For a particular target system, however, the configuration rarely changes. If it does, and the system has a TPM and uses it, expected or deliberate changes are likely to be accepted, changing the reference configuration.

If the attacker encounters the system running and has enough time available, the attack is successful immediately: keys can be read from memory using one of the known online attack techniques, and used to decrypt disk contents.

3.4 Adversary Capabilities

With physical access to the target system the attacker can perform one or multiple actions depending on the time available, the state of the system and whether (later) detection is acceptable or not. We are thinking along the lines of [9] here but do not attempt to establish a comprehensive model. Our lists below are likely incomplete.

Brief visits During a brief visit the attacker could:

Power off or on the system, depending on its current state. Powering off implies that the original state cannot be restored if the attacker does not know the secrets requested at boot time. Although the change is noticeable, the attacker may get away with it if the change is plausible to the user. If the system is originally powered off, the attacker can revert to the original state at any time. The attacker will not be able to boot into the regular system without knowing the boot-time secrets.

Modify boot code if the system is powered off or can plausibly be left in this condition. Whether and when such a modification might be detected by the user depends on various side conditions, particularly on the use of a TPM and the actions the code performs. We can expect such code to be executed at least once.

Steal the system, turning a brief visit into permanent possession of the machine. Theft will usually be detected. The attacker may or may not be able to preserve the boot state of the machine, depending on whether it has a sufficiently charged battery or not.

Replace the machine with one that looks exactly the same. This requires some preparation and investment but seems feasible. In addition to getting hands on the target machine permanently the attacker retains the ability of interacting with the target user. The attack will be noticed as soon as the copy behaves differently than the target would, or any other distinguishing feature becomes visible and noticed.

Copy small amounts of data from the disk but not an entire volume or disk. This leaves no traces if the system is powered off, or changes its state if it was powered on. (We assume the software running on the machine cannot be exploited to this end.) Copying entire disks fails not because of any security mechanism but due to the amount of time required.

Possibly copy small amounts of RAM contents if a DMA-capable interface is available and supported, and the system is up and running.

Temporary control If the attacker can spend more, but sill limited time with the system, additional actions become available:

Install a concealed hardware extension such as a key logger. Whether the system has to be powered down if encountered running depends on the type of extension and various details of the hardware design.

Copy the entire disk or an entire volume of data. This leaves no traces if the system is powered off, or may change its power state if it was powered on. However, given sufficient time the attacker may be able to disconnect the disk from the system without resetting the machine, connect it to another machine, copy all data, and reconnect the drive to the target machine.

Non-destructive attacks against a TPM or other hardware components such as a TPM reset attack [10, 11]. Such attacks may have specific prerequisites. A reset attack against a TPM, for instance, requires that the attacker knows the proper sequence of PCR values during regular boot. One way of obtaining this sequence would be to record it during such a boot process, which requires the ability to boot the unmodified system to the point where the TPM is used.

Permanent access An attacker in permanent possession of the target system has all the capabilities described above. The difference from temporary control is that the attacker does not have to hide anything from the user. Destructive attacks become an option.

Communication. Communication with the attacker is possible whenever the attacker manages to run his own software on the target system or controls existing software with communication capabilities. There are different modes of communication. The most common are: storing data locally where they can be picked up later; transmitting data through a network interface card to the local wired or wireless network; or using an IP network if the computer is connected to one. None of these options requires that the attacker uses the operating system installed on the target system.

4 Attack Strategies

4.1 Plausible Recovery

The attacker modifies the BitLocker code on disk, adding a backdoor. Such a backdoor could be as simple as saving a clear key in some location on disk or elsewhere in the system from where it can be retrieved later. This modification will of course be detected the next time the system is started by a legitimate user. However, the attacker hopes that the user applies one of the TPM-independent recovery mechanisms to overcome the problem. The attacker later visits the system again to collect the key. Encrypted volume data could be copied during each visit to the target system as the actual encryption key does not usually change.

Requirements This attack requires:

 - that recovery mechanisms are used at all, and
 - that the attacker can physically access the target machine at just the right time without taking it away permanently, and

– that the reported platform validation error seems plausible for the victim.

One obvious implementation of this attack would be to wait for a situation that plausibly changes the state of the platform, such as a repair. It may also be possible to provoke such a situation. The attacker will then have to sneak into the process somewhere before the user accepts the seemingly legitimate modification. This would mask the malicious change with the legitimate one.

Result If the attack succeeds, the attacker has successfully planted a backdoor into the system in such a way that *all* software-based security features could be circumvented. The attack is unlikely to be noticed by the victim. In order to get both the encrypted data and the secret key the attacker will have to visit the target system at least twice. However, the backdoor may also use other channels to leak cleartext data, possible increasing the risk of detection.

4.2 Spoofed Prompt

Similar to the plausible recovery attack, the attacker modifies BitLocker on the target system and lives with the fact that the TPM will detect this modification. The attacker adds code that spoofs the user interface of BitLocker up to the point where the user has given up his secrets. The malicious code may spoof either the normal-operations UI or the prompt for a recovery key.

Requirements This attack requires that the attacker can physically access the target system. It is not necessary that the attacker takes the system away permanently.

Result The attack is easily detected as soon as secrets have been provided to the spoofed prompt. After detection it is generally possible to prevent the attacker from interacting with the compromised system again. Also, the TPM will refuse to unseal its part of the key material while the platform is in this modified state. If a recovery prompt is successfully spoofed and operated by the user, the attack will yield sufficient key material for decryption of a volume.

Extensions Although it may work under some circumstances, this attack does not appear very critical. However, the next subsection describes a more critical extension.

4.3 Tamper and Revert

The tamper and revert attack extends the spoofed prompt attack. Instead of simply accepting that platform modifications can be detected, the attacker attempts to exploit tampering yet hiding it. This becomes surprisingly easy if one additional boot cycle is possible. The attacker could make a temporary modification to TPM-verified code. If we stick to the spoofed prompt example, this means to add a cleanup function to the malicious code, whose purpose it is to

restore the former platform state. After a reboot—which might be initiated by the malicious code after showing a bogus error message—the platform state as measured will be compliant with the reference PCR values again.

Requirements Requirements are similar to those of the *spoofed prompt* attack. In addition the attacker needs to get away with a boot cycle after platform integrity failure without disturbing the victim so much as to spoil further steps of the attack. Depending on how the credentials or keys obtained are transmitted to the attacker, a further visit to the system may or may not be required.

Results This attack yields copies of keys controlled by the user. In a simple implementation these keys will end up in clear somewhere on the target system itself but more sophisticated approaches can be imagined, for instance sending the key somewhere using a built-in WLAN interface. Additional effort is required on the attacker's part to gain access to TPM-managed key material.

4.4   Replace and Relay

This is a hardware-level phishing attack. The attacker replaces the entire target machine with another computer prepared for the attack. The replacement, when turned on, produces all the messages and prompts that the original machine would have produced. Up to the point where BitLocker would start, it takes all user inputs (via keyboard or USB) and relays them to the attacker, e.g. using radio. The attacker, being in possession of the unmodified original system, uses this information to start up the stolen computer.

Requirements This attack requires that:

- the attacker is capable of replacing the BitLocker-protected machine altogether with an identically-looking copy, and
- the machine is plausibly turned off or in suspend-to-disk mode when the legitimate user returns, and
- the replacement device is capable of relaying user input to the attacker.

The attacker will have to remain—or leave some device—in proximity to the target until the next boot is initiated by the victim. The attacker will also need some prior knowledge of non-secret facts, specifically everything that might be needed to perfectly reproduce the user experience.

Result As a result of this attack, the attacker receives the user-controlled secrets. Depending on the mode in which BitLocker is deployed on the target system, the result is either key material or authentication credentials or both. Either one can be used in conjunction with the unmodified system to start up the operating system. Security mechanisms of the operating system remain intact; another attack will be required to actually access any encrypted data. Such attacks exist [8, 7]. The attack will likely be noticed right after the victim provided credentials or keys to the spoofed machine. This attack may be combined with any attack that yields the TPM-managed portion of the key material.

Extensions and Variants A more sophisticated version of this attack involves two-way communications, turning the replacement into a terminal of the stolen target machine. This would probably require quite some additional effort but might extend the time span between success and detection of the attack. All variants of this attack may also be attempted against recovery mechanisms, which yields sufficient key material to decrypt disk contents immediately.

4.5 Preemptive Modification

This attack is similar to the plausible recovery attack, but at a different point in time. The recovery attack targets systems on which BitLocker has already been activated. Preemptive modification attacks earlier, before BitLocker has been activated at all.

When defining the reference state for future booting, the operator has no choice other than using the current platform state. BitLocker does not provide the user with any means of verifying that this current state has or hasn't any particular property. If an attacker manages to modify critical parts of the platform before BitLocker is activated, this modification therefore goes unnoticed and will be incorporated into the trusted (but not trustworthy) platform state.

Requirements Preemptive modification requires that the attacker gets physical access to the target system *before* BitLocker is activated. Arbitrary modifications are possible at that time that would weaken the security of the BitLocker instance affected forever. Another physical visit may be required later to retrieve a disk image for decryption or leaked cleartext. However, the system may also be modified in such a way that it leaks data at runtime. Everyone who gets physical access to the machine or OS installation media before BitLocker setup is a potential attacker.

Results The attacker potentially gains read and write access to all data handled on the system throughout its lifetime. This attack is hard to detect unless there are additional means of verifying the integrity of executable code against external references.

5   Causes and Contributing Factors

This section identifies factors that make the overall system—a PC with BitLocker and Trusted Computing technology—vulnerable to the attacks described above. Factors include fundamental properties of the security mechanisms involved as well as features in the design and implementation of BitLocker and the Trusted Computing platform.

Authenticated boot. Theory states that secure booting requires an appropriate action if the measured state deviates from the reference. The boot process could be stopped or it may be possible to fix the issue once it had been

detected [2]. So far, however, we have only authenticated boot. The TPM does not enforce a trusted platform state, it only refuses to unseal a key if the state is currently not trusted. This leaves loopholes for attacks, but also makes it easier to provide recovery mechanisms if they are desired.

No trusted path to the user. BitLocker uses secrets to authenticate the user: the PIN and key material. The channel between the legitimate user and the system in a trusted state is prone to spoofing and man-in-the-middle attacks (replace and relay; spoofed prompt; and tamper and revert). or specifically, the system lacks context-awareness and the user is unable of authenticating the system. Similar problems exist elsewhere, e.g. ATM skimming. Both directions of authentication can be discussed separately:

No context-awareness. The BitLocker has no means of determining whether the computer is under control of a legitimate user or somebody else. It simply assumes that whoever provides the correct key or credential is a legitimate user. Although requesting a PIN or key may be interpreted as authentication, it is not a very strong one, and adding stronger authentication may be difficult.

Lack of system authentication. While BitLocker is capable of authenticating its user at least in the weak sense described above, the user has no means of verifying authenticity and integrity of the device. Keys and passwords are to be entered into an unauthenticated computer.

History-bounded platform validation. The Trusted Computing platform detects and reports platform modifications only within the scope of the current boot cycle. BitLocker uses this feature through the sealing function of the TPM and does not add anything. The system is therefore unable to detect, and react to, any tampering in the past that has not left permanent traces in the system.

Incomplete diagnostic information. If current and reference state are out of sync, it is difficult or even impossible for the user or administrator to determine the exact cause(s). This leaves the user with a difficult choice: to use recovery mechanisms blindly, or not to use them at all. The lack of diagnostic information contributes to the plausible recovery attack. Note that detailed diagnostic information may not be required where a trusted state can be enforced, e.g. by re-installing software from trusted sources.

Lack of external reference. This is another issue that has already been discussed in the literature. BitLocker is capable only of using any current platform state as a reference for future boot cycles. There are no means of verifying that this reference state is trustworthy, opening the road to preemptive modification attacks.

Recovery mechanisms that circumvent the TPM altogether. Except for TPM reset and preemptive modification, all attacks described above do or may profit from the recovery mechanisms built into BitLocker. These mechanisms pose a particularly attractive target as they yield a key that is independent from the TPM and thus can be used more flexibly. The plausible recovery attack would not even be possible without recovery mechanisms.

Online attacks. Disk encryption does not protect from online attacks [6–8] and is not expected to do so. They must be considered, however, as they offer a straightforward way of finishing the attack once the attacker has obtained the target system along with sufficient secrets to complete the boot process.

Large amount of unprotected disk space. This is a secondary contributing factor to attacks involving purposeful, detectable modification of the platform (plausible recovery; spoofed prompt; tamper and revert). Large amounts of disk space are available for the attacker to install software or data in. This may be difficult to avoid, though.

Confidentiality as the security objective. The security objective of disk encryption also has an impact on attacks. In order to achieve this objective, the attacker has to obtain a small amount of protected data—the key—along with a larger amount of unprotected data—the ciphertext—or a functioning decryption device. This entails a great deal of flexibility on the attacker's part: individual steps of the attack can be executed in almost arbitrary sequence, and there is little the victim can do to restore secrecy once it has been lost. The latter is what Whitten and Tygar call the *barn door property* [12].

	Replace and relay	Plausible Recovery	Spoofed prompt	Tamper and revert	Preemptive modification
Authenticated boot			•	•	
No trusted path to user	•		•	•	
No context awareness	•				
Lack of system authentication			•	•	
History-bounded platform validation				•	
Incomplete diagnostic information		•			
Lack of external reference		•			•
Online attacks	•		•	•	
Recovery mechanisms circumventing TPM	•	•	•	•	
Unprotected disk space		•	•	•	
Confidentiality as security objective	•	•	•	•	•

Table 1. Attack scenarios and contributing factors.

Table 5 shows how these causes and factors contribute to the attacks described before. Each column represents an attack, each line a cause or factor. If a factor contributes to an attack—makes it possible, makes it easier, or makes the result more useful for the attacker—the respective cell is marked with an X. The last two lines contain question marks in all cells: the authors do not fully understand the impact of these factors yet.

6   Proof-of-concept Implementation

We have fully implemented the *tamper and revert* attack described in section 4.3, in order to show its practical feasibility.

The attacker installs the BitLocker PIN Trojan on the victim's computer by starting it from a specially prepared USB drive which contains a stripped down Linux system. This takes less than two minutes and requires no interaction.

The BitLocker PIN Trojan consists of a boot loader installed into the MBR and a second stage that is loaded from an unencrypted NTFS partition which is part of every BitLocker installation. An installation tool is responsible for storing the second stage along with the old MBR as a normal file into a continuous area of the unencrypted NTFS partition. The LBA address of the beginning of the file is written into the MBR.

At the next boot, the MBR boot loader loads this file and transfers control to it. A fake BitLocker prompt is displayed (see figure 1); the entered PIN is stored in the NTFS partition, the original MBR is restored and the system rebooted. Later, the entered PIN can be read from the NTFS partition.

Fig. 1. Spoofed BitLocker PIN-entry screen.

7 Conclusion

We outlined five strategies for targeted attacks against BitLocker, a TPM-supported, software-based disk encryption system. All five strategies require and exploit physical interaction of the attacker with the target computer. While Trusted Computing is expected to help protect against such attacks, our research shows this is not necessarily the case. Using a TPM for key management in a straightforward way provides only very limited protection against a dedicated attacker. None of our attack strategies targets the TPM as such. They all exploit the way it is being used by one particular implementation of disk encryption.

Designers as well as users of disk encryption solutions should be aware of these attack strategies in order to realistically assess how much security they get out of trusted computing. The most important lesson to be learned is that even with Trusted Computing a system needs additional physical protection for good security. This has been known for the *running* system [7]; our research shows that it is true as well if the attacker gains physical access only when the system is powered off. Even if direct physical attacks are excluded from consideration, Trusted Computing does not offer the same level of protection as conventional measures of physical security [13, 14]

Out of the contributing factors discussed above, three seem crucial. First, Trusted Computing as of today is limited to authenticated boot. The technology cannot enforce any policy for a program, malicious or not, that does not require any support from the TPM to run. Second, to successfully attack an encryption scheme one needs to find just one way to obtain a small secret, the key. The TPM helps protect keys but only during part of their life cycle. Third, the user is forced to trust the computer with his secrets, regardless of its state. There is no way for the user to detect skillful tampering and man-in-the-middle attacks.

A proof-of-concept implementation of the tamper and revert attack shows that malicious manipulation of boot code is not only a theoretical issue.

Our work leads to several questions for further research:

- Design standards and evaluation criteria for TPM-supported disk encryption. The point of this paper is not to dismiss Trusted Computing as useless but rather to get a better idea how it should be used in particular applications to achieve the security properties desired.
- More generally, it seems that the exact role of Trusted Computing technology within applications is still unclear. Like any technology, Trusted Computing provides primitives and building blocks that need to be employed and arranged in meaningful ways. We hope that a set of patterns will emerge over time to show developers how to apply Trusted Computing properly.
- Solutions to particular problems, such as establishing trusted channels between the user and the TPM, or providing useful diagnostic and reference information for users and operators to help them in making their security decisions.
- Modeling of attacker capabilities. We have not found a practical method that would have allowed us to model and analyze in a systematic way what

an attacker might do to a system. An easy way of describing a system and analyzing what could or could not be done to it would be helpful.

References

1. Mitchell, C.J. (ed.): Research workshop on future TPM functionality: Final report. `http://www.softeng.ox.ac.uk/etiss/trusted/research/TPM.pdf`
2. Arbaugh, W.A., Farbert, D.J., Smith, J.M.: A secure and reliable bootstrap architecture. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 65–71. IEEE Computer Society (1997)
3. Fergusson, N.: AES-CBC + Elephant diffuser: A disk encryption algorithm for windows vista. Tech. rep., Microsoft (2006)
4. Technet, M.: BitLocker Drive Encryption Technical Overview. `http://technet.microsoft.com/en-us/library/cc732774.aspx` (updated: May 8, 2008)
5. NVlabs: NVbit: Accessing bitlocker volumes from linux. Web page, URL: `http://www.nvlabs.in/node/9` (2008)
6. Hendricks, J., van Doorn, L.: Secure bootstrap is not enough: Shoring up the trusted computing base. In: Proceedings of the Eleventh SIGOPS European Workshop, ACM SIGOPS. ACM Press (2004)
7. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. Tech. rep., Princeton University (2008)
8. Becher, M., Dornseif, M., Klein, C.N.: Firewire: all your memory are belong to us. Slides, URL: `http://md.hudora.de/presentations/#firewire-cansecwest`
9. Templeton, S.J., Levitt, K.: A requires/provides model for computer attacks. In: Proceedings of New Security Paradigms Workshop, pp. 31–38. ACM Press (2000)
10. Sparks, E.R.: Security assessment of trusted platform modules. Tech. rep., Dartmouth College (2007)
11. Sparks, E.R.: TPM reset attack. Web page, URL: `http://www.cs.dartmouth.edu/~pkilab/sparks/`
12. Whitten, A., Tygar, J.D.: Why Johnny can't encrypt. In: Proceedings of the 8th USENIX Security Symposium (1999)
13. Weingart, S.: Physical security devices for computer subsystems: A survey of attacks and defenses. LNCS **1965/2000**, 302–317 (2000). DOI 10.1007/3-540-44499-8_24
14. Weingart, S.: Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses 2008, updated from the ches 2000 version. `http://www.atsec.com/downloads/pdf/phy_sec_dev.pdf` (2008)
15. Drimer, S., Murdoch, S.J.: Keep your enemies close: Distance bounding against smartcard relay attacks. In: USENIX Security 2007 (2007)
16. Tygar, J.D., Yee, B.: Dyad: A system for using physically secure coprocessors. Tech. rep., Proceedings of the Joint Harvard-MIT Workshop on Technological Strategies for the Protection of Intellectual Property in the Network Multimedia Environment (1991)
17. Grawrock, D.: The Intel Safer Computing Initiative: Building Blocks for Trusted Computing. Intel Press (2006)
18. Hargreaves, C., Chivers, H.: Recovery of encryption keys from memory using a linear scan. In: Proceedings of Third International Conference on Availability, Reliability and Security, ARES 08, pp. 1369–1376 (2008). DOI 10.1109/ARES.2008.109