

A Brief Reflection on Trusted Platform Module Support

Martin Pirker^a and Robert Haas

Institute of IT Security Research, St. Pölten University of Applied Sciences, Austria

Keywords: Trusted Computing, TPM, TSS.

Abstract: Trusted Computing and its Trusted Platform Module were introduced about 20 years ago. However, their impact is still limited, only a small number of applications use a TPM, only a few people know that their computer hosts one and what it can be used for. With the ongoing transition from now dominant Windows 10 to Windows 11, every common PC is required to have a TPM to run Windows 11. This short paper reflects on the current environment and state of support for TPMs. It investigates a selection of TPMs, their features, and surveyed the available software stacks to use them. It reports on the findings and the finer details discovered while using TPMs. Overall, this paper contributes to the ongoing discovery and learning about TPM v2, as it will be inevitably a part of our computing with PCs future.

1 INTRODUCTION

We live a digital life—on our PCs, laptops, mobile phones, tablets, or other devices. As the latter ones increase their market share, traditional PCs (or laptops) are still used for much of our computing needs. For common end-users the standard PC-class device is usually sold with and runs the operating system Microsoft Windows. An observation here is that any market majority has positive and negative qualities.

From a security point of view, if the majority of users run the same operating system, it is important that there are no grave security issues. Microsoft continuously invests resources and engineering effort to harden Windows installations against attacks and malicious modifications. To assess and enforce the state of an operating system's installation is not as easy as it appears to an end-user. An end-user may just install and run a virus scanner; however more technically knowledgeable minds know that software that searches for other software running on the same platform is not enough. Once malicious software is able to compromise an operating system early in the boot process, it can (theoretically) hide very well, if not perfectly, from detection attempts by other software.


With the introduction of Windows 11 (in 2021) Microsoft now demands¹ all PCs to have a Trusted Platform Module (TPM) v2 and UEFI secure boot.

This is an explicit requirement for a component, a dedicated TPM hardware chip or an equivalent implementation, that provides certain security support functions, in order to implement a higher level of security.

The Trusted Platform Module is not a new technology. It was devised more than 20 years ago and version 1 has been available on the market for about 15 years, it being replaced in recent years with the improved and reimagined major revision version 2.

Although TPMs and their security potential have been ignored by many until now, with the expected end-of-life of current standard Windows 10, its main retirement date projected to be October 2025² and Windows 11 succeeding it, in the near future every PC will have a TPM v2. One could argue Microsoft is pushing the TPM into the market by force, however one could also see an improvement of security on every PC as a result, which would be very impressive.

Contribution. This short paper reflects on the current overall state of support for TPM v2. For this, we do a brief survey on the availability of TPMs v2 with hardware available to us, their revisions, and features. Then, we examine what (open-source) software support is available to work with the TPM v2 already today, both under Windows and Linux. We report on the insights we learned and reflect on them.

^a  <https://orcid.org/0000-0003-3517-5479>

¹ <https://www.microsoft.com/en-us/windows/windows-11-specifications>

²Enterprise long-term support can be bought separately. <https://learn.microsoft.com/en-us/lifecycle/products/windows-10-home-and-pro>

Outline. The background Section 2 first provides a brief review on the history of Trusted Computing and the Trusted Platform Module, then examines the security of TPMs, and well-known selected TPM-using applications. Section 3 provides our detailed findings on the TPM v2 ecosystem: The TPMs examined, a survey of TSSs, and some learnings from using them. The paper concludes in Section 4.

2 BACKGROUND

2.1 Trusted Computing History

More than 20 years ago, if PCs failed due to “computer problems”, for example a computer virus that infected all PCs of a hospital, this was worthy for a report in the evening TV news (and was obviously a novel problem for all depending on the care of the hospital). PCs deployed everywhere would surely not disappear again, but inevitably PCs would be used in more and more applications.

Mass-market PC Security. The challenge of PC reliability and security inspired the so-called Trusted Computing Platform Alliance (TCPA). They observed: *Since e-Business runs on the PC, enhancing trust in the computing platform is an issue of fundamental and growing importance for the PC industry.* (The Trusted Computing Platform Alliance, 2000). Their insight was the augmentation of the common PC platform with a hardware-based trust component and integrating it into security processes. They proposed the Trusted Platform Module (TPM), a new hardware chip that provides basic primitives supporting security such as public-key cryptography, isolated key generation, cryptographic hashing, true random-number generation, and more supplemental functions useful in implementing and strengthening trust and security.

TPM v1.x Generation. The first version of the TPM design ready for the mass-market was v1.1b (Trusted Computing Platform Alliance (TCPA), 2002), published by the now renamed consortium Trusted Computing Group (TCG). Software support followed with the TCG Software Stack (TSS) Specification (Trusted Computing Group (TCG), 2003a). The TSS specification provides an architecture on how to integrate the TPM, from the low-level chip, over software layers with C-language specified functions, up to an application that wants to take advantage of the TPM. The TPM v1.2 followed

soon (Trusted Computing Group (TCG), 2003b) and an update for the TSS (Trusted Computing Group (TCG), 2006) a bit later.

Once TPM chips became more widely available, integrated into common PC mainboards and therefore easy to buy, hardware vendors also included a TSS for the specific chip on the mainboard. Although the TSS C-language functions themselves were specified by TCG, this was still insufficient for code to “just recompile” with a TSS provided by a different vendor than it was originally developed for.

There were also two major open-source efforts to provide a full TSS, they enabled open learning, understanding, and experimenting with Trusted Computing and the TPM: 1) The TrouSerS³ effort, a TSS written in C, and the Trusted Java⁴ effort, a TSS (and more components beyond that) implemented in Java.

This first generation of TPMs and related technologies were not perfect, as to be expected from novel, complex security technologies. The specifications were not unambiguous enough.⁵ Consequently, the TCG established a certification program (Trusted Computing Group (TCG), 2009), where vendors could verify security and functional compliance of their products that they developed. The first discrete TPM chip was certified at the end of 2009.⁶

TPM v2.x Generation. The TPM v1.2 was for about 15 years the standard, however the 1) known issues, subtle specification imprecisions and incompatibilities needed to be fixed, 2) cryptography advanced and the core algorithms of the TPM v1.2, RSA-based keys and SHA-1 as core hash function, were no longer state of the art (secure), and 3) technological progress such as hardware virtualization⁷ in common PCs posed new challenges, such as the transparent sharing of one hardware TPM between different virtual machines. The TCG introduced the TPM v2.0 revision. (Trusted Computing Group, 2013). It was a major redesign of its commands and data structures; it tackles the known problems and also enables new types of applications and scenarios.

³<https://trousers.sourceforge.net/>

⁴<https://trustedjava.sourceforge.net/>

⁵For example, see Trusted Java TSS file `src/jtss.tsp/src/iaik/tc/tss/impl/java/tsp/TcPcrCompositeInfo.java`, function `getNumPcrs()` for figuring out the number of PCRs on a TPM v1.2. Practical testing with different TPM implementations showed that a special case for one vendor was required.

⁶<https://trustedcomputinggroup.org/membership/certification/tpm-certified-products/>

⁷In the 2005/2006 timeframe both Intel and AMD introduced hardware virtualization support into their processors.

Still, the TPM v2 sees limited application use today, although right from the start accompanying literature from authors close to the v2 specification process was available, for example (Proudlar et al., 2014) and (Arthur et al., 2015).

2.2 TPM v2 Applications

As a TPM is usually affixed to a certain platform, this predestines its use for platform related applications.

Bitlocker. One well-known application in the Windows operating system is Bitlocker, a full disk encryption (FDE) software with the ability to harness TPM functionality. Bitlocker uses a Volume Master Key (VMK) and a Full Volume Encryption Key (FVEK), which is protected by the VMK, for each encrypted volume. If a measurement chain of the boot process is available, a TPM key can be sealed to the correct boot state. The TPM-based key is then used to encrypt the VMK.⁸ An optional PIN-code can provide additional authentication, in addition to the required unmodified boot process. A separate recovery key can be configured to bypass all this in case of TPM failure or a forgotten PIN.⁹

LUKS. On Linux, the Linux Unified Key Setup (LUKS) in cooperation with the dm-crypt subsystem provides full disk encryption. The architecture is similar to Bitlocker, by using an equivalent approach to the VMK and FVEK. However, additional software components for the use of TPM keys with LUKS are needed. The Systemd¹⁰ suite contains the tooling to interact with the TPM: Sealing and unsealing keys for LUKS use, also including support for sealed TPM keys with an additional PIN.

2.3 TPMs and Security

The introduction of TPMs for common PCs in order to enhance their security and their expected use, for example in the operating system Windows 11, might convey a “with the TPM everything is secure” thinking. This TPM chip magically solves security issues. However, for a TPM to be a mass-market device certain trade-offs had to be accepted.

⁸<https://blog.scr.ch/2023/09/15/a-deep-dive-into-tpm-based-bitlocker-drive-encryption/>

⁹<https://web.archive.org/web/20230709213454/https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc732774%28v=ws.10%29>

¹⁰<https://media.ccc.de/v/all-systems-go-2023-186-linux-tpms>

Physical Attacks. If one has physical access to a platform with a TPM, then a TPM can be influenced to a certain degree. Some selected examples:

In 2010, an attack was demonstrated that involves removing layers and reverse engineering of a dTPM chip to probe specific locations inside. It shows that physical attacks, despite intricate protection schemes such as light sensors inside the chip, may eventually lead to success, given enough time (several months) and equipment (a focused ion beam microscope). (Tarnovsky, 2010)

In 2013, a paper describes attacks on dedicated hardware TPMs (dTPMs) v1.2 via LPC and I²C busses for embedded systems, by modifying clock cycle signals and selective resets of the TPM or platform. The approach requires some hardware and resembles a Man-in-the-Middle attack. The result is a broken chain of trust during boot, but without compromise of TPM secrets. (Winter and Dietrich, 2013)

A blog article¹¹ from 2019 describes the process to attack a dTPM v2 chip of a Surface Pro 3, by soldering an FPGA to the LPC connection. This allows reading the traffic coming from the TPM. The result is a decoded Bitlocker VMK, bypassing the FDE. The attack can be mitigated by configuring TPM+PIN in Bitlocker. A TPM v2 supports encrypted parameters, but this functionality was not used by Bitlocker.

Another blog article¹² from 2021 describes a similar attack on a dTPM v2 of a Lenovo Laptop, via the SPI bus. The laptop had the CMOS chip on the same SPI bus, which allowed easier physical connection. The result was a broken FDE and access to the OS.

In 2020, a paper described a timing-based side channel attack on Intel fTPM and STMicroelectronics dTPMs by reading the clock cycle count right before and after a TPM command is executed. The attack specifically targets the elliptic curve signature schemes in the TPM v2 standard. The authors managed to break VPN authentication relying on the TPM, remotely over network, by analysing replies to crafted traffic. (Moghimi et al., 2020)

In May 2023, (Jacob et al., 2023) presented a novel way to abuse voltage fault injection attacks, which is a previously released exploit of the AMD Secure Processor, to expose the internal fTPM state. This attack demonstrates that physical attacks are also available for AMD fTPMs.

Firmware Bugs. A modern TPM chip is a processor core that executes a firmware that implements

¹¹<https://pulsesecurity.co.nz/articles/TPM-sniffing>

¹²<https://dolosgroup.io/blog/2021/7/9/from-stolen-laptop-to-inside-the-company-network>

the functions of a TPM—and naturally, this firmware may be defective.

For example, in 2017 it was discovered (CVE, 2017) that one TPM vendor’s firmware mishandled RSA key pair generation, meaning the RSA key pairs were of insufficient quality and made it feasible for an attacker to recover the private key corresponding to a public key in certain cases.

In 2023 it was discovered that the reference implementation in the TPM 2.0’s Module Library contained an out-of-bounds write vulnerability (CVE, 2023a) and an out-of-bounds read vulnerability (CVE, 2023b). All products that used this C code as model for their own implementation inherited these problems (and would subsequently require a firmware update and reset to factory default values).

With the addition of firmware-based TPM functionality (fTPM) to AMD’s Ryzen 2xxx CPUs and newer models, an issue in the implementation resulted in a performance and stability impact, by intermittent freezings of the whole system (“stutter”). These freezes were noticeable by end users, who then made the connection between the fTPM and the issue during troubleshooting. As explained by AMD (AMD, 2022) the issue is a result of extended SPI-attached memory transactions. While end user reports appeared already in mid 2021,¹³ AMD only provided a solution around Q2 2022.

3 AN EXPLORATION OF THE TPM V2 ECOSYSTEM

3.1 Software Support

TPM Simulators. There are two software simulators that resemble a hardware TPM chip and therefore are invaluable tools to develop and debug code that is communicating with a TPM, as a TPM’s internal current state can be examined in detail.

1) Microsoft provides an *Official TPM 2.0 Reference Implementation*,¹⁴ which is true to its name, the official TCG reference implementation of the TPM 2.0 Specification. These are the same fragments of code that are part of the TPM v2 specification documents, as they are available from the TCG homepage.

2) An alternative implementation of the TCG TPM 2.0 specification is *IBM’s Software TPM 2.0*.¹⁵ It is “based on the TPM specification Parts 3 and 4

source code donated by Microsoft, with additional files to complete the implementation.”

TPM Software Stack. According to Wikipedia¹⁶ there are currently 5 noteworthy open-source TPM software libraries. Their features and APIs vary:

1) The *Trusted Computing Group’s (TCG) TPM2 Software Stack (TSS)*¹⁷ project is the official implementation of a TSS as envisioned by the TCG. It is open-source, permissively BSD-2 licensed, which makes it easy to integrate in projects. It is also full-featured, meaning it is the only TSS project that implements all layers, from a low-level System-API that resembles low-level TPM chip commands basically 1:1, up to a high-level Feature API that enables a simple(r) usage of the TPM for common use cases.

This TSS is the standard package that is usually included in Linux distributions. The included INSTALL documentation suggests Windows support, although the included VS solution appears outdated and required some tweaks for the libraries/DLLs to build under Windows at our time of testing.

2) *IBM’s TPM 2.0 TSS*¹⁸ is another implementation in C of a TSS-like library to talk to the TPM. It provides a 1:1 mapping of all TPM commands and some abstractions on top, plus sample CLI programs to exercise each command.

3) The *TPM Software Stack from Microsoft Research*¹⁹ (TSS.MSR) project appears to be an ambitious project to auto-generate the interfacing to the individual TPM commands for multiple programming languages. The project contains the TPM commands extracted from the specifications as XML data and a generator then creates code for the languages C(++), JavaScript/Typescript, Java, C#/NET and Python. We are unsure about the state/maintenance of the project and the quality/completeness of the individual languages/generated code. The last check-in of code in the code repository was in the middle of 2022, and for example for Python there are more actively maintained projects like *tpm2-pytss*.²⁰

4) The *wolfTPM*²¹ effort is a portable, open-source (GPL v2) TPM 2.0 stack implemented in C and designed for embedded use. It has native support for Linux, Windows, and smaller platforms like Raspberry Pi, STM32, and many more. Again, all TPM

¹⁶https://en.wikipedia.org/wiki/Trusted_Platform_Module#TPM_software_libraries

¹⁷<https://github.com/tpm2-software/tpm2-tss>

¹⁸<https://sourceforge.net/projects/ibmtpm20tss/>

¹⁹<https://github.com/microsoft/TSS.MSR>

²⁰This projects wraps the TCG’s TSS into a Python wrapper <https://github.com/tpm2-software/tpm2-pytss>

²¹<https://www.wolfssl.com/products/wolftpm/>

¹³For example in forum: <https://linustechtips.com/topic/1353904-amd-ftpm-causing-random-stuttering/>

¹⁴<https://github.com/microsoft/ms-tpm-20-ref>

¹⁵<https://sourceforge.net/projects/ibmswtpm2/>

commands and example code for common use cases are provided.

5) As another non-C implementation, *go-tpm*²² is a Go library that communicates directly with a TPM device on Linux or Windows machines. It does not implement the entire specification; however, the provided examples cover common use cases.

TPM Access Multiplexing. There is usually only one TPM hardware chip on a platform, and a TPM is usually a rather resource constrained device. If multiple applications want to use a TPM, access to it must be managed so that each applications gets access to the resources as required, and if necessary, data is swapped in or out from the TPM on demand. Both Windows and Linux must solve this.

Windows. On the Windows platform, Microsoft introduced an abstraction layer between applications and the TPM: The Windows TPM Base Services²³ were introduced with the Windows Server 2008 and Windows Vista generation as a system service. The TBS schedules calling applications so that they cooperatively get TPM access. Applications work with virtual resources provided by the TBS, for example virtualized key handles, but the TBS then transparently manages these to map to the actual physical resources of the underlying TPM.

Linux. On the Linux platform the so-called TPM2 Access Broker & Resource Manager system daemon implements the respective TCG specification. It swaps objects, sessions, and sequences in and out of the TPM’s limited resources as needed and handles multi-process synchronization of access to the TPM. A process accessing the TPM can be guaranteed that it will be able to complete a TPM command without interference from other competing processes. (Trusted Computing Group, 2019).

3.2 TPM Models

In order to get a better understanding on the availability of TPMs v2 (and their supported specification revision), we tested computers available to us. We chose models of different hardware generations, if possible. Unfortunately, the PCs available to us were mainly Intel-based platform, only one AMD-based, and further two kinds of virtual machines.

²²<https://github.com/google/go-tpm>

²³<https://learn.microsoft.com/en-us/windows/win32/tbs/tpm-base-services-portal>

We discovered the discrete hardware TPM chips were solely provided by Infineon (IFX). Some platforms contained Intel’s firmware-based TPM, which is part of Intel’s Platform Trust Technology (PTT),²⁴ and AMD’s firmware TPM (fTPM)²⁵

Table 1 is an overview of our collected data.

Table 1: Overview of hardware platforms queried for their TPM’s self-reported properties. Revision is TCG’s TPM specification version, errata format is day of year/year.

platform	rev	errata	chipmodel, by vendor
VMWare 17	1.16	15/2016	VMware TPM2, by VMW
Virtualbox 7	1.64	75/2021	SW TPM, by IBM
Intel GeminiLake	1.38	352/2019	Intel, by INTC
AMD Ryzen 3000	1.38	61/2018	AMD, by AMD
Intel 8 th generation	1.16	265/2016	SLB9670, by IFX
Intel 9 th generation	1.38	8/2018	SLB9670, by IFX
Intel 10 th generation	1.38	303/2019	Intel, by INTC
Intel 10 th generation	1.38	8/2018	SLB9670, by IFX
Intel 13 th generation	1.38	352/2019	ADL, by INTC

We observe several things:

1) The quite new 13th generation Intel CPU-based PC uses an integrated Intel firmware-based TPM that is still based on specification revision 1.38, and not on the newest TPM specification revision 1.59. The reported chipmodel ADL refers to previous 12th generation, and indeed the chipset in this PC was a B660, originally released for earlier Alder-Lake (12th gen) and the fTPM runs on the chipset, not the CPU.

2) The newest reported revision is 1.64, by the TPM of Virtualbox 7, although the TCG has not published such a specification, yet? When we check the sources of the SW TPM (see Section 3.1), indeed in *TpmTypes.h* it is 1.64. We assume a forthcoming revision is already being implemented and tested via this TPM software simulator, which is then integrated as a software TPM by the Virtualbox project.

3) The TPM errata levels vary wildly. The format of “day of year/year” is uncommon, but for example for version 1.38 there are really 10 errata documents²⁶ readily available from the TCG’s TPM page. The matching errata are identified by the timestamps. How and how much these small patches to the specification and resulting slightly different behaviours of TPMs affect practical deployment and use, we are not experienced enough to assess.

4) VMWare Workstation 17 Pro, released end of 2022, introduced a new virtual Trusted Platform Module 2.0 support, to support the Windows 11 operating

²⁴PTT does not run on the main CPU, but instead on the Intel’s Management Engine (Intel ME), an embedded microcontroller integrated on some Intel chipsets.

²⁵The AMD fTPM runs on Zen’s dedicated Platform Security Processor (PSP).

²⁶Errata 1.0 to 1.13, seemingly not all revs are published.

system.²⁷ We only had brief access to this setup, and now looking at this result in the context of the other platforms we tested, we would like to take another, closer look. However, unfortunately we were unable to do this in time for this paper.

3.3 Algorithms of TPMs

One of the learnings from the TPM 1.x generation was its hardwired use of only a restricted set of cryptography. As the years pass, cryptographers are creative and invent new attacks to weaken and even break established algorithms and protocols. To our knowledge the TPM v1 was never broken due to cryptography (see also Section 2.3), however its fixation on mainly RSA and SHA-1 in its design was unfortunate.

Therefore, the TPM v2 introduced algorithm agility throughout its data structures and the TCG specified (Trusted Computing Group, 2023a) so far about 60 algorithms (respectively ALG_ID constants) for various operations used by a TPM.

As an example, Table 2 lists the algorithms we found supported on our Intel B660-based platform fTPM (see Section 3.2). This list is quite extensive, with 30 different algorithms, about half of the ones the TCG has specified. In our testing the SW TPM used by Virtualbox offered the most, 33 algorithms in total. In contrast, an Infineon hardware TPM provides only 19, see Table 3. We interpret this due to the limited resources on a hardware chip, whereas a software implementation is not as constrained.

3.4 TPM Applications not in C

As discussed in Section 3.1, the TCG Software Stack specification is primarily C-language based. Naturally, programs are not solely implemented in C, other programming languages also want to communicate with the TPM. The already mentioned TSS.MSR and the tpm2-pytss projects provide for that.

An interesting aspect are the typical errors that happen in the C-language, see for example the out-of-bound reads/writes in Section 2.3. Instead of a TSS in C, one candidate is the modern (and rising in popularity) programming language Rust, which claims on its homepage:²⁸ *A language empowering everyone to build reliable and efficient software.*

At time of writing there is a project that wraps the official TCG C-based TSS and provides a Rust API,

²⁷<https://docs.vmware.com/en/VMware-Workstation-Pro/17.0/rn/vmware-workstation-170-pro-release-notes/index.html>

²⁸<https://www.rust-lang.org/>

Table 2: Algorithms implemented on the fTPM that is part of Intel’s B660 chipset. asy=asymmetric algorithm, sym=symmetric algo., hash=hash algorithm, obj=an object type, sign=a signing algorithm, enc=an encryption mode, met=a method such as a mask generation function.

id	algorithm	Asy	Sym	Hash	Obj	Sign	Enc	Met
0x01	Rsa	asy	—	—	obj	—	—	—
0x04	Sha1	—	—	hash	—	—	—	—
0x05	Hmac	—	—	hash	—	sign	—	—
0x06	Aes	—	sym	—	—	—	—	—
0x07	Mgf1	—	—	hash	—	—	—	met
0x08	KeyedHash	—	—	hash	obj	sign	enc	—
0x0a	Xor	—	sym	hash	—	—	—	—
0x0b	Sha256	—	—	hash	—	—	—	—
0x0c	Sha384	—	—	hash	—	—	—	—
0x12	Sm3_256	—	—	hash	—	—	—	—
0x13	Sm4	—	sym	—	—	—	—	—
0x14	RsaSsa	asy	—	—	—	sign	—	—
0x15	RsaEs	asy	—	—	—	—	enc	—
0x16	RsaPss	asy	—	—	—	sign	—	—
0x17	Oaep	asy	—	—	—	—	enc	—
0x18	EcDsa	asy	—	—	—	sign	—	met
0x19	EcDh	asy	—	—	—	—	—	met
0x1a	EcDaa	asy	—	—	—	sign	—	—
0x1b	Sm2	asy	—	—	—	sign	—	met
0x1c	EcSchnorr	asy	—	—	—	sign	—	—
0x20	Kdf1Sp800_56a	—	—	hash	—	—	—	met
0x22	Kdf1Sp800_108	—	—	hash	—	—	—	met
0x23	Ecc	asy	—	—	obj	—	—	—
0x25	SymCipher	—	—	—	obj	—	—	—
0x3f	Cmac	—	sym	—	—	sign	—	—
0x40	Ctr	—	sym	—	—	—	enc	—
0x41	Ofb	—	sym	—	—	—	enc	—
0x42	Cbc	—	sym	—	—	—	enc	—
0x43	Cfb	—	sym	—	—	—	enc	—
0x44	Ecb	—	sym	—	—	—	enc	—

Table 3: Algos implemented on an Infineon hardware TPM.

id	algorithm	Asy	Sym	Hash	Obj	Sign	Enc	Met
0x01	Rsa	asy	—	—	obj	—	—	—
0x04	Sha1	—	—	hash	—	—	—	—
0x05	Hmac	—	—	hash	—	sign	—	—
0x06	Aes	—	sym	—	—	—	—	—
0x08	Keyedhash	—	—	hash	obj	sign	enc	—
0x0a	Xor	—	sym	hash	—	—	—	—
0x0b	Sha256	—	—	hash	—	—	—	—
0x14	RsaSsa	asy	—	—	—	sign	—	—
0x15	RsaEs	asy	—	—	—	—	enc	—
0x16	RsaPss	asy	—	—	—	sign	—	—
0x17	Oaep	asy	—	—	—	—	enc	—
0x18	EcDsa	asy	—	—	—	sign	—	met
0x19	EcDh	asy	—	—	—	—	—	met
0x1a	EcDaa	asy	—	—	—	sign	—	—
0x20	Kdf1Sp800_56a	—	—	hash	—	—	—	met
0x22	Kdf1Sp800_108	—	—	hash	—	—	—	met
0x23	Ecc	asy	—	—	obj	—	—	—
0x25	SymCipher	—	—	—	obj	—	—	—
0x43	Cfb	—	sym	—	—	—	enc	—

the *TSS 2.0 Enhanced System API Rust Wrapper*.²⁹ It builds on the C-based code, with an automatically generated foreign function interface (FFI) layer and adds a custom Rust data types-based API on top.

The advantage is that Rust is e.g. stricter in checking of data types as C and thus certain errors do not

²⁹<https://github.com/parallaxsecond/rust-tss-esapi>

happen in Rust. A disadvantage is that certain API calls turn out to be unergonomic. For example, a query call to determine the maximum size of a digest that can be produced by a certain TPM is:

```
let res : Result = context.get_capability(
    CapabilityType::TpmProperties,
    TPM2_PT_MAX_DIGEST, 1);
```

The expected return value is the number of bytes, an integer. However, the returned data structure is:

```
Ok(
    (
        TpmProperties(
            TaggedTpmPropertyList {
                tagged_tpm_properties: [
                    TaggedProperty {
                        property: MaxDigest,
                        value: 48,
                    },
                ],
            },
            true,
        ),
    ),
```

It is a Rust Result (call succeeded=Ok), encapsulating an enum CapabilityData, more specifically a TpmProperties(TaggedTpmPropertyList), containing one TaggedProperty. It is the MaxDigest we asked for and the value is 48 (bytes) on this specific TPM.

While strong API data typing reduces errors, we wonder whether a design is possible where the trade-off between convenience of use and robustness is better? This Rust-based API is still a work in progress, as for example Windows support is still missing, but we are looking forward on how this project develops further.

3.5 TPMs and Certificates

The non-volatile storage of a TPM, meaning data stored in a TPM’s memory that survives power loss, is very limited. Certain data has to be kept inside the TPM, such as the unique Endorsement Key (EK). The EK can be thought of as the unique identity of a specific TPM. The public EK alone is not sufficient, an EK certificate supplies further information about the EK. A manufacturer issued EK certificate is then the evidence that a specific public key belongs to hardware TPM (and not to a software simulated TPM).

We surveyed the non-volatile storage of our TPMs for EK certificates. They have specific handle numbers in the 0x01c0xxx range (Trusted Computing Group, 2022). The most commonly found ones are just individual EK certificates, in RSA and ECC variants. An example here from our oldest IFX TPM:

```
0x01c00002 = RSA 2048 EK Certificate
0x01c0000a = ECC NIST P256 EK Certificate
```

Modern TPMs have more storage and also have supplemental certificates to build the certificate chain(s) for validation:

```
0x01c10102 = EK Certificate Chain
0x01c10103 = EK Certificate Chain
0x01c10104 = EK Certificate Chain
0x01c10105 = EK Certificate Chain
```

Some TPMs NV-storage contained nothing at all by default, such as the TPM of the Virtualbox TPM, which makes sense as it is a software simulator TPM. However, the NV storage was also empty on our GeminiLake platform with its Intel PTT-based TPM.

The B660 chipset TPM provided the most variants, one RSA-based and two ECC-based:

```
0x01c00002 = RSA 2048 EK Certificate
0x01c0000a = ECC NIST P256 EK Certificate
0x01c00016 = ECC NIST P384 EK Certificate(H-3)
```

The P384-one was uncommon and the Rust TSS (Section 3.4) did not initially support this handle in its API, to be able to retrieve it. However, a recent patch added additional certificate handle constants, as listed in the EK Credential Profile Specification.

3.6 Additional Insights

Detailed Parameters for Algorithms. The tables presented in Section 3.3 give an overview on the available algorithms, but for practical use this is not sufficient. While testing the TPMs, we found that AES as a standard algorithm for symmetric en-/decryption is always supported, but not all key sizes are.

The explanation for this behaviour can be found in TCG’s PC Client Platform TPM Profile (PTP) Specification (Trusted Computing Group, 2023b), which specifies for the PC platform the features to be implemented (a subset of all the features in the full TPM specification). In the PC Client TPM Algorithms table a note states *TPM_ALG_AES: SHALL support for 128- and 256-bit keys and TPM_ALG_CFB at a minimum. [...] AES 256 support is mandatory as of PTP 1.03.* Consequently, the first TPMs for PCs followed the initial version of the PTP specification, where AES-256 was not mandatory to be implemented.

This matches our observed behaviour that e.g. for the command TPM2_StartAuthSession the symmetric block cipher has to be TPM_ALG_CFB, and AES_128_CFB worked on all tested TPMs, but AES_256_CFB did not work on all of them.³⁰

Low-level TPM Interactions. Section 3.4 already showed an example of trade-offs to be made when programming the TPM. Another aspect of a strict API that implements all TCG specific commands and data

³⁰start_auth_session: value is out of range or is not correct for the context (associated with parameter number 4) Parameter 4 is *symmetric*, meaning *the algorithm and key size for parameter encryption*.

structures is that it does not implement vendor specific commands and data types.

Our selection of TPMs tested was limited, however a closer look at the wolfTPM (see Section 3.1) source, file `tpm2.h` specifically, suggests TPM vendors do indeed add custom additional low-level commands to their TPMs. This makes sense, as e.g. TPMs for embedded projects surely benefit from additional commands specific for embedded use. However, a TSS must offer these.

A strictly typed API, where the command codes are an enum(eration) of the officially specified TPM commands, and data input and outputs are strictly defined structs, may not be as easily adapted for these low-level commands. These extra commands for each vendor must be implemented in the TSS. So far, we have only found wolfTPM to support them.

4 CONCLUSION

Originally this paper was motivated by the upcoming requirements for a TPM v2 in every PC in the near future, driven by Windows 11. Once we identified candidate PCs in our working environment, we were motivated to explore the finer differences between the individual hardware and software TPM implementations. Further, as Wikipedia only provides a high-level overview, we set out to explore the support software stacks, their trade-offs and how to use the TPM v2 in different languages.

This paper provides an intermediate review, a snapshot of the ecosystem, a reflection of our insights and experiences made. This is still a work in progress and not an exhaustive survey. As we discovered, the TSSs are a work in progress, too. This short paper is another contribution in the discovery and learning about TPM v2.

ACKNOWLEDGEMENTS

The work presented in this paper was done at the Josef Ressel Center for Blockchain Technologies and Security Management, St. Pölten University of Applied Sciences, Austria.

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital, and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

REFERENCES

- AMD (2022). Intermittent System Stutter Experienced with fTPM Enabled on Windows® 10 and 11. <https://www.amd.com/en/support/kb/faq/pa-410>.
- Arthur, W., Challener, D., and Goldman, K. (2015). *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*. Apress Berkeley. <https://doi.org/10.1007/978-1-4302-6584-9>.
- CVE (2017). CVE-2017-15361. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-15361>.
- CVE (2023a). CVE-2023-1017. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-1017>.
- CVE (2023b). CVE-2023-1018. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-1018>.
- Jacob, H. N., Werling, C., Buhren, R., and Seifert, J.-P. (2023). *faultTPM: Exposing AMD fTPMs' Deepest Secrets*. <https://arxiv.org/abs/2304.14717>.
- Moghimi, D., Sunar, B., Eisenbarth, T., and Heninger, N. (2020). TPM-FAIL: TPM meets timing and lattice attacks. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2057–2073. USENIX Association. <https://www.usenix.org/conference/usenixsecurity20/presentation/moghimi-tpm>.
- Proudler, G., Dalton, C., and Chen, L. (2014). *Trusted Computing Platforms: TPM2.0 in Context*. Springer Verlag. <https://doi.org/10.1007/978-3-319-08744-3>.
- Tarnovsky, C. (2010). Hacking the Smartcard Chip. Blackhat DC'10. <http://www.blackhat.com/html/bh-dc-10/bh-dc-10-briefings.html#Tarnovsky>.
- The Trusted Computing Platform Alliance (2000). Building A Foundation of Trust in the PC. TCGA Whitepaper, <http://www.trustedcomputinggroup.org/>.
- Trusted Computing Group (2013). TPM Specification Version 2.0 Revision 00.96. <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.
- Trusted Computing Group (2019). TCG TSS 2.0 TAB and Resource Manager Specification, Family 2.0, Level 00 Version 1.0 Revision 18. <https://trustedcomputinggroup.org/resource/tss-tab-and-resource-manager/>.
- Trusted Computing Group (2022). TCG EK Credential Profile, For TPM Family 2.0 Version 2.5 Revision 2. https://trustedcomputinggroup.org/resource/http-trustedcomputinggroup-org-wp-content-uploads-tcg-ek-credential-profile-v-2-5-r2_published-pdf/.
- Trusted Computing Group (2023a). TCG Algorithm Registry, Family 2.0 Level 00 Revision 01.34. <https://trustedcomputinggroup.org/resource/tcg-algorithm-registry/>.
- Trusted Computing Group (2023b). TCG PC Client Platform TPM Profile (PTP) Specification for TPM 2.0, Version 1.05. <https://trustedcomputinggroup.org/resource/pc-client-platform-tpm-profile-ptp-specification/>.
- Trusted Computing Group (TCG) (2003a). TCG Software Stack (TSS) Specification, Version 1.10 Golden. <https://trustedcomputinggroup.org/resource/tcg-software-stack-tss-specification/>.

- Trusted Computing Group (TCG) (2003b). TPM Main Specification Level 2 Version 1.2. <https://trustedcomputinggroup.org/resource/tpm-main-specification/>.
- Trusted Computing Group (TCG) (2006). TCG Software Stack (TSS), Specification Version 1.2, Level 1. <https://trustedcomputinggroup.org/resource/tcg-software-stack-tss-specification/>.
- Trusted Computing Group (TCG) (2009). TCG Certification Programs. <https://trustedcomputinggroup.org/membership/certification/>.
- Trusted Computing Platform Alliance (TCPA) (2002). TCPA Main Specification Version 1.1b. <https://trustedcomputinggroup.org/resource/tcpa-main-specification-version-1-1b/>.
- Winter, J. and Dietrich, K. (2013). A hijacker's guide to communication interfaces of the trusted platform module. *Computers & Mathematics with Applications*, 65(5):748–761. <https://doi.org/10.1016/j.camwa.2012.06.018>.

