CHAPTER 1

History of the TPM

A Trusted Platform Module, also known as a TPM, is a cryptographic coprocessor that is present on most commercial PCs and servers. In terms of being present in computers, TPMs are nearly ubiquitous, but until recently they've been mostly invisible to users due to lack of compelling applications that use them. That situation is rapidly changing. With the recent awarding of Federal Information Processing Standards (FIPS) certification to various TPM designs, and recommendations from the President's Council of Advisors that the United States government begin using TPMs to defend the nation's computers, the TPM has become a strategic asset for computer owners to defend their cryptographic assets. It is still true that very few people know enough about TPMs to use them in an advantageous manner, a situation that motivated the writing of this book. This chapter introduces you to TPMs, starting with TPM 1.1b, and describes the history of TPM 2.0's predecessors.

Why a TPM?

In the 1990s, it became increasingly obvious to people in the computer industry that the Internet was going to change the way personal computers were connected, and that commerce was going to move toward this environment. This immediately led to a realization that there was a need for increased security in personal computers. When PCs were first designed, little thought was given to their security, so the hardware did not support it. Additionally, software was designed without any thought to security—ease of use was the main driving force in software development.

The computer engineers who got together to develop the first TPMs—and who were part of what came to be known as the Trusted Computing Group (TCG)—were trying to reverse this trend and create a hardware anchor for PC system security on which secure systems could be built. Cost pressures dictated that any such solution had to be very cheap. The result was a hardware TPM chip intended to be physically attached to the motherboard of a PC. The TPM command set was architected to provide all functions necessary for its security use cases, detailed in Chapter 3; but anything not absolutely necessary was moved off chip to software, to keep the cost down. As a result, the specification was hard to read, because it wasn't obvious how commands were to be used when a lot of the logic was left to software. This started a trend in unreadability that has continued through all the updates to the specification.

Hardware security is not an easy topic to begin with, and it doesn't help to have a specification that is hard to understand. A good starting place to understand this technology is the history of its development.

History of Development of the TPM Specification from 1.1b to 1.2

The first widely deployed TPM was TPM 1.1b, which was released in 2003. Even at this early date, the basic functions of a TPM were available. These included key generation (limited to RSA keys), storage, secure authorization, and device-health attestation. Basic functionality to help guarantee privacy was available through the use of anonymous identity keys, based on certificates that could be provided with the TPM; owner authorization was required to create those identity keys. A new network entity called a *privacy certificate authority (CA)* was invented to provide a means to prove that a key generated in the TPM came from a real TPM without identifying which TPM it came from.

Areas of dynamic memory inside the TPM, called Platform Configuration Registers (PCRs), were reserved to maintain the integrity of a system's boot-sequence measurements. PCRs, together with identity keys, could be used to attest to the health of the system's boot sequence. This began the building of a secure architecture based on the TPM anchor, one of the key aims of the TCG.

One specific non-goal of the TCG was making the TPM design immune to physical attacks. Although such capabilities are possible, it was decided to leave physical protections to the manufacturers as an area where they could differentiate. Any software attacks are within scope for TPM-based security, however.

In a manner similar to smart-card chips attached to the motherboard, IBM PCs were the first to use TPMs (similar security coprocessors had been used in mainframe computers for decades). HP and Dell soon followed suit in their PCs, and by 2005 it was difficult to find a commercial PC that did not have a TPM.

One drawback of TPM 1.1b was incompatibilities at the hardware level. TPM vendors had slightly different interfaces, requiring different drivers, and package pinouts were not standardized. TPM 1.2 was developed from 2005–2009 and went through several releases. Its initial improvements over 1.1b included a standard software interface and a mostly standard package pinout.

The TCG realized that although TPM 1.1b protected keys against attackers who did not know a key's authorization password, there was no protection against an attacker trying one password after another in an attempt to guess a correct password. Attackers who do this usually try passwords from a dictionary of common passwords; this is known as a *dictionary attack*. The TPM 1.2 specification required that TPMs have protection against dictionary attacks.

Privacy groups complained about the lack of implementations of privacy CAs. This led to the inclusion in TPM 1.2 of new commands for a second method of anonymizing keys to help address this concerns—direct anonymous attestation (DAA)—and a method of delegating key authorization and administrative (owner-authorized) functions.

It turned out that shipping a machine with a certificate for its TPM's endorsement keys on the hard disk was in many case impractical, because IT organizations often erased the hard disk when they received it and installed their own software load. When they did so, the certificate was deleted. In order to provide a solution, a small amount of nonvolatile RAM (usually about 2KB) was added in TPM 1.2; it had specialized access controls along with a small number of monotonic counters.

The 1.1b specification had a means of copying migratable keys from one TPM to another TPM in case a machine died or needed to be upgraded. This process required the approval of the key owner and the TPM owner. It was designed with the assumption that an IT administrator would be the TPM owner and the user would be the key owner. But in 1.2, the user needed to be able to use the TPM owner authorization to mitigate dictionary attacks and create NVRAM, which made this design impractical. Therefore, in 1.2, another technique was designed to let users create keys that could only be migrated by a designated third party. Such keys could be certified to this effect and hence were called Certified Migratable Keys (CMKs).

Signing keys are often used to sign contracts, and having a timestamp of when the signing takes place is useful. The TCG considered putting a clock into the TPM, but the problem was that the TPM loses power whenever the PC is turned off. Although putting a battery in the TPM is possible, it is unlikely that the increased function would be worth the higher cost. Therefore the TPM was given the ability to synchronize an internal timer with an external clock and then to sign with the value of the internal timer. As an example, this combination could be used to determine when a contract was signed. This functionality also lets the TPM be used to distinguish how much time elapsed between two signature operations performed by the TPM.

No changes were made to existing application programming interfaces in 1.2, which preserved binary compatibility of software written to the 1.1b specification. A side effect was that the TPM 1.2 specification became even more complex, because special cases had to be used to maintain compatibility.

Before TPMs became ubiquitous, security coprocessors such as smart cards were used by some applications to store keys to identify users and keys to encrypt data at rest. TPMs are well equipped to take over this task. But they can do much more: because the security coprocessor is integrated onto the system's motherboard in the form of a TPM, it has additional uses (such as device identification) that are detailed in Chapter 3.

TPM 1.2 was deployed on most x86-based client PCs from 2005 on, began to appear on servers around 2008, and eventually appeared on most servers. Just having hardware does nothing, however—software needs to use it. In order to make use of the TPM hardware, Microsoft supplied a Windows driver, and IBM open sourced a Linux driver. Software began to be deployed, as described in Chapter 4.

How TPM 2.0 Developed from TPM 1.2

In early 2000, when the TCG was faced with the choice of a hash algorithm, it had two choices: MD5, which was most widely deployed; and SHA-1, which was stronger and was deployed widely, although not as widely as MD5. SHA-1 was the strongest commercial algorithm at the time and could feasibly be used in a small, cheap package. This was a fortunate choice, because MD5 weaknesses became apparent shortly afterward.

Around 2005, cryptographers published the first significant attack on the SHA-1 digest algorithm. The TPM 1.2 architecture relied heavily on the SHA-1 algorithm and had hard-coded SHA-1 everywhere. Although an analysis of the attack showed that it did not apply to the ways SHA-1 was used in the TPM, a common axiom in cryptography is that cryptographic algorithms only become weaker over time, never stronger. The TCG immediately began work on a TPM 2.0 specification that would be agile with respect to digest algorithms. That is, the specification would not hard-code SHA-1 or any other algorithm, but rather would incorporate an algorithm identifier that would permit design of a TPM using any algorithm without changing the specification. With this change (and other changes allowing all cryptographic algorithms to be agile), the TCG hoped the new specification could be the last major TPM specification to be released.

The original mandate of the TPM 2.0 Work Group within the TCG was only that: digest agility. However, even a cursory look at the TPM 2.0 specification shows that it's far more than TPM 1.2 plus an algorithm identifier. How did that happen?

TPM 1.1b had carefully crafted structures so that serialized versions (the structures translated into byte streams) were compact enough to be encrypted with a 2,048-bit RSA key in a single encryption. That meant there were only 2,048 bits (256 bytes) to work with. No symmetric encryption algorithms were required in the design, which kept the cost down and avoided problems when exporting designs that could do bulk symmetric encryption. RSA was the only required asymmetric algorithm, and performance required that structures be encrypted in one operation.

TPM 2.0 had to provide for digests that were larger than SHA-1's 20 bytes, so it was clear the existing structures were too large. An RSA key could not directly encrypt the serialized structures in one operation. Using multiple operations to encrypt the structures in blocks was impractical, because RSA operations are slow. Increasing the size of the RSA key would mean using key sizes that were not widely used in the industry and would also increase the cost, change the key structures, and slow the chip. Instead, the TPM Work Group decided the specification had to use the common practice of encrypting a symmetric key with the asymmetric key and the data with the symmetric key. Symmetric key operations are well suited for encrypting large byte streams, because they are much faster than asymmetric operations. Symmetric key encryption thus removed the barrier on the size of structures. This freed the specification developers to use their creativity to architect several functions that were distinct from TPM 1.2.

It is said that the best designs come about when the architects make a design and then, having learned all the problems they will encounter, throw away the first design and start over with a second. TPM 2.0 gave the architects the opportunity to do this. However, they still had to make sure that the opportunities for software development that were enabled by the TPM 1.2 design were not lost with the new architecture.

History of TPM 2.0 Specification Development

The specification made slow but steady progress for several years, with features being debated, added, and deleted. David Grawrock of Intel was the chair of the specification committee; under his leadership, the group selected the major features of the design and settled on a basic feature set and high-order design. At this point, the committee decided to change all the structures to allow for algorithm independence in the specification—this is called *algorithm agility*. All the authentication techniques were unified with

a technique originally called *generalized authorization* and now called *enhanced authorization* (EA). This increased the flexibility of authorization while simultaneously reducing the cost of the implementation and reducing the cognitive difficulty of understanding the specification. All objects and entities use the same authentication techniques. Many discussions took place regarding the problems created by having algorithm flexibility while still allowing a user to determine precisely what algorithms were used, both by a given key and also to protect the key, so the overall security of any key held by the TPM could be determined.

When Grawrock left the chairmanship due to changing responsibilities at Intel, Microsoft contributed a full-time editor, David Wooten, and HP took over the chairmanship. It was decided at this point that the specification should be compilable, which drove Wooten to create an emulator while writing the specification. A compilable specification has the advantage of much-reduced ambiguity: if there is doubt about how the specification is supposed to work, it can be compiled into an authoritative emulator. Additionally, the generalized authentication structure was moved from Polish notation (such as used in a TI calculator) to Reverse Polish notation (such as used in an HP calculator), which made implementing the specification easier (but made understanding the specification harder). The committee decided to add multiple key hierarchies to accommodate different user roles.

Wooten worked tirelessly to develop an implementation of the specification and provided strong leadership that drove the specification to its current feature set. When HP's Graeme Proudler stepped down from the chairmanship, David Challener of Johns Hopkins Applied Physics Laboratory formed a joint chairmanship first with Julian Hammersly of AMD, and later with Ari Singer of DMI. Kenneth Goldman (of IBM) took over the editorship from David Wooten after the first release, reprising a role he held for many years with the TPM 1.2 specification.

As new members joined the group over the years and began trying to understand the specification, some of them, notably Will Arthur and Kenneth Goldman, dove deep into the specification line by line. They submitted many bug and readability fixes to the TPM Work Group, and most of those resulted in changes to the specification that enhanced its consistency and readability. Even with these changes, it still is not easy reading, which led to the original impetus for this book.

Summary

The TPM specification has been developed twice. The first time, it developed from 1.1b to 1.2, evolving to incorporate capabilities as they came to be known to the specification committee. This feature-creep form of evolution made the final specification very complicated. In the second generation, TPM 2.0, after the cryptographic weaknesses of SHA-1 caused the need for a change, the architecture was redesigned from scratch—resulting in a much more integrated and unified design. The next chapter introduces the cryptographic concepts that will be used throughout the rest of the book. A good high-level understanding of these is imperative for you to understand TPM 2.0.