A Systematic Assessment of the Security of Full Disk Encryption

Tilo Müller and Felix C. Freiling

Abstract—Organizations as well as private users frequently report the loss and theft of mobile devices such as laptops and smartphones. The threat of data exposure in such scenarios can be mitigated by protection mechanisms based on encryption. *Full disk encryption* (FDE) is an effective method to protect data against unauthorized access. FDE can generally be classified into *software-* and *hardware-based* solutions. We assess the practical security that users can expect from these FDE solutions regarding physical access threats. We assume that strong cryptography like AES cannot be broken but focus on vulnerabilities arising from practical FDE implementations. We present the results of a comprehensive and systematic comparison of the security of software- and hardware-based FDE. Thereby, we exhibit attacks on widespread FDE standards in many common scenarios and different system configurations. As a result, we show that neither software- nor hardware-based FDE provides perfect security, nor is one clearly superior to the other.

Index Terms—Full disk encryption, self-encrypting drives, physical access threats, cold boot/DMA/evil maid/hot plug attacks

1 Introduction

Many people store private data like emails, chat histories, contact lists, access credentials, and calendar entries on mobile devices such as laptops and smartphones. Above that, employees of governmental institutions, commercial enterprises and healthcare facilities often store highly sensitive information on their devices. This information is consistently at risk to get *physically* lost or stolen, e.g., at public places like airports. Setting user passwords and screen locks on such devices is not sufficient to protect data against unauthorized access, because adversaries can simply bypass the OS software of a device and access the data storage directly. Only the use of strong cryptography can guarantee data confidentiality in the case that storage media are directly accessed.

1.1 Full Disk Encryption (FDE)

Full disk encryption characterizes the process of encrypting entire volumes, meaning whole drives or partitions. As illustrated in Fig. 1, disk encryption can either be performed inside the operating system, i.e., on kernel-level (*software-based FDE*), or inside a hard drive, i.e., on firmware-level (*hardware-based FDE*). Historically, FDE became widely available to users through software-based solutions like BitLocker (Windows), FileVault (Mac), and dm-crypt (Linux/Android). Only later, vendors of solid state drives (SSDs), such as Intel's SSD 320 and 520 series, began to ship their products with hardware-based FDE. These drives are commonly referred to as *self-encrypting drives* (SEDs), because they perform encryption inside the disk drive controller, such that encryption keys are not present in a computer's main memory or CPU.

SEDs provide full disk encryption including the master boot record (MBR). With a software-based solution, the MBR is necessarily present in unencrypted form for bootstrapping. Briefly spoken, software-based approaches allow MBR manipulation attacks such as *evil maid*, as explained later. Furthermore, software-based solutions store encryption keys in memory, enabling attacks on RAM in the form of *cold boot* and *direct memory access (DMA)-based attacks*, as also explained later. In contrast, with SEDs encryption keys do not enter RAM or the CPU permanently, removing these entities as potential attack vectors. So simply by comparing known attack vectors, it seems that SEDs are superior to software-based FDE. However, it is not entirely clear which approach offers better protection.

1.2 Threat Model

Because of the different attack vectors, it is hard to compare hardware- and software-based FDE. While the attacks may differ, the threat model however is the same. We assume that an attacker wishes *access to the data of an encrypted hard drive* by means of *physical access*. FDE does not protect data when a user logs into a system and leaves it unattended, nor does it protect data against system subversion through (remotely injected) malware. Disk encryption protects data in the case that a drive is lost, stolen, or seized. However, also in these scenarios many practical attacks against disk encryption exist. Briefly spoken, such attacks usually require the target to be locked but running or to be in standby mode. In this article, the main example for our investigations therefore are laptops which are in standby mode, and smartphones which are locked but running. Note, however, that servers and desktops are threatened as well when they are found to be running (or to be in standby mode). To be conservative in our security evaluation, we assume that strong cryptography like AES cannot be

• *The authors are with theDepartment of Computer Science, Friedrich-Alexander-University Erlangen-Nuremberg, Germany. E-mail: {tilo.mueller, felix.freiling}@cs.fau.de.*

Manuscript received 13 Sept. 2013; revised 20 Oct. 2014; accepted 27 Oct. 2014. Date of publication 9 Nov. 2014; date of current version 16 Sept. 2015. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TDSC.2014.2369041

broken. Our focus lies on vulnerabilities arising from practical FDE implementations.

1.3 Overview and Contributions

In this article, we assess the security that users can expect from FDE regarding physical access threats. We present the results of a comprehensive and systematic comparison of software- and hardware-based FDE solutions. While many attacks against software-based FDE on PCs have already been known for years, we exhibit novel attacks against smartphones and hardware-based FDE. We show that for each attack on software-based FDE there exists a "corresponding" attack on hardware-based FDE. As a result, we show that neither software- nor hardware-based FDE provides perfect security, nor is one clearly superior to the other. More precisely, our contributions are:

1) *Software-based FDE security survey*. We present an overview about known attacks against software-based FDE in a chronological order, including *DMA-based attacks* [1], *cold boot attacks* [2] and *evil maid attacks* [3].
2) *Hardware-based FDE security*. We systematically investigate if, and to which extent, SEDs are possibly affected by similar attacks as software-based FDE. We show that, depending on the hardware configuration of a system, there exists a new class of attacks that is specific to self-encrypting drives, namely *hot plug attacks* [4]. Additionally, we study the applicability of other attacks and come to the conclusion that hardware-based FDE is generally as insecure as software-based FDE.
3) *FDE security experiments*. We illustrate our findings by giving details of a comprehensive evaluation of old and new attacks on FDE. With FROST [5], we show that Android's FDE is vulnerable to *cold boot attacks*, similar to PC-based solutions. Moreover, we study the practicability of attacks against hardware-based FDE. This study is based on experiments with twelve different computer systems, including eight laptops and four desktop machines. Overall, only a few hardware-based FDE systems withstood more attacks than software-based FDE.
4) *Forensic guidelines to thwart FDE*. We provide guidelines for forensic examiners, e.g., from law enforcement, that describe best practices to thwart full disk encryption. Our guide considers both software-based and hardware-based FDE, as well as different construction types like servers, workstations, laptops, and smartphones.

1.4 Roadmap

In Section 2, we first give necessary background on software- and hardware-based FDE and formalize the different states of a system in the Advanced Configuration and Power Interface (ACPI) standard. In Section 3, we survey the attack landscape of software-based FDE, including *DMA-based attacks* [1], *cold boot attacks* [2] and *evil maid attacks* [3]. While this can be also regarded as background, it sets the stage for new and corresponding attacks on hardware-based FDE, presented in Section 4. In Section 4, we systematically investigate if, and to which extent, SEDs are possibly affected by similar attacks as software-based FDE. We also introduce the class of *hot plug attacks* [4] in that section.

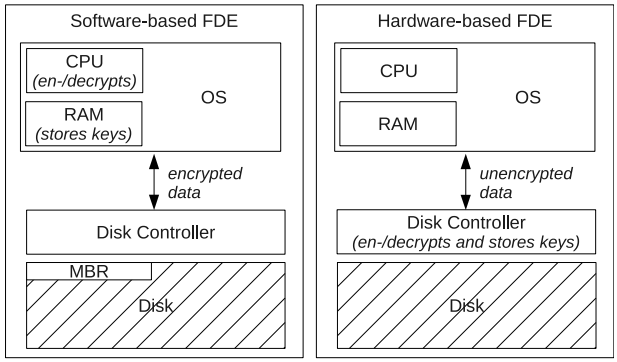

Fig. 1. Software-based FDE compared to hardware-based FDE. The shaded areas represent encrypted data.

We illustrate our findings with a detailed security analysis of Android's FDE and common desktop and laptop hardware in Section 5. In Section 6, we give forensic guidelines that describe best practices on thwarting full disk encryption. Finally, in Section 7, we conclude with a list of possible countermeasures.

2 Background

To reduce the risk of data exposure when a storage drive is physically lost, encryption is widely accepted as a strong and convenient countermeasure. Disk encryption can either be performed inside the operating system on kernel-level, or inside the drive itself on firmware-level, as shown in Fig. 1. We first give an overview about kernel-level encryption and then about firmware-level encryption. As we show, disks are protected against unauthorized access only if a machine is *powered off*. In other system states (*running* and *sleep* modes) FDE is often vulnerable. Therefore, we also give an overview about ACPI power states.

2.1 Software-Based Full Disk Encryption

Software-based encryption has long existed in the form of file and container encryption, but common solutions for *full disk encryption* became available only in the early 2000s. The history of FDE begins with Linux' cryptoloop, the predecessor of dm-crypt, which was published in 2003. In the same year, Apple released its initial version of FileVault, but this version could only be applied to a user's home partition. In 2004, the popular cross-platform utility TrueCrypt was first released, and in 2006, Microsoft's BitLocker for Windows was published. Finally, in 2011, Apple released an updated version of FileVault, which can also encrypt the system partition.

All software-based FDE solutions have in common that they run encryption inside the CPU and that they store necessary keys in main memory. Besides that, support for the symmetric block cipher AES (*Advanced Encryption Standard*) as the underlying cryptographic primitive seems to be a de-facto standard as of 2014. To illustrate this, in the following we have a look at the

case of IBM-compatible PCs running Windows as well as Android-driven ARM devices.

2.1.1 FDE Support under Windows

A common example of an FDE solution under Windows is Microsoft BitLocker. BitLocker is available in the Enterprise and Ultimate editions of Windows versions from Vista to Windows 8 and relies on the AES algorithm with 128- or 256-bit secret keys applied in cipher block chaining mode. With BitLocker, almost all data on a disk is encrypted, including swap files, hibernation files, unallocated space, the OS itself, and user files. For bootstrapping reasons, however, every software-based disk encryption requires a small part of the disk to remain unencrypted as pre-boot environment. If a correct password is entered, the decryption key can be derived and the system partition is decrypted. Only then, control is passed on to the (then unencrypted) Windows OS. From there on, BitLocker's encryption is handled by a device driver inside the Windows kernel.

TrueCrypt, which is a free FDE solution and largely open source, basically pursues the same two-staged boot sequence comprising (1) an unencrypted pre-boot environment, and (2) a kernel-level device driver. Apart from that, TrueCrypt is basically a comprehensive tool set. For instance, besides FDE it can be used for file- and folder-level encryption, and it supports more cryptographic parameters than just AES and CBC.

2.1.2 FDE Support on Android Smartphones

As of 2014, there is only one example for FDE under Android, namely the official encryption solution by Google (available since Android 4.0). While apps that extend the functionality of Android can be written by third parties and are provided via app stores, the disk encryption feature requires system privileges and resides entirely in system space. Google's official solution is based on dm-crypt, a known FDE solution for Linux. Although dm-crypt can basically encrypt entire disks, Android only encrypts user partitions mounted at `/data`.

Android's FDE can be activated if PIN-locks or passwords are in use. In Android, PINs consist of 4 to 16 numeric characters, and passwords consist of 4 to 16 alphanumeric characters with at least one letter. Other screen locking mechanisms like pattern-locks and face recognition are less secure and not possible in combination with FDE. Pattern-locks, for example, can be broken by *Smudge Attacks* [6], and face recognition can simply be tricked by showing a photo of the smartphone owner [7].

2.2 Hardware-Based Full Disk Encryption

The advantage of solid-state drives compared to previous hard disks is their robustness and performance impact that stems from built-in NAND flash memory. Additionally, many SSD vendors offer another feature on the consumer market, namely drive-level encryption. Although drive-level encryption was available before the rise of SSDs, we focus on Intel's SSD models from 2012, in particular the SSDs 320 and 520. These SSDs provide built-in encryption based on Advanced technology attachment (ATA) security.

2.2.1 ATA Security Lock

Advanced technology attachment [8] is the standard interface for connecting mass storage devices since the 1990s. The ATA security feature was standardized in ATA-3 in 1997. As of 2014, serial ATA (SATA), standardized 2003 in ATA-7, is the prevalent bus for connecting SSDs. ATA drives have the ability to get locked and then remain inaccessible until the correct password is entered. ATA security defines two kinds of passwords, *user* and *master*, as well as two security levels, *high* and *maximum*. On high security level, the user and master password can be used interchangeably for unlocking a drive. On maximum level, the master password can only be used to securely erase the drive and to reset the user password, but not to read data. In other words, the master password enables a company or vendor to reset disks in the case of password loss.

ATA passwords provide only little more security than BIOS and OS passwords, because an ATA password does *not* imply that the drive is encrypted. To remove ATA passwords, attackers can bypass the drive controller and access storage memory directly. Usually, this requires specialized hardware, but some models even allow for software-only attacks by accessing the firmware of a drive [9]. To overcome these threats, and to complete the physical security layer, ATA passwords must be paired with drive-level encryption.

2.2.2 Self-Encrypting Drives

Many SSDs have built-in encryption facilities based on AES and are therefore called *self-encrypting disks* or *drives*. Each SED has a unique encryption key which is generated from entropy sources inside the drive, such that it is not known to the manufacturer. This key is called the *media encryption key* (MEK). The MEK is used to encrypt the user data and is encrypted by means of the *key encryption key* (KEK). The KEK is derived from the user password; disks are powered up locked until the correct password is entered. Re-encryption is avoided because only the MEK must be newly encrypted with the KEK when a password is changed.

Interestingly, if no password is set, a drive is unlocked and can freely be accessed, but still encrypts data by means of the MEK. That is, encryption works out-of-the-box for SEDs, but to activate any degree of protection, drive passwords must be set. These passwords can be set via ATA commands, as described above, from any compatible BIOS.

2.3 ACPI System Level States

The *Advanced Configuration and Power Interface* is a long-term standard for PCs that puts power management of hardware devices under control of the operating system. Once ACPI is activated, the OS is required to control all power events. For example, when a laptop lid is closed, an interrupt induces the OS to put the laptop into sleep mode. ACPI was initially released in 1996 by Intel, Microsoft, and Toshiba as a replacement for the older *Advanced Power Management* (APM).

ACPI defines six system *sleep states*, denoted Sx for $x \in \{0, \ldots, 5\}$. The Sx sleep states are important as they directly affect all physical access attacks. Depending on the Sx state, an SED might be locked or unlocked. Accordingly,

keys of software-based FDE might be in RAM or not. Keys are usually available in RAM during S0 to S3; only in S5 they are definitely lost.

- S0: Non-sleep mode, i.e., the machine is up and *running*. SEDs must be unlocked during S0, and the key of software-based FDE must be in RAM.
- S1/S2: The CPU stops working, but is still powered on. Users do not experience a difference to S0, and thus we consider these states as *running*, too. SEDs are unlocked in S1/S2, and software-encryption keys are available in RAM.
- S3: Known as *suspend-to-RAM* because the CPU is powered off and its context is swapped out into RAM. SEDs are powered off and locked during S3, but in almost all cases they get automatically unlocked upon wakeup. Furthermore, software-based FDE does not require users to reenter keys upon wakeup, but keeps the keys in RAM during S3.
- S4: Referred to as *suspend-to-disk* because CPUs and RAM are powered off and the system state is swapped out to non-volatile memory of the disk (which is then powered off as well). SEDs get locked during S4, and the ATA password is always required to unlock SEDs upon wakeup. Software-based FDE usually loses the key during S4.
- S5: The machine is *powered off* and no system state is preserved. A full boot into S0 is required on power-up. SEDs get locked in S5, and ATA passwords are needed to unlock them. Keys for software-based FDE must be reentered on power-up.

3 Software-Based FDE Security

We now give a comprehensive overview of all known attacks on software-based FDE. These attacks serve as inspiration to new attacks on hardware-based FDE explained in Section 4. Overall, we identified three types of physical access attacks against software-based FDE, that we present in chronological order.

3.1 DMA-Based Attacks

In 2005, *direct memory access* attacks were pioneered by Dornseifet al. [1]. Their work was the beginning of a series of attacks that exploit DMA interfaces like FireWire. In principle, all DMA-capable ports exhibit the same vulnerability, including ExpressCard [10], [11], PCI Express (PCIe) [12], and, as shown in 2012, Thunderbolt [13]. In the original attack, an Apple Macintosh was compromised via direct memory access from a malicious iPod. However, in the original attack, disk encryption was not considered explicitly. One possibility to deploy DMA attacks against disk encryption is to target the key of software-based FDE in main memory. Using DMA, main memory can be scanned for possible keys and these keys can later be used to decrypt the disk. Another attack stems from the fact that DMA allows to write into RAM and to manipulate the system space. This can be exploited to unlock an OS lock screen, as proven by attacks against Windows Vista [14] and Windows 7 [15].

DMA-based attacks succeed if the target PC is running (S0–S2) or in standby mode (S3). If the target is switched off, the disk encryption key is not present in RAM but must be reentered upon boot. If the target PC is in S3, it can be woken up without reentering the key in all practical implementations we tested. The academic disk encryption solution TRESOR [16] (*TRESOR Runs Encryption Securely Outside RAM*) changes this behavior and asks users to reenter their passwords upon wakeup. However, TRESOR does not protect *running* machines against DMA attacks [17]. To protect running machines, DMA attacks must be counteracted on the hardware layer. With the virtualization technology for directed I/O, DMA transfers can be filtered by means of the IOMMU. The IOMMU prevents DMA devices from accessing certain memory regions like the system space. TreVisor [18] (*TRESOR HyperVisor*), for example, makes use of the IOMMU to filter accesses to the hypervisor space. Until to date, however, this feature is not supported by widely used OSs including Windows up to version 8.

3.2 Cold Boot Attacks

In 2008, Halderman et al. [2] broke several FDE solutions with a technique known as the cold boot. The basic idea of this attack is to retrieve keys from RAM after rebooting a system with a mini OS from a USB drive. RAM can generally be read out after a reboot, because in contrast to common belief, memory contents do not disappear immediately. Quite the contrary, memory contents fade away gradually over time after power is cut. In practice, it can take as long as 30 seconds for memory contents to fade away completely.

The underlying effect of cold boot attacks is called the *remanence effect* [19], [20]. One aspect of this effect is that low temperatures slow down the fading process, such that by cooling down RAM chips, the remanence interval can be extended from 30 seconds up to ten minutes. Anderson and Kuhn first outlined attacks that exploit the remanence effect of cooled down RAM chips [21]. In applied cryptography, the remanence effect can also be used as a time source [22] and as an entropy source [23]. With respect to disk encryption, another way to access keys in main memory is to replug cooled down RAM chips *physically* into another PC. This variant is more generic than the reboot variant, because it works irrespectively of BIOS and boot sequence settings.

Cold boot attacks pose a generic threat to all software-based FDE technologies, including dm-crypt, BitLocker and TrueCrypt. Similar to DMA-based attacks, the target of a cold boot attack must be running or in standby mode. In academia, so-called *CPU-bound encryption systems* like AESSE [24], TRESOR [16], LoopAmnesia [25] and ARMORED [26] are known to protect against cold boot attacks. CPU-bound encryption systems never store keys in RAM but only inside CPU registers over the entire uptime of a system. The protection of CPU-bound encryption, however, is limited to classic cold boot scenarios. Other attacks, such as DMA and JTAG access on running machines, cannot be defeated by CPU-bound encryption, as shown by Blass and Robertson [17].

3.3 Evil Maid Attacks

The term "evil maid" was coined by Rutkowska [3] and is based on the following scenario: Let the victim be a traveling salesman who leaves his encrypted laptop in a hotel

room and goes out for dinner. An evil maid can gain physical access to her target unsuspiciously, and so she can replace the MBR with a modified version that additionally performs keystroke logging. Later on, the unaware salesman boots up his machine and enters the password as usual. On her next access to the machine, the evil maid reads out the logged password.

In comparison to the other threats, evil maid attacks do not require the target system to be running or in standby mode, but they are also effective against switched off systems. The original evil maid attack was implemented against TrueCrypt in 2009. Earlier that year, another bootkit known as the *Stoned Bootkit* [27] circumvented TrueCrypt, too. As of 2013, TrueCrypt is still vulnerable to these attacks and no future improvements are planned. To the contrary, the authors argue that bootkits "require the attacker to have [...] physical access to the computer, and the attacker needs you to use the computer after such an access. However, if any of these conditions is met, it is actually impossible to secure the computer." [28]

As opposed to this, Microsoft's BitLocker defeats software keyloggers up to a certain degree as it measures the integrity of the boot process by means of the *Trusted Platform Module* (TPM). TPMs are used to build trusted checksums over sensitive boot parameters, such as the firmware, the BIOS, and the bootloader. BitLocker's decryption key can only be derived if these checksums are in line with the reference configuration from system setup. Otherwise, users cannot decrypt their data and hopefully become suspicious that somebody manipulated their machines. Nevertheless, in 2009, even BitLocker was successfully compromised by bootkit attacks. Türpe et al. [29] practically performed *tamper-and-revert* attacks that (1) tamper with the bootloader to introduce keylogging, (2) let the victim enter the password into a forged text-mode prompt, (3) revert to the original bootloader, and (4) reboot. The victim may wonder about the reboot, but most likely the victim enters the password again and proceeds as usual—unaware of the fact that the password has already been logged.

In theory, tamper-and-revert attacks can be defeated by mutual authentication schemes like STARK [30] (*STARK Tamperproof Authentication to Resist Keylogging*), which prove the integrity of a PC towards the user before a password is entered.

4 Hardware-Based FDE Security

Hardware-based full disk encryption drives are widely believed to be a fast and secure alternative to software-based solutions like TrueCrypt and BitLocker. Since encryption keys are stored inside a crypto chip of the disk drive, rather than in RAM or inside the CPU, traditional attacks like cold boot seemingly fail. However, the actual security given by hardware-based FDE has not been investigated in the literature yet.

In 2012, it was demonstrated that hardware encryption saves energy in comparison to software-based FDE [31]. With respect to security we show that, depending on the configuration of a system, hardware-based FDE is generally *as insecure* as software-based FDE. The main reason is a class of surprisingly simple attacks that exploit the fact that self-encrypting drives do not notice whether SATA cables are replugged. We also show how to adapt known attacks from software-based FDE.

4.1 Hot Plug Attacks

We now introduce hot plug attacks [4], a threat that is specific to hardware-based FDE; software-based solutions are not affected. We first describe the general concept of these attacks and put them into context. However, we argue why similar attacks fail, underlining the advantage of the simplicity of hot plug attacks.

4.1.1 Attack Concept

Roughly speaking, the technique is similar to a variant of the cold boot attack. In the generic variant of the cold boot attack, RAM chips are removed from running PCs and then replugged into another PC in order to be read out. As it is pointless to replug RAM chips of a system that employs hardware-based FDE, the entire disk must be replugged instead. This measure turns out to be an effective and practical attack against all SEDs.

The underlying implementation flaw in SEDs is the fact that SEDs do not detect whether SATA cables are unplugged, as long as they stay connected to power. An SED gets locked only if its power connection is cut. If its data connection is cut, it stays unlocked. This leads to the following attack scenario: With physical access to a running PC, the SATA connector of an SED is unplugged and then replugged to a PC that is under control of the attacker. During this procedure, the original PC acts as energy supplier and keeps the disk powered on. The second PC, which is under control of the attacker, acts as data collector. The original PC crashes a few seconds after disk removal (e.g., with a blue screen) but the SED stays unlocked. If the attacker's PC supports SATA hot-plugging, SEDs can directly be read out and full access to the data is possible without the need to know the password.

This attack is called the *hot plug attack*. Note that the hot plugging feature we exploit is an intended feature of the SATA standard for multi-drive applications like RAID. Similar to attacks against software-based solutions such as cold boot, hot plug attacks require the system to be running. On running desktop and server systems, hot plugging is even advantageous over cold boot because it works irrespectively of BIOS settings like "PW on reboot" and the "boot device order".

Since server systems are commonly running 24/7, they constitute a perfect target for hot plug attacks. This can, for example, be relevant for law enforcement executing a search warrant against server clusters. Another advantage of desktop and server systems is their construction type: Most often, SEDs are connected to the motherboard with flexible SATA cables that can be replugged into nearby machines easily. However, the case of laptops is considered more critical, because laptops are at risk to get lost or stolen while travelling. Unlike servers, which can be protected by means of physical security measures, laptops fall into the wrong hands easily. In laptops, the power and data connection of an SED cannot be handled independently as in most desktops. To the contrary, the power and SATA interfaces inside

a laptop are connected directly to the board without cables in between. Another difficulty with laptops is that they are often carried along in standby mode and only get switched on before usage. In other words, it is more likely that a laptop is lost or stolen when it is in standby mode than when it is running.

The problem is that in suspend-to-RAM mode (S3), an SED gets switched off and is consequently locked. Hence, the question is how can hot plug attacks be deployed against laptops that are suspended to RAM? Rather surprisingly, this is often possible although the disk is switched off and locked. Laptops require an ATA password on boot, but they unlock the disk *automatically* on wakeup from S3. This behavior leads to the following attack: When a disk is switched off and locked, it can easily be removed out of the laptop chassis. So the disk is removed during S3 and SATA and power extension cables are installed between the drive and the board. Afterwards, the laptop is woken up and, regardless of the extension cables that were installed, the SED gets unlocked. The remaining attack is in analogy to the desktop case, i.e., the SATA cable is replugged into a second machine that accesses the data. As a result, laptops in S3 mode can often be attacked with hot plug attacks like desktop systems. Moreover, since attackers can put a running laptop into sleep mode, e.g., by closing the lid, running laptops are equally affected.

4.1.2 Related Attacks

The problem of missing SATA cables inside laptops (and also inside a few desktop PCs) is a mechanical problem. If the SATA connection between SEDs and motherboards can be disconnected without removing power, similar attacks succeed against laptops. Another way to overcome the problem would be given if an attacker can replug power connectors so quickly that an SED does not get locked. If we could do so, we could pull SEDs out of running laptops and connect them to external power supplies. So by analogy to the cold boot attack, one might conjecture that SEDs stay unlocked if power is cut only briefly, because SEDs hold keys inside RAM chips internally. We unplugged power cables and replugged them within less than a second several times during our experiments, and we even cooled down the SEDs, but we did not succeed.

Another interesting point is that SEDs get unlocked automatically upon wakeup from S3. SEDs can only get unlocked with the correct password. This implies that the ATA password must be present in RAM or NVRAM during S3, because other components are not energized. Furthermore, this implies that the password must be present in RAM or NVRAM during the entire uptime, because the ATA password cannot be regained from a disk before entering standby mode. As a consequence, the password must be retrievable from a computer's memory, similar to encryption keys of software-based solutions. We tried to locate the password in RAM images that we acquired via cold boot attacks, and we also acquired NVRAM images with dedicated BIOS interrupts, but we could not find the password. An explanation might be that ATA passwords are stored inside a protected NVRAM region that is not easily accessible via BIOS interrupts. We could only access the first 256 bytes of NVRAM, but we conjecture that the real NVRAM size is greater and that the upper bytes are protected and designed to store sensitive information like passwords.

4.1.3 Countermeasures

BIOS vendors could take action to secure hardware-based FDE against hot-plug attacks. For example, SATA passwords must not be stored in RAM or NVRAM during S3. That is, when an SED loses power, it must never get unlocked automatically, but the user must always be prompted to reenter his or her password upon wakeup. As explained in Section 5.2.4, in our experiments only *one out of twelve* tested systems implemented S3 like this.

Additionally, to defeat hot plug attacks against running PCs, hardware-based locks must be introduced that are sensitive to the SATA connectivity. SEDs get locked only when power is cut, but they do not get locked when the SATA data cable is cut. As shown in Section 5.2, *none* of our test systems was able to detect a SATA cable unplugging event.

4.2 Adaptation of Known Attacks

Given the simplicity and effectiveness of hot plug attacks, we now ask whether more attacks against SEDs succeed. As a starting point, we chose well-known attacks from software-based FDE. That is, we systematically expose hardware-based FDE to the same threats which are known to be effective against software-based solutions. We argue that for all physical access attacks on software-based FDE, either the very same attack works on hardware-based FDE, or there exists an adaptation that succeeds under similar conditions.

4.2.1 Evil Maid Attacks on SEDs

In software-based FDE, the master boot record of an encrypted hard disk can be manipulated because it must be present unencrypted for bootstrapping. We argue that this attack can be adapted to SEDs even though their MBRs are encrypted: As in the original evil maid scenario, let the victim be a traveling salesman who leaves his hotel room to go out for dinner. The evil maid breaks into his room, removes the target SED, steals it, and *replaces* it with her own drive before she leaves the room. Later on, the unaware salesman boots up his machine, and a one-to-one copy of his familiar password prompt is displayed. He enters the password, since the forgery can visually not be noticed, and the password is then sent to the evil maid over a network connection.

Of course, the salesman becomes suspicious when the OS and his user data cannot be decrypted, but then it is too late because the evil maid already owns the SED as well as the password. To leave no traces about the connection, or the attack in general, the replaced drive wipes itself after transmission of the password. If an identical drive model is used, the salesman may not even recognize that the original drive is gone and may believe in a hardware failure. Compared to traditional evil maid attacks, this variant requires only *one* physical access—given the fact that a network is in range. But with the increasing availability of wireless networks there is almost no limitation. The attack demonstrates that the boot process of a PC can be manipulated in a variety of ways, even though MBRs are encrypted.

4.2.2 DMA Attacks on SEDs

One possibility to deploy DMA attacks against software-based FDE is to target necessary keys that are kept in main memory. Using direct memory access, RAM can be scanned for keys and these keys can be used to decrypt the disk. Another possibility to deploy DMA attacks stems from the fact that attackers can *write* into memory to manipulate the system space and to unlock the screen.

When attacking SEDs, the first variant of DMA attacks (accessing keys in RAM) clearly fails because the encryption key is not present in RAM. But the second variant (unlocking the screen) works the same way. The basic assumption about the target is that the target must be running or in standby mode. As explained above, SEDs are unlocked automatically upon wakeup from S3, and therefore, DMA attacks are effective against suspend-to-RAM, too. We successfully reproduced DMA attacks over FireWire against systems with self-encrypting drives. Our targets were up-to-date Windows 7 machines with password protected SEDs. On the attacking side, we used *Inception* [32] to unlock the screens of our targets, a physical memory manipulation tool that can unlock Windows, Mac OS X, and some Linux distributions. The unlocking process takes only a few minutes and afterwards any password is accepted.

4.2.3 Cold Boot Attacks on SEDs

Rebooting a target system in order to acquire its RAM contents is useless against SEDs because the key is not in RAM. But most of our test machines (see Section 5.2) did not ask for ATA passwords on reboot, and on such machines "cold boot" attacks become trivial. Only a few laptops provide a BIOS setting called "ATA password on reboot" that can be activated. Many laptops and all desktop machines in our test set did not provide this setting. Hence, the vulnerability is not only a configuration problem but the settings *are missing*. We were able to reboot vulnerable systems with external drives to start live systems. From the live system, we were able to mount partitions of the (still unlocked) SED, and so we could read out data without knowing the password. This attack is simple, even simpler than the hot plug attack, but it effectively breaks the encryption of all SEDs.

Intel states about its SEDs that "the drive password is required each time the drive is powered on", meaning that SEDs get locked each time they are powered off. Hence, an SED does not get locked on reboot because it fails to realize that the system actually reboots as long as it stays connected to power. We ascertained that hardware resets and software reboots are equally effective (because hardware resets might cut power briefly). Since physical reset buttons are often not available, we induced hardware resets by connecting two pins on the motherboard. If BIOS settings disallowed us to boot from external devices, we inserted a second HDD where that was possible (i.e., in desktops, not in laptops) and booted from that.

On those few test systems where an SED gets locked during reboot, we conjecture that it gets explicitly locked in software. But software-locking is *not* an effective method to prevent reboot attacks, because we were always able to circumvent this mechanism. After inducing a reboot, we unplugged the data cable briefly before shutdown, and replugged it immediately after the machine came up again. This attack requires precise timing, but it prevents the disk from getting locked and enables us to access data without knowing the password. The reason is that an SED does not get locked when it is not present at the time that the ATA lock command is sent. We believe that this is a general flaw because we were able to attack different laptops from different vendors with the same approach. A countermeasure against this attack would be to cut power to SEDs during reboot. None of our 12 test systems, however, did cycle power off during reboot.

5 Evaluation

We practically applied all attacks we identified in the previous sections to two targets: (1) Android-driven smartphones (particularly, the Samsung Galaxy Nexus) and (2) self-encrypting drives (particularly, Intel's SSD 320 and 520). In the following sections, we give technical information on how we performed these attacks and present our evaluation results.

5.1 Attacks on Encrypted Android Smartphones

Smartphones constitute a perfect target for cold boot attacks because they are frequently lost while *running* and get powered off only seldomly. To prove that cold boot attacks against ARM devices are possible, we developed FROST (*Forensic Recovery of Scrambled Telephones*) [5]. FROST is a tool that can be installed into the recovery partition of a smartphone after an attacker got physical access to it. It is not necessary to have FROST installed on a device before the attack, but it is necessary that the bootloader of the device is *unlocked*. On smartphones where the bootloader is unlocked, FROST can be deployed to recover encryption keys from RAM and to decrypt the user partition. We have developed FROST for Galaxy Nexus devices from Samsung as an example. On devices where the bootloader is locked, it must get unlocked first. Galaxy Nexus devices can always get unlocked with physical access but the unlocking process wipes the encrypted user partition. Consequently, it becomes pointless to retrieve the encryption key from RAM (although this is still possible). In that case, FROST can be deployed to acquire complete RAM dumps in order to search for personal data directly.

5.1.1 Implementation of the FROST Recovery Image

We now present the implementation of FROST. Technically, FROST is a set of tools that are bundled together in a recovery image based on Cyanogenmod. The recovery image displays a graphical user interface which allows IT practitioners to recover encryption keys and to unlock encrypted partitions with only a few clicks. We also implemented an option for brute force attacks against weak PINs and implemented an option that saves full memory dumps to a PC.

The centerpiece of FROST is a loadable Linux kernel module (LKM). The FROST LKM is based on the x86 utility aeskeyfind [33] that searches for AES keys in a given memory image. Aeskeyfind iterates through each byte of a memory image and treats the following sequence as a potential AES key schedule. If the total number of bits violating a

	ε	$0.5-1s$	$1-2s$	$3-4s$	$5-6s$
$5-10$ °C	0 (0%)	2 (0%)	1911 (5%)	8327 (25%)	24181 (73%)
$10-15$ °C	0 (0%)	976 (2%)	2792 (8%)	18083 (55%)	25041 (76%)
$15-20$ °C	0 (0%)	497 (1%)	4575 (13%)	20095 (61%)	25433 (77%)
$20-25$ °C	0 (0%)	421 (1%)	16461 (50%)	23983 (73%)	27845 (84%)
$25-30$ °C	1 (0%)	2204 (6%)	16177 (49%)	27454 (83%)	28661 (87%)

Fig. 2. Number of bit flipping errors per page (in total and percentage) dependent on the phone temperature and the time of battery removal. It is possible to retrieve perfect RAM images from Galaxy Nexus devices by cooling the devices down to less than $10°$C and replugging the battery in less than 500 ms.

correct AES key schedule is smaller than a certain threshold, the key is printed out [2]. Contrary to the original version of aeskeyfind, FROST is implemented for ARM and searches for AES keys *on-the-fly*, i.e., directly on the phone.

The FROST LKM basically supports two search modes: *quick search* and *full search*. Quick search is highly optimized for Galaxy Nexus devices and looks for AES keys at certain RAM addresses. In quick search mode, the recovery process finishes within seconds but it might fail on other devices because the search space might be too specific. Therefore, we implemented a full search mode which considers the entire physical memory space. The full search mode uses a sliding window mechanism that looks at each physical RAM page twice. In quick search mode, AES key schedules which are spread over multiple pages might be missed.

If neither the quick search mode nor the full search mode succeeds, the memory image is too noisy, meaning that too many bit flipping errors occurred during cold boot (see Section 5.1.3). Aeskeyfind discards key candidates as soon as a given threshold of bits is reached that is not in line with a typical key schedule structure. If this threshold is too high, pseudo keys are identified from irrelevant memory regions. If the threshold is too low, the recovery algorithm becomes prone to bit flipping errors. As a tradeoff value, which is based on results of our experiments, we have chosen 64 as threshold. That means, 64 of 1,280 bits can be disturbed per key schedule at maximum, or, in other words, the bit error ratio is not allowed to exceed 5 percent.

The FROST recovery image allows to decrypt and mount partitions directly on the phone with previously recovered keys. To decrypt the user data partition, we integrated a statically linked ARM binary of the *dmsetup* utility [34]. This option becomes available only if one of the key recovery methods succeeds, i.e., after the decryption key is known.

5.1.2 Application of the FROST Recovery Image

We now give details regarding the practical application of FROST. First, the device must be *cooled down* to increase the success rate of cold boot attacks. As a rule of thumb, we have positive experience with putting the device into a -15 °C freezer for 60 minutes. Note that the device should be packed up in a freezer bag to protect it against water condensation.

After the phone has been cooled down, we can begin with the cold boot attack. We assume the bootloader of our target is unlocked. Since a Galaxy Nexus device has no reset button, we have to reboot it by unplugging the battery briefly. To install the FROST image onto the phone after reboot, the buttons *volume up* and *volume down* must be held during boot. This combination puts the phone into the *fastboot* mode. Once the phone is in fastboot mode, it must be connected to a PC via USB. FROST can now be installed to the device with the fastboot command line utility. Afterwards, the *recovery mode* option must be selected from the phone's boot menu in order to bring up the FROST GUI. Eventually, keys can be recovered with FROST as described above.

5.1.3 Impact of the RAM Operating Temperature

In the following, we give exact benchmarks for the remanence effect of Galaxy Nexus devices. Fig. 2 lists the bit error rate of memory pages as a function of the device temperature and the time without power before reboot. To determine the device temperature, we utilized an infrared thermometer and pointed it to the exactly same position on the phone's motherboard each test run. To cool down the phone, we put it into a-$15°$C freezer. 25-$30°$C is the normal operating temperature of a Galaxy Nexus, 20-$25°$C is reached after 10 minutes, 15-$20°$C after 20 minutes, 10-$15°$C after 40 minutes, and 5-$10°$C after 60 minutes inside the freezer. To determine the bit error rate, we filled pages at fixed physical addresses entirely with 0xff. After rebooting the device, we reconsidered the pages that we filled and counted the bits that were now zero.

In this way, we were able to calculate the total number of decayed bits and to estimate the overall bit error rate, as listed in Fig. 2. Note that the highest possible bit error rate is 87.5 percent, and not 100 percent, because the passive state of 50 percent of RAM lines is 0xc0. The most inaccurate measures of our experiments were the times that a device was without power. The problem of rebooting Galaxy Nexus devices is that battery removal is a manual, mechanical task and milliseconds are crucial for the success of cold boot attacks. With ε we define the quickest unplugging/replugging procedure that we were able to perform; we claim this was consistently below 500 ms.

5.1.4 Recovery of Non-Key Data from RAM

If the bootloader is locked, we additionally have to unlock the bootloader before overwriting the recovery partition. Unfortunately, this process requires us to confirm the following warning on the phone: "To prevent unauthorized access to your personal data, unlocking the bootloader will also delete all personal data from your phone". Once we confirm this warning, the encrypted user partition gets wiped. Nevertheless, in that case FROST can still be used to recover sensitive RAM contents. Specifically, personal data such as contact lists, messages, photos, and calendar entries are of interest. Technically, we used the FROST recovery image to acquire full memory dumps. For our main test

Fig. 3. A Droid-bitmap in the RAM of a Galaxy Nexus device after $0s$, ε, 0.5, 1, 2, 4, and $6\,\mathrm{s}$ without power. The cold boot attacks have been deployed at room temperature.

case, we set up a Galaxy Nexus device as personal phone and used it for everyday communications over a week before we took the memory dump. We were able to recover 68 JPEG and 199 PNG pictures, 36 OGG tracks, 295 HTML and 386 XML files, 215 SQlite databases, 28 ZIP and 105 JAR archives, $1{,}214$ ELF binaries, 485 JAVA source codes, and $6{,}331$ text files. The memory dump of our main test case was near optimal, meaning that we had a bit error ratio of 0 percent according to Fig. 2. While most PNG images that we recovered were system images and logos, most JPEG files were personal photos. We were able to recover both, pictures that were recently taken and older pictures. In some cases, we recovered two variants of a photo, a small thumbnail and a high-resolution picture. In most cases, however, we could only recover the thumbnail variant.

We visualize the influence of bit errors in Fig. 3. For the picture series in Fig. 3, we used a 4,096-byte bitmap that exactly fits into one physical page. We used a bitmap rather than a JPEG, because using a JPEG entire blocks get destroyed rather than single pixels. We then increased the interval that the phone was without power during boot gradually from ε to 6 seconds. Whenever the header of a bitmap got destroyed, we fixed it manually in order to have an intact format of the image. Fig. 3 graphically shows the remanence effect and the distribution of bit errors.

Even more critical, we found personal text files and emails in RAM, and we found the entire chat-history from WhatsApp. We also explicitly searched for names of our contact list and we found each name to be present in RAM several times, as well as respective phone numbers and e-mail addresses. We recovered birthday dates from the calendar, and we even found passwords in RAM. For example, we searched for the SSID of our WiFi and could locate the according password in plaintext.

5.2 Attacks on Self-Encrypting Drives

We now present results from applying attacks against SEDs in practice. An early insight from our experiments was that system security depends on the motherboard and BIOS version, but not on the particular SED model. We never found different behaviors among different SEDs, and so we did extensive testing with only two drives: the Intel SSD 320, and the newer Intel SSD 520. On the system side, we investigated a set of twelve computer systems from three different vendors: four workstations from Fujitsu, four laptops from Lenovo, and four laptops from Dell.

All test systems are from 2010 or later, with the exception of a ThinkPad R60e from 2008. The choice of systems was mainly determined by models that were available in our department, but the set also represents models that are generally available on the market. We performed hot plug and cold boot attacks practically against all test machines whenever that was possible (see below). Furthermore, we performed DMA attacks against all laptops with DMA interface. In the official PCIe specification [35], it is stated that PCIe supports hot-plugging for desktop machines. However, we found this feature to be either not supported by the OS or the motherboard, and consequently we could not demonstrate DMA attacks against desktop machines.

Networked evil maid attacks were exemplarily demonstrated against one class of test machines, as explained in the next section.

5.2.1 Fujitsu Workstations

Fujitsu workstations were the targets of our networked evil maid variant. They are ideally suited for this kind of attack because they display a consistent, text-based password prompt. On an Esprimo P900, for example, the legitimate password prompt is displayed 11 seconds after power-up, while our fake prompt is displayed after 16 seconds. We find a delay of 5 s acceptable and not too suspicious for ordinary users.

System	Motherboard	BIOS
Esprimo P9900	D2912-A1	Phoenix V6.0R1.20.2912
Esprimo P900	D3062-A1	AMI V4.6.4.0R1.5.0
Esprimo P900	D3062-A1	AMI V4.6.4.0R1.14.0
Celsius R570-2	D2628-C1	Phoenix V6.0R1.21.2628

More interesting than evil maid attacks, however, are attacks against running machines. Cold boot attacks are not defeated by any of the Fujitsus because BIOS settings for "ATA password on reboot" are missing. Hot plug attacks are not defeated on Fujitsu PCs, too, because the PC chassis can be opened easily at run time, such that SATA cables can be replugged to another machine. Moreover, hot plug attacks against S3 are not defeated because BIOS settings for "ATA password on S3 wakeup" are missing.

- *Fujitsu Esprimo P9900*. The P9900 saves the ATA password in NVRAM after it has been set, and never prompts for it again, neither on boot nor when we completely remove the SED. The SED gets locked when it is cut from power, but it gets always unlocked automatically on boot. Hence, the P9900 renders most attacks including evil maid attacks superfluous.
- *Fujitsu Esprimo P900/R1.5*. The P900 is more secure than its predecessor P9900, because a password must be entered on boot and on S4 wakeup. Apart from that, the P900 is as vulnerable as the P9900, i.e., it is vulnerable to hot plug and cold boot attacks.

- *Fujitsu Esprimo P900/R1.14*. This PC is almost identical to the previous one. The reason that we list it is that Fujitsu added a BIOS setting to store the ATA password inside NVRAM, thus eliminating the need to enter it on boot, effectively behaving like the P9900. However, even if we do *not* enable this option, the password is stored in NVRAM. As a consequence, we were able to circumvent the password prompt as follows: Once the prompt appears, we hit `Enter` and `F2` repeatedly, and the BIOS opens without ATA password due to implementation flaws. Inside BIOS, we can simply disable "ATA password on boot" and then reboot. From there on, the disk gets unlocked automatically—without requiring the attacker to know the password.
- *Fujitsu Celsius R570*. This PC behaves similarly to the Esprimo P900/R.1.5, meaning that a password is required only on boot. The ACPI state S3 did not work for us, but we suppose that the password is not needed for wakeup from S3.

5.2.2 Lenovo Laptops

Contrary to Fujitsu PCs, DMA attacks can be deployed against all Lenovo ThinkPads of our test set since each of it has a DMA port which is suitable for hot-plugging (FireWire, ExpressCard, or PCMCIA). This compensates the fact that hot plug attacks are mostly not applicable. Lenovo detects if SEDs are disconnected from power during S3 and hence, hot plug attacks are prevented. Nevertheless, attacks are equally effective against running machines and machines in S3, because options for "ATA password on S3 wakeup" are missing.

System	Motherboard	BIOS
ThinkPad R60e	0657CTO	7EETC6WW Rev. 2.22
ThinkPad T520	4242PT2	8AET52WW Rev. 1.32
ThinkPad x201	3323DBG	6QET52WW Rev. 1.34
ThinkPad x220i	4290G53	8DET54WW Rev. 1.24

- *Lenovo ThinkPad R60e*. The R60e has a PCMCIA slot and is therefore vulnerable to FireWire attacks. Moreover, it is vulnerable to cold boot attacks and, most interestingly, it is the only Lenovo ThinkPad that is vulnerable to hot plug attacks. Since Lenovo's SED removal detection was missing back in 2008, an attacker can put the R60e to sleep and install SATA extension cables.
- *Lenovo ThinkPad T520*. The T520 prompts for ATA passwords on reboot and detects if SEDs are unplugged during S3, thus defeating both cold boot and hot plug attacks. However, this laptop comes with two DMA ports and does not require ATA passwords on S3 wakeup, enabling DMA attacks.
- *Lenovo ThinkPad x201*. The x201 behaves similar to the T520, with the difference that it does not shut down on SED removal detection but displays a prompt to reenter the password. The x201 has a DMA port, namely ExpressCard, for DMA-based attacks.
- *Lenovo ThinkPad x220i*. The x220i behaves similar to the x201 and x220i series. DMA-based attacks are possible due to an ExpressCard interface, cold boot attacks are not possible, and hot plug attacks are at least not trivial.

5.2.3 Dell Laptops

Whereas Lenovo ThinkPads have a more or less consistent configuration, there is a great variety among laptops from Dell. Two of the weakest laptops in our test set (Precision M4600, and Latitude XT3) as well as the most secure laptop (Latitude 2120) are among them.

System	Motherboard	BIOS
Vostro 3300	030DMJ-A10	Dell Inc. A10Rev.1.2
Precision M4600	O8V9YG-A00	Dell Inc. A05Rev.4.6
Latitude 2120	0YY3FH	Dell Inc. A01Rev.1.0
Latitude XT3	067RKH-A00	Dell Inc. A00Rev.4.6

- *Dell Vostro 3300*. The Vostro 3300 prompts for ATA passwords on boot, reboot, and S4 wakeup, but not on S3 wakeup, and does not detect SED removals during S3. Hence, it is not vulnerable to cold boot, but to hot plug attacks. Furthermore, DMA attacks are possible because of an ExpressCard slot.
- *Dell Precision M4600*. The Precision M4600 has two DMA ports: FireWire and ExpressCard. ATA passwords are only required on boot but not on reboot, and SED removals are not detected. Hence, it is one of the weakest laptops, allowing for FireWire, cold boot, and hot plug attacks.
- *Dell Latitude 2120*. The netbook Latitude 2120 is the strongest machine in our test set, because (1) it has no DMA port, (2) it requires ATA passwords on boot, reboot, and S4 wakeup, and (3) it requires ATA passwords even on S3 wakeup. Neglecting the risk of evil maid attacks, we were not able to attack this machine.
- *Dell Latitude XT3*. The Latitude XT3 tablet has a built-in FireWire port as well as an ExpressCard slot for DMA attacks. Furthermore, ATA passwords are only required on boot and S4 wakeup, and SED removals are not detected, thereby enabling both cold boot and hot plug attacks.

5.2.4 Summary of Results

Our test results are summarized in Fig. 4, giving an overview about system-specific behaviors per target. Note that during our tests, we assumed that the strongest ATA security configuration *which is possible* is enabled. For example, if settings for "ATA password on reboot" and "ATA password on wakeup" exist, they are enabled.

The first block in Fig. 4 indicates whether ATA passwords are required to access a system. With the exception of two Fujitsu PCs, which store the passwords in NVRAM permanently, this is always the case on boot and on S4 wakeup. But only four out of twelve systems prompt for ATA passwords on reboot and reset. Even worse, only one out of twelve systems prompts for ATA passwords on S3 wakeup.

		Fujitsu		Lenovo				Dell		
		P9900/P900	R570/P900	R60e	T520	x201	x220i	3300	M4600/XT3	2120
ATA-PW	on boot		X	X	X	X	X	X	X	X
	soft reboot				X		X	X		X
	hard reset				X		X	X		X
	S3 wakeup									X
	S4 wakeup		X	X	X	X	X	X	X	X
DMA	FireWire				X				X	
	ExpressCard				X	X	X	X	X	
	PCMCIA			X						
S3 removal detection					X	X	X			

Fig. 4. Summary about relevant behaviors and properties in our test set. An *X* indicates that a certain behavior or property is available. ATA passwords and S3 removal detections are desirable properties from a security point of view, i.e., an *X* makes a system more secure. DMA interfaces, on the other hand, make a system more insecure and consequently, the absence of an *X* increases security.

The second block in Fig. 4 indicates whether hot-pluggable DMA ports are present. Seven out of eight laptops have such an interface, but none of the desktop systems. With the exception of the R60e, which has an older PCMCIA port, laptops usually have ExpressCard slots or additionally a FireWire port. The last row of Fig. 4 indicates whether a laptop detects the removal of an SED power connector during S3. This feature seems to be Lenovo-specific. Three out of four ThinkPads have this feature, but none of the Dell or Fujitsu systems.

To conclude, for every setting in which an attack on software-based FDE exists, we could perform an attack on hardware-based FDE. These scenarios include DMA-based attacks, cold boot attacks, and evil maid attacks. Moreover, we identified the threat of *hot plug attacks*, which are surprisingly simple but effective against hardware-based FDE in particular. Not all systems are equally vulnerable because the security of an SED depends on its environment. The majority of machines is equally vulnerable with hardware- and software-based FDE, and some machines are arguably more vulnerable with hardware-based FDE.

6 Forensic Guidelines to Thwart FDE

The increasing use of FDE has severe consequences for IT forensics because it prevents access to digital evidence [36]. Once an examiner shuts down an encrypted target PC or smartphone, which is a widely accepted procedure to prevent the corruption of evidence, purely technical ways to recover unencrypted data become difficult. Practical attacks against full disk encryption, such as cold boot attacks and hot plug attacks, succeed only if the target device is running or in standby mode. As a consequence, forensic examiners must reconsider the practice of "pulling the plug" of seized PCs and smartphones. Instead, they must be equipped with techniques for data recovery in the field, including equipment for cold boot, hot plug, and DMA-based attacks. Summarizing our insights from Sections 3, 4, and 5, we now provide general guidelines that point to the most promising attack for hardware-based FDE and software-based FDE.

6.1 Guidelines for Hardware-Based FDE

Fig. 5 illustrates a decision graph that leads to the *most suitable attack* against an SED for different systems. Basically, the graph differentiates between the construction type of stationary devices (desktops, workstations, servers) and mobile devices (laptops). For stationary devices, hot plug attacks are always possible but for laptops, other attacks must be taken into account. If a machine is switched off, evil maid attacks are mostly the only option. But if a machine is running, or in standby mode, several alternatives are possible. We therefore suggest the following priority order of attacks:

1) *Hot plug attacks*. Hot plug attacks are the most generic attacks as they are never defeated by the construction type of desktop PCs. The situation is different for laptops, but we suggest hot plug attacks as the first option for laptops, too, whenever SATA extension cables can be installed.
2) *DMA-based attacks*. DMA attacks are the second most generic attacks, because they can be deployed against all systems with hot-pluggable DMA interfaces, i.e., against most laptops. However, on many

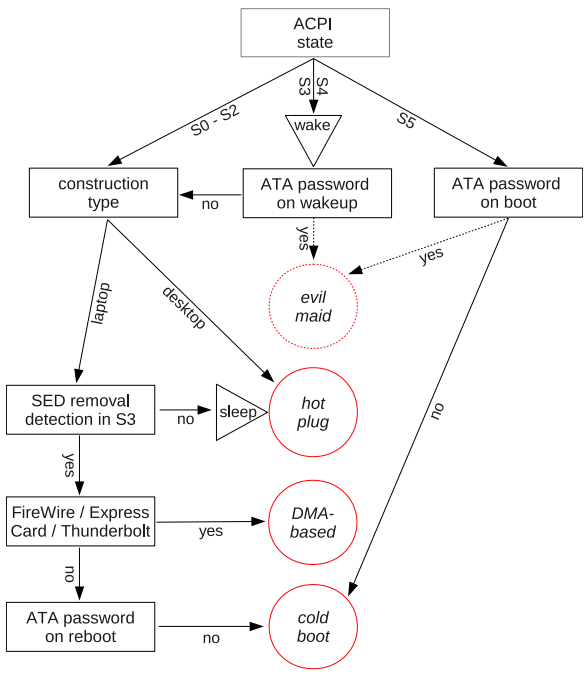

Fig. 5. Decision graph for attacking self-encrypting disks. The graph does not consider all possible attacks per system, but leads to the most promising one.

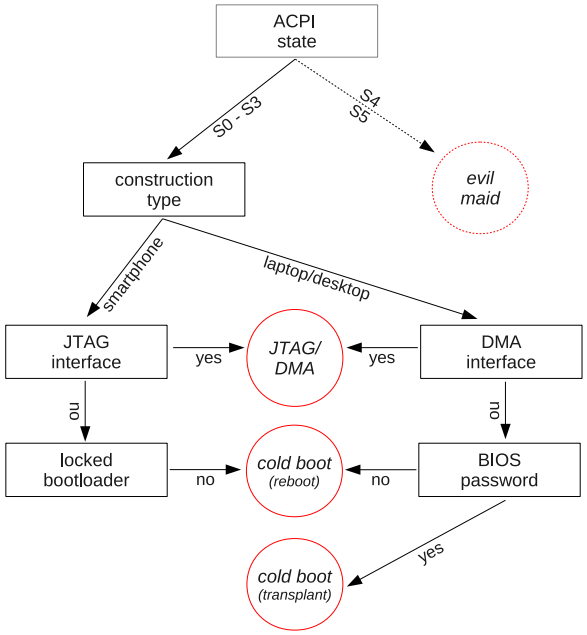

Fig. 6. Decision graph for attacking software-based FDE. Again, the graph does not consider all possible attacks per system, but leads to the most promising one.

desktop PCs hot-pluggable DMA ports are not available. Broadly speaking, DMA attacks are most promising against laptops, and hot plug attacks are most promising against desktops.

3) *Cold boot attacks*. Cold boot attacks depend on additional BIOS settings like the "supervisor password" and "boot device order" which are not foreseeable by attackers. Therefore, rebooting a PC with a live system from USB drive should only be considered if other attacks fail.
4) *Evil maid attacks*. Evil maid attacks are always possible, but they are not always relevant in the scenario of lawful seizure because the legitimate user has to take action. We consider evil maid attacks only if the PC was switched off or in standby mode, or if the user expects an ATA password prompt.

6.2 Guidelines for Software-Based FDE

Fig. 6 illustrates a decision graph for software-based FDE. Again, the graph leads to the most suitable attack per system. For software-based FDE, we differentiate between the construction type of smartphones and the construction type of ordinary PCs, including laptops and desktops. We identify three types of attacks against systems that are running or in standby mode: (1) system infiltration via interfaces like FireWire and JTAG, (2) cold boot attacks that reboot a system, and (3) cold boot attacks that replug RAM chips and "transplant" them to another device. Based on these attacks, we suggest the following priority order of attacks against software-based FDE:

1) *DMA/JTAG-based attacks*. We consider system code infiltration via DMA and JTAG as most promising because, given that such an interface is present, these attacks are deterministic. DMA-based attacks do not fail if forensic investigators have knowledge about the targeted OS and the appropriate equipment at hand.
2) *Cold boot attacks (reboot)*. Cold boot attacks that acquire keys from RAM after a short power-down (reboot) can be difficult, because BIOS passwords and locked bootloaders are not foreseeable for forensic investigators. However, we suggest this cold boot variant first because the bit error rate is negligible (due to the short power-down). On smartphones, for example, FROST can be used.
3) *Cold boot attacks (transplantation)*. Cold boot attacks that acquire keys from memory after RAM chips have been transplanted into another PC are more precarious. Even when chips are cooled down, the bit error rate increases rapidly after power is cut. However, this cold boot variant can be deployed *after* other attacks failed. For example, when a BIOS password is required at reboot, RAM chips can still be removed.
4) *Evil maid attacks*. Again, evil maid attacks are always possible in theory, but their relevance during lawful seizures is questionable.

7 Conclusions and Countermeasures

Today, the most effective countermeasure against attacks on FDE is to leave PCs unattended only after power-off (S5). This countermeasure, however, is inconvenient in practice. To increase FDE security in the future, we identify four levels for practical countermeasures: users, operating systems, BIOS/motherboards and SEDs.

- *User level*. Practical countermeasures that can be taken by users are strong BIOS settings, including a restrictive boot order as well as a boot password which is different to the SATA drive password. This can defeat at least a class of cold boot attacks.
- *OS level*. To defeat DMA and JTAG attacks, systems without DMA and JTAG ports can be bought. However, this measure again is unreasonable for many users. Instead, DMA attacks must be defeated on the OS level. With the virtualization technology for directed I/O, DMA transfers can be filtered by means of the IOMMU. While it was originally invented to restrict virtualized guests, the IOMMU can also prevent DMA devices from accessing system space. At the time of this writing, it is not supported by current OSs.
- *BIOS level*. As stated in Section 4.1.3, BIOS vendors can take action to secure hardware-based FDE. For example, SATA passwords must never be stored in RAM or NVRAM. When a SED loses power, it must never get unlocked automatically, and the user must always be prompted to re-enter his or her password manually. It is implemented like this only in one of our test systems (Dell Latitude 2120).

To increase the security against evil maid attacks, BIOS vendors must begin to pair user passwords with the TPM state before communicating it via ATA commands to the drive. In Microsoft BitLocker, for instance, the decryption key depends on the TPM

state, effectively preventing the drive from being decrypted on other machines.

- *SED level.* As stated in Section 4.1.3, to defeat hot plug attacks independently of the OS, a hardware-based locking mechanism must be introduced that is sensitive to the SATA cable connectivity. Today, self-encrypting drives get locked only when power is cut, but ideally locks should be based on the *data* connection.

  Another countermeasure in this direction is to connect SEDs to the board in a way that power and data is transmitted over the same carrier, as it is already the case for RAM modules and PCIe devices today. There are even SSDs that get connected via PCIe for performance reasons, like Intel's SSD 910 series [37]. Such SSDs, if they were self-encrypting, would be secure against hot plug attacks.

Summarizing, security of hardware-based FDE is manifold and includes at least four layers: user behavior, operating systems, BIOS/motherboards, and self-encrypting drives. Each layer is a potentially weak point, but can be used to increase FDE security in future, too. Although we disclosed some flaws of hardware-based FDE, security can evolve in future.

REFERENCES

[1] M. Dornseif, M. Becher, and C. N. Klein "FireWire—All your memory are belong to us," in *Proc. Annu. CanSecWest Appl. Secur. Conf.*, 2005.

[2] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: Cold boot attacks on encryptions keys," in *Proc. 17th USENIX Security Symp.*, Aug. 2008, pp. 45–60.

[3] J. Rutkowska. (2009, Oct.). Evil maid goes after trueCrypt [Online]. Available: http://theinvisiblethings.blogspot.com/2009/10/evil-maid-goes-after-tru ecrypt.html, The Invisible Things Lab.

[4] T. Müller, "(Un)Sicherheit hardware-basierter festplattenverschlüsselung," in *Proc. 29th Chaos Commun. Congr.*, Dec. 2012.

[5] T. Müller and M. Spreitzenbarth, "FROST: Forensic recovery of scrambled telephones," in *Proc. 11th Int. Conf. Appl. Cryptography Netw. Security*, Jun. 2013, pp. 373–388.

[6] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. 4th USENIX Conf. Workshop Offensive Technol.*, Aug. 2010, pp. 1–7.

[7] M. Kumar, "Android facial recognition based unlocking can be fooled with photo," The Hacker News, Nov. 2011.

[8] C. Stevens *AT Attachment 8—ATA/ATAPI Command Set (ATA8-ACS)*, Revision 6 ed., Working Draft Project Americaion Nat. Standard Jun. 2008.

[9] M. Szczys. (2011, Feb.). Hard drive password recovery [online]. Available: hackaday.com/2011/02/18/hard-drive-password-recovery/, Hack a Day.

[10] D. Hulton, "Cardbus bus-mastering: 0wning the laptop," in *Proc. ShmooCon*, Jan. 2006.

[11] C. Devine and G. Vissian, "Compromission physique par le bus PCI," in *Proc. SSTIC*, Jun. 2009, pp. 169–193.

[12] B. D. Carrier and E. H. Spafford, "Getting physical with the digital investigation process," *Int. J. Differential Equations*, vol. 2, no. 2, pp. 1–20, 2003.

[13] R. D. Graham. (Feb., 2011). Thunderbolt: Introducing a new way to hack Macs [Online]. Available: http://erratasec.blogspot.com/2011/02/thunderbolt-introducing-new-way-t o-hack.html

[14] P. Panholzer "Physical security attacks on windows vista," SEC Consult Vulnerability Lab, Vienna, Austria, Tech. Rep., May. 2008.

[15] B. Böck, *Firewire-Based Physical Security Attacks on Windows 7, EFS and BitLocker*, Secure Business Austria Research Lab, Aug. 2009.

[16] T. Müller, F. Freiling, and A. Dewald, "TRESOR runs encryption securely outside RAM," in *Proc. 20th USENIX Secur. Symp.*, Aug. 2011, p. 17.

[17] E.-O. Blass and W. Robertson, "TRESOR-HUNT: Attacking CPU-bound encryption," in *Proc. Annu. Comput. Appl. Conf.*, Dec. 2012, pp. 71–78.

[18] T. Müller, B. Taubmann, and F. Freiling "TreVisor: OS-independent software-based full disk encryption secure against main memory attacks," in *Proc. 10th Int. Conf. Appl. Cryptography Netw. Secur.*, Jun. 2012, pp. 66–83.

[19] P. Gutmann, "Data remanence in semiconductor devices," in *Proc. 10th USENIX Secur. Symp.*, Aug. 2001.

[20] S. Skorobogatov, "Data remanence in flash memory devices," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2005, pp. 339–353.

[21] R. Anderson and M. Kuhn, "Tamper resistance—A cautionary note," in *Proc. 2nd USENIX Workshop Electron. Commerce Proc.*, Nov. 1996, p. 1.

[22] A. Rahmati, M. Salajegheh, D. Holcomb, J. Sorber, W. Burleson, and K. Fu, "TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices withou clocks," in *Proc. 21st USENIX Secur. Symp.*, Aug. 2012, p. 36.

[23] N. Saxena and J. Voris, "We can remember it for you wholesale: Implications of data remanence on the use of RAM for true random number generation on RFID tags," in *Proc. 5th Workshop RFID Secur.*, Jul. 2009.

[24] T. Müller, A. Dewald, and F. Freiling, "AESSE: A cold-boot resistant implementation of AES," in *Proc. 3rd Eur. Workshop Syst. Secur.*, Apr. 2010, pp. 42–47.

[25] P. Simmons, "Security through amnesia: A software-based solution to the cold boot attack on disk encryption," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2011, pp. 73–82.

[26] J. Götzfried and T. Müller, "ARMORED: CPU-bound encryption for Android-driven ARM devices," in *Proc. 8th Int. Conf. Availability Rel. Secur.*, Sep. 2013, pp. 161–168.

[27] P. Kleissner, "Stoned Bootkit," in *Proc. Black Hat Conf.*, Jul. 2009.

[28] TrueCrypt Foundation. (Mar., 2014). TrueCrypt: Free open-source on-the-fly disk encryption software for windows, Mac OS X and Linux [Online]. Available: http://www.truecrypt.org/

[29] S. Türpe, A. Poller, J. Steffan, J.-P. Stotz, and J. Trukenmüller, "Attacking the BitLocker boot process," in *Proc. Trusted Comput. 2nd Int. Conf.*, Apr. 2009, pp. 183–196.

[30] T. Müller, H. Spath, R. Mäckl, and F. Freiling, "STARK tamper-proof authentication to resist keylogging," in *Proc. Financial Cryptography Data Secur.*, Apr. 2013, pp. 295–312.

[31] A. Fujimoto, P. Peterson, and P. L. Reiher, "Comparing the power of full disk encryption alternatives," in *Proc. Green Comput. Conf.*, Jun. 2012, pp. 1–6.

[32] Break & Enter: Improving security by breaking it. (Mar., 2014). Inception [Online]. Available: http://www.breaknenter.org/projects/inception/

[33] N. Heninger and A. Feldman. (Jul., 2014). AESKeyFind [Online]. Available: https://citp.princeton.edu/research/memory/code/

[34] M. Zugelder. (Apr., 2012). androidcrypt.py [Online]. Available: https://github.com/michael42/ androidcrypt.py/

[35] *PCI Express Base Specification*, Revision 1.0a ed., Apr. 2003.

[36] E. Casey and G. Stellatos, "The impact of full disk encryption on digital forensics," *Digit. Invest.*, vol. 8, pp. 129–134, 2011.

[37] *Solid-State Drive 910 Series*, Intel Corporation. (2012) [Online]. Available: http://www.intel.com/content/www/us/en/solid-state-drives/ssd-910-serie s-specification.html

Tilo Müller studied computer science from the University of Aachen and the University of Helsinki. He received the diploma from the University of Aachen in 2010. He received his doctoral degree in 2013. After receving the diploma, he has been employed as a research assistant at the chair for IT Security Infrastructures in Erlangen. His research interests include system security, mobile security, and software protection.

Felix C. Freiling is a professor of computer science at Friedrich-Alexander-University Erlangen-Nuremberg (FAU). He is interested in the theory and practice of dependable distributed systems.