CHAPTER 1

Introduction to Trust and Intel® Trusted Execution Technology

Every contrivance of man, every tool, every instrument, every utensil, every article designed for use, of each and every kind, evolved from very simple beginnings.

—Robert Collier

Intel® Trusted Execution Technology (Intel® TXT) is a technology that uses enhanced processor architecture, special hardware, and associated firmware that enable certain Intel processors to provide the basis for many new innovations in secure computing. It is especially well suited for cloud computing and other uses where data integrity is paramount. Its primary goal is to establish an environment that is known to be trusted from the very start and further provide system software with the means to provide a more secure system and better protect data integrity. This is essential in that if the platform cannot be protected, then the data it will store or process cannot really be protected. At a minimum, this technology provides discrete integrity measurements that can prove or disprove a software component's integrity. These software components include, but are not limited to, code (such as BIOS, firmware, and operating system), platform and software configuration, events, state, status, and policies.

By providing a hardware-based security foundation rooted in the processor and chipset, Intel TXT provides greater protection for information that is used and stored on servers. A key aspect of that protection is the provision of an isolated execution environment and associated sections of memory where operations can be conducted on sensitive data, isolated from the rest of the system. Likewise, Intel TXT provides for sealed storage where sensitive data such as encryption keys can be securely kept to shield them from being compromised during an attack by malicious code. Attestation mechanisms verify that the system has correctly invoked Intel TXT to make sure that code is, in fact, executing in this protected environment.

This book describes how this technology can benefit the datacenter, and it specifically addresses the concerns of the cloud service provider and the cloud service client. The goals of this book are as follows:

- To explain what this technology does and its underlying principles
- To describe the roles and responsibilities for enabling and using this technology
- To guide the system administrator in establishing the right launch control policy
- To discuss how software (local and remote) can take advantage of this technology
- To look at some current and future innovations and how they apply to public and private clouds

Clearly these are important topics, so let's get started!

Why More Security?

Intel Trusted Execution Technology is relatively new for servers. It was initially developed for desktop and mobile computing platforms. It first debuted in servers in 2010 as attacks on datacenters started to escalate and calls for platform hardening formalized. For quite some time, server platforms were thought to be immune from attacks because they are typically kept in secure rooms and managed by highly skilled professionals. That is no longer sufficient. The frequency of attacks against datacenters is growing at astounding rates and those attacks are increasingly likely to have come from organized crime, corporate competitors, or sponsored by foreign governments with very sophisticated methodologies and deep resources.[1] Corporate executives worry that a breach could be very costly—both socially and economically, in the face of new regulations and stiffer penalties for failing to protect personal information.

Andy Grove, one of the Intel's founders, wrote the book *Only the Paranoid Survive* (Doubleday Business, 1996). This book is about recognizing inflection points and taking action. Server security is definitely at one of those inflection points. A successful attack could do significant damage to a company, regardless of whether that company is providing the service or using it—especially if adversaries can demonstrate that the company failed to use available security precautions. But it doesn't even take a successful attack. Just the failure to use available security precautions, or to make them available to your customers, could do irreparable harm.

Furthermore, this need for vigilance and caution doesn't only apply to business environments. The individual making online purchases, retail companies using cloud computing services, and cloud service providers all want more assurances that transactions and data are protected.

The answer is a higher level of security, better ability to mitigate attacks, and the means to prove those capabilities are enforced. Intel Trusted Execution Technology (or Intel TXT) can now be a significant part of that solution.

Types of Attacks

Bad people do bad things for all sorts of reasons. Some try to prevent use of the platform (denial of service, a.k.a. DoS attacks), some want to destroy or corrupt data, and others want to steal data. They will attack data in flight, at rest, and in use.

Data-in-flight refers to the transmission of data from one place to another. Using a secure channel that encrypts data during transport helps defend against in-flight attacks, but one also has to make sure that the recipient is the intended recipient and guard against replay attacks and the man-in-the-middle attacks. A replay attack is where an attacker intercepts a transmission and resends a copy of that transmission at a later time to fool the recipient into thinking it came from an authorized source portraying a real transaction. The man-in-the-middle attack is where an entity inserts itself into the communication link, forwarding transactions between the two endpoints, and thus gaining access to the privileged information. For this case, entity A starts a session with the attacker (the "man in the middle") thinking it is a session with entity B. The attacker then starts a secure session with the intended entity B, claiming it is entity A. The attacker is now able to steal or modify the data being transmitted between A and B.

Data-at-rest refers to the storage of data. Encrypting the data before storing on disk is a growing practice, but it also requires protection of the encryption keys. One answer is sealing of data, such as keys, so that it can only be used by authorized agents. But again, this raises the question of who to trust and how to prevent untrusted entities from gaining access.

[1]Example from the US Federal Bureau of Investigation (FBI), November 2011, http://www.fbi.gov/news/stories/2011/november/malware_110911.

Data-in-use refers to the data being manipulated by the application. Applications typically work on unencrypted data, so an attacker gaining access to that data circumvents any transport or storage protections. These attacks come from many different directions and target various components and operations. For example:

- *Frontal attack*: A direct attack on a system operation, generally by modifying or corrupting the system code or tricking the system.
- *Flanking*: *Reset attacks* have proven effective. This is where an attacker forces a reset after it has manipulated the BIOS to boot malicious software, which then inspects secrets and other privileged information that remain in memory.
- *Spying*: *Root kits* are an example of where an attacker causes the platform to boot malicious code that uses the platform's virtualization capabilities to then load the expected operating system in a false virtual environment. The system software is unaware that it is in a virtual environment and the root kit is now in control of the platform and has access to everything that the system places in memory.

A successful defense not only requires mechanisms to protect data, but also the means to detect changes and establish trust in the platform.

What Is Trust? How Can Hardware Help?

For most people, *trust* means that you have faith in someone or something to do the right thing. A broader definition would be "faith in someone or something to do something consistently." For instance, one might say that they "trust" that a computer virus will do harm—but they don't "trust" the virus. Generally, we trust the operating system to protect data and system resources, but we don't trust an operating system that has been infected with a virus or influenced by other malicious code. The challenge is to tell the difference.

Can we trust the system to determine if it is trustworthy? If you were to ask anyone if they can be trusted, they will likely respond "yes"—especially criminals. The same holds true for software, especially malicious software. To trust software, we need the ability to verify it in a way such that any change to the software changes its credential. For instance, if an operating system can prove that it is an authentic version and has not been modified, it deserves more trust than one that cannot. In fact, knowing that software has been modified is also valuable—because it allows corrective action such as quarantine and repair.

Thus we cannot depend on the software itself to detect if it has been modified (for example, infected with malicious code), because the malicious code might control or block the module that makes the trust decision. Thus we look to hardware to help make that determination.

The industry's leading security experts formed a nonprofit industry initiative known as the Trusted Computing Group[2] (TCG) to enhance trust and therefore security on computing platforms. The charter of the TCG is to develop industry standards for trusted computing building blocks and define their use for various platform types. Their efforts have produced a specification for a special security component (called a Trusted Platform Module, or TPM) and specifications for how to use that module in PCs and servers. These documents are available from the TCG web site (`www.trustedcomputinggroup.org/`). Several companies produce TPM chips that comply with these specifications, specifically for use on PCs and servers. The TPM chip is a very secure device that provides a number of security functions, and it is implemented in systems according to the TCG specifications by your favorite server manufacturers.

[2] `http://www.trustedcomputinggroup.org/`

A TPM chip is also one of the base components of Intel TXT, and provides the means to securely measure software components such as firmware, platform configuration, boot code, boot configuration, system code, system settings, and so on, to form a set of credentials for the platform components and system software. It also provides for accurately using those measurements as proof of trust and the ability to protect those measurements.

The measurement process for Intel TXT starts with hardware (that is, microcode designed into the Intel processor) and uses hardware (special chipset registers) to start the measurement process and store the measurements in the Trusted Platform Module chip in a way that cannot be spoofed by software.

What Is Intel® Trusted Execution Technology?

Intel TXT uses a combination of hardware, firmware, and software, and is built on and fully compliant with the Trusted Computing Group PC Client and Server specifications.

In general, Intel TXT is:

- A collection of security features.
- A means to verify that those features are in use.
- The means to securely measure and attest to system configuration and system code.
- A means to set policies to establish a trust level based on those measurements. Based on those policies, it is a means to:
  - Invoke enhanced capabilities (secure mode) for systems that meet that policy.
  - Prevent a system that fails the policy from entering the secure mode.
  - Determine if the system has entered the secure mode environment.
- The means for a trusted OS (that is, the system operating in the secure mode environment) to provide additional security features.

The ultimate goal of Intel TXT is to provide a more secure and hardened computing environment, but it is not a means to an end. Rather it is a tool for achieving higher levels of trust, and that trust does not come from simply enabling Intel TXT. The platform owner (e.g., datacenter IT manager) plays a key role in establishing what trust means, and Intel TXT provides the flexibility so that the system administrator can create a trust policy to match the requirements of the datacenter. Servers that implement Intel TXT can demonstrate (attest) that they comply with a specific trust policy, and thus can be used to form pools of trusted servers based on the established trust policy. Servers without Intel TXT and those that don't meet the trust policy can be excluded from the trusted pools. With the right policies in place, the datacenter can provide trusted pools of servers, allowing service clients to create "*use policies*" depending on the trust level. One example of this is the ability to confine critical applications (such as e-commerce services processing credit card information) to run only within a trusted pool. This has many applications for both private and public cloud providers, as well as their clients.

Before one can create a trust policy, it is important to understand what Intel TXT does and does not do. To put it in perspective, if we look at a bank's security, we find multiple security methods in use (locks on the doors, bars on the windows, a security alarm system, a surveillance system, a vault, armed guards, and so on). This is *defense in depth* and implies that no one method can do it all and that various methods come into play at different times (for example, doors and vaults tend to be unlocked and portions of the security alarm system are disabled during banking hours when armed guards are present). Of particular interest is to note that some of the methods are to prevent intrusion and others are just to detect and report it. The same is true for computer security; knowing when a breach has occurred is very important—just as a bank manager would not open the vault until he knows the bank is secure, Intel TXT can be configured to only allow the OS to launch if it knows the platform and system software are secure and trusted (as defined by the datacenter's policy). This *secure launch* allows the software to operate as a trusted OS, but only after validating that the platform configuration and system software meet the datacenter's policy.

So how do we define *secure* in this context? The short answer is to make sure that the system software is using all of the protection afforded by the processor and chipset architectures.

And how do we define *trusted?* The answer requires a means to measure the platform and the software. For servers, Intel TXT does that by incorporating two TCG concepts—a *static chain of trust* and a *dynamic chain of trust,* as illustrated in Figure 1-1. The static chain of trust measures the platform configuration, and the dynamic chain of trust measures the system software, software configuration, and software policies.

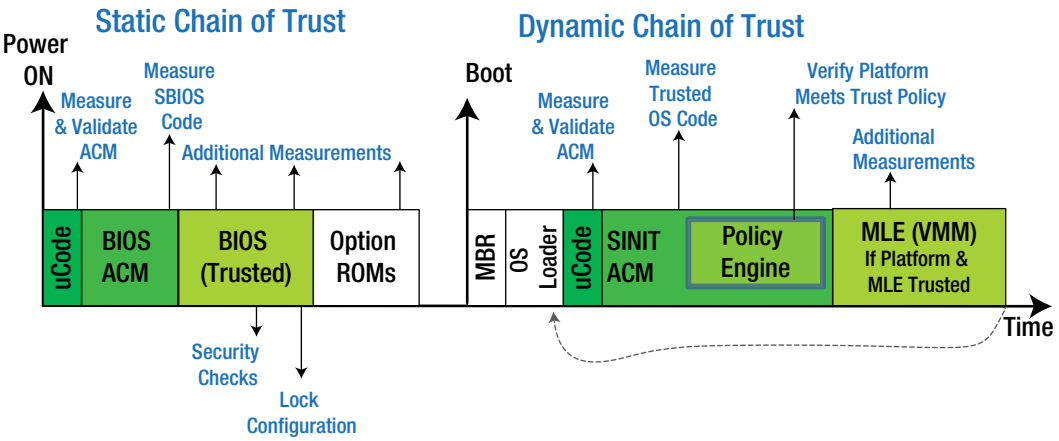

Figure 1-1. *Intel® TXT launch timeline with static and dynamic chain of trust*

Static Chain of Trust

The static chain of trust measurements start when the platform powers on. It starts with microcode that is embedded in the processor measuring an *authenticated code module* (ACM) provided by Intel. The ACM is a signed module (whose signature and integrity is authenticated by microcode) that performs special security functions. The ACM measures a portion of the BIOS code. That code then measures the remainder of BIOS code and configuration. Note that BIOS code does not execute until after it has been measured. These measurements are stored in special *platform configuration registers* (PCRs) inside a TPM (remember that the TPM is a very secure hardware device specified by the TCG). Different PCRs hold different launch component measurements (as specified by the TCG). During this process, the BIOS is required to turn on certain security features, and then call a function in the ACM that performs a security check. Before the BIOS executes any additional code, such as option ROMs on third-party adapters, it must lock the platform configuration by calling the ACM again, which performs additional security checks, and then locks certain platform and processor registers. The BIOS also measures the Master Boot Record (MBR) and the OS Loader when it boots the operating system.

The main takeaways are that measurements are started by hardware, measurements are protected from tampering by the TPM, and code is measured before it is executed.

These “static” measurements are done only once, each time the platform powers on. These measurements are referred to as *platform configuration* measurements.

Dynamic Chain of Trust

The dynamic chain of trust starts on request by the operating system (OS) via a special processor instruction, which measures and verifies another ACM (the SINIT ACM), which will oversee the secure launch. The SINIT ACM performs additional checks, which include making sure the BIOS passed its security checks and has locked the platform configuration. The ACM then measures the OS (actually a portion of the OS referred to as the *trusted OS*) and invokes

a *launch control policy* (LCP) engine that will determine if the platform configuration and OS will be trusted (as defined by the policy set by the system administrator).

A *trusted OS* (a term used by the TCG for system code that invoked and passed the secure launch process—proving it meets the launch control policy) is allowed access to additional (privileged) security functions, including using additional PCRs. The trusted OS can measure such things as additional code, data, and configuration into those PCRs, and use the content of any of the PCRs to make trust decisions and seal/unseal data. Applications (both on and off the platform) can also use those measurements to make trust decisions.

Virtualization

Intel TXT works equally well for both virtualized and nonvirtualized platforms. In the course of bringing Intel TXT to market, the feedback has been quite strong that the primary and most compelling usage model for Intel TXT on servers is with virtualized and cloud-based server systems. This is generally because it addresses some of the key challenges introduced by moving to shared infrastructures. Specifically, it provides the ability to better control applications (in the form of virtual machines) and migration of applications among available platforms (the cloud) based on host trustworthiness. In other words, high availability and optimum utilization is achieved by migrating applications to available servers while still maintaining a trusted environment.

The term *OS* for a server platform can be confusing because of the existence of multiple operating systems on a virtualized host platform. This book uses the term *host OS* to refer to a hypervisor, virtual machine monitor (VMM), or in the case of a nonvirtualized platform, the traditional bare-metal operating system or any other application or utility that boots first. In any case, the host OS is the first system control program to boot. This is in contrast to an OS instantiated in a virtual machine (VM), which will be referred to as a *guest OS*. Furthermore, a trusted OS is a host OS that has successfully performed a *secure launch*. As a side note, while it is possible for the host OS to provide the same type of secure launch to its guest operating systems, that capability is outside the scope of the current version of Intel TXT and not discussed in this book.

OK, that last paragraph can be a little hard to read. So let's try these definitions:

- *OS*: The system software that manages platform resources and schedules applications. This includes
  - A hypervisor or virtual machine monitor that manages the *virtual machines*.
  - An OS that executes in a VM (i.e., the guest OS).
  - An OS executing on a nonvirtualized platform.
- *Host OS*: An OS that is not executing in a virtual machine, which can perform a secure launch, and thus can be measured by hardware (i.e., Intel TXT).
- *Guest OS*: An OS that is executing in a virtual machine, which is not measured by hardware.
- *Trusted OS*: A host OS that has performed a measured launch and passed the datacenter policy.

Measured Launch Environment

Because *trust* is a subjective term, Intel refers to the trusted OS as the *measured launch environment* (MLE), because Intel TXT certifies the measurement of the trusted OS (not its trust) and enforces the platform owner's policy—that is, it has measured the OS and platform configuration and verified that they meet the platform owner's launch policy. The measurements (i.e., the values in the PCRs) can be used by any entity to make a trust decision. For example, the platform owner can specify multiple OS measurements and platform configurations that will pass its launch policy, but an application can be more restrictive and trust only a subset. As we will see later, this concept can be extended and is the basis of attestation, which will be explained in detail later in this book.

The host OS is allowed to enter and exit the measured launch environment without having to reboot the platform. To exit, the host OS simply invokes another special processor instruction that will reset the PCRs that were used to measure the launch and those that were set by the trusted OS. Of course, this action also curtails the host operating system's access to some of the additional security resources. Reentering the secure environment restarts the dynamic chain of the trust process, and thus recalculates the MLE measurements storing them in those PCRs, allowing a different trusted OS (or different configuration of the same OS) to execute with its own set of trust credentials.

Finding Value in Trust

One of the nice things about Intel TXT is that its value can be appreciated by a number of different entities for various purposes. Some of these values have already been realized and there are many more to come.

Cloud Computing

Looking at a typical cloud management model, as illustrated in Figure 1-2, the cloud service provider maintains a pool of application servers, which hosts various applications provided by the cloud service clients. These applications are scheduled to run at times and locations to meet both provider and client polices.

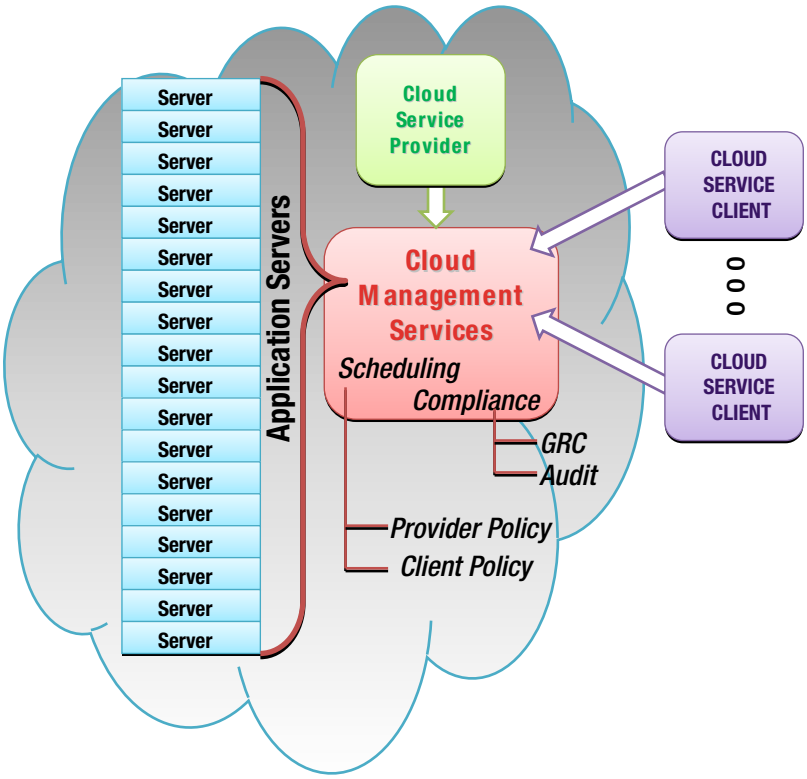

Figure 1-2. *Cloud management model*

The service provider must be able to demonstrate that the cloud meets governing regulations, and provide an audit trail to assure the client that all client and government requirements have been meet.

Because Intel TXT provides measurements of platform configurations and operating systems, adding Intel TXT to the picture allows the trust status of the application servers to enter into the equation. It allows both the service provider and the service client the ability to establish trust policies based on Intel TXT measurements and to identify servers that meet those polices, as illustrated in Figure 1-3.

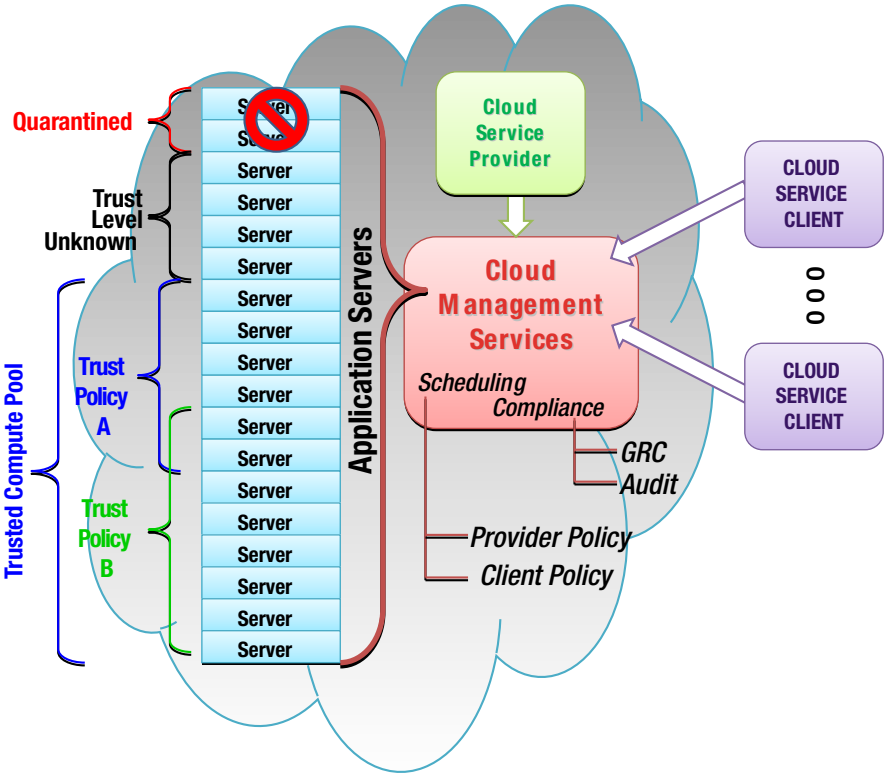

Figure 1-3. *Cloud management trust model*

Up till now, we have talked about measuring components, but the measurements themselves don't prove anything, or do they? Actually, it is the combination of the components that are measured, how the measurements are made, how the measurements are stored, and how the measurements are accessed that provides the basis for attestation.

Attestation: The Founding Principle

Furthering our discussion of how a bank uses multiple security concepts, let's focus on the banking transaction. Before executing a transaction, bank tellers require proof of the customer's identity. In the United States, that would typically be a driver's license or some other government-issued identification. State-issued driver's licenses and IDs have evolved over the years. In 1960, a license was simply a piece of paper printed with the state seal and folded in half to about the size of a business card. Personal information such as date of birth, height, weight, color of eyes, and hair color were typed in the respective boxes on that form by the issuing agency. Needless to say, it was fairly easy to modify, and thus subject to abuse. To reduce misuse and fraud, states started laminating each license with a photograph, thus making it harder to alter the data, and the photograph provided better identification. Of course, this assumes that the state issues the license to the right person. After the 9/11 attack, the US federal government

placed more requirements on states—both on how the ID is created (including holographic images and special watermarks) and how the state validates the identity of the person being issued the license.

Just as a bank teller has to be able to verify the customer's identity, software needs a means to identify its environment. A driver's license is simply a certificate of attestation, and the same principles apply to validating software. One way to look at it is that the description on a license along with the photograph constitutes measurements of the person. That certificate also identifies who issued the certificate, and has features used to detect if it has been altered.

For Intel TXT, *attestation* is the ability to securely verify a set of component measurements. It is the founding principal for Intel TXT and accomplished by performing a cryptographic hash of each component. These components include BIOS code, BIOS configuration settings, I/O adapter option ROM code, I/O adapter configuration, MBR, bootstrap code, boot settings, ACM code, trust policies, OS code, OS configuration, and anything else that the OS wishes to measure. By measuring components using a cryptographic hash, corrupted components are easily detected via a change in their measurements.

But attestation involves more than just making the measurements. Those measurements constitute a certificate, and Intel TXT has to guarantee the following:

- The certificate comes from a trusted source
- The certificate is accurate
- The certificate has not been modified (at the source or in transit)

Value to System Software

You might ask why you need Intel TXT if you already trust the OS, since the processors and chipsets already provide a number of security features that the OS uses to protect itself against attacks, and you have installed virus protection and other security software. The first realization comes from the ability for the OS to use those functions, as well as load additional security software only after the OS loads. So what happens when the OS is not running? What about attacks against the relatively unprotected BIOS and the pre-boot environment? The following are some of the benefits that Intel TXT provides:

- Protection against reset attacks
- Protection against root kits
- Sealing data
- The ability to provide more secure services to VMs and applications
- System credentials

■ **Note** Processors and chipsets provide a number of security features that an OS uses to protect itself from unintentional or malicious modification while it is running. Intel TXT extends that protection to when the OS is not running. That is, the OS has to start out uncorrupted for it to remain uncorrupted, and Intel TXT helps prove that the OS starts up uncorrupted.

Intel TXT makes it possible to detect unauthorized changes to BIOS and BIOS configuration, changes to the boot sequence, and changes to the OS and OS configuration. More important is that it provides an attestation mechanism that can be used by any entity that wishes to make a trust decision.

Cloud Service Provider/Cloud Service Client

Intel TXT's attestation opens up a whole new world of possibilities that can be realized by both local and remote entities. In addition to securely measuring and maintaining platform/OS measurements, Intel TXT provides the means to securely convey those measurements to external entities. Intel TXT also provides the means to determine if a platform is operating in the secure mode and for determining the policy for allowing secure mode. The datacenter sets the bar establishing the minimum trust level by setting the launch policy. Later, we take a close look at just what that means and how it is done, and discuss considerations in selecting a policy.

Management software can use this to provide capabilities such as:

- The ability to quarantine platforms with unauthorized modifications (that is, platforms that don't match "known good" configurations).
- Trusted compute pools and trust policy–based migration.
- Additional trust and security control metrics such as geo-tagging.
- Enhanced audit trail and compliance support.

Figure 1-3 illustrates how platforms can be categorized based on Intel TXT measurements. A trusted compute pool consists of those servers that have demonstrated that they have a known good configuration. This excludes those with an unknown trust level (those that don't support Intel TXT or don't have it enabled) and, of course, those that fail the Intel TXT launch control policy. The trusted compute pool can be further categorized into pools of trust based on various trust policies set by the service provider and service client.

As we continue to unravel the mysteries of Intel TXT, we will discover that the platform provides for 24 different secure measurements. Some are defined for specific purposes and others allow innovation. If one of the measurements included a geographic location identifier (geo-tag), policies could then limit applications to specified countries, states, or regions. This is especially important because a growing number of privacy laws and other governmental regulations stipulate controls and protections that vary depending on location or restrict data to certain geographical boundaries.

These capabilities are valuable for both public and private clouds. Even within a private cloud, different workloads have different security requirements—just as different workloads are assigned to different classes of servers in a traditional nonvirtualized datacenter. The ability to prove trust and maintain trust levels allows the cloud to host more restrictive applications, such as a finance or human resources department workloads that would no longer need separate compute facilities. These capabilities can give public cloud service providers the ability to enforce and charge according to the security level required. They give the public cloud service clients the tools to specify required trust levels and geographical boundaries, as well as the means to validate compliance to those requirements.

Before we discuss these capabilities—and they will be covered in more depth in subsequent chapters—we need to understand more about how Intel TXT works and from where the trust comes. As we progress through this book, we will answer the following questions:

- How does Intel TXT provide attestation, and what makes it so powerful?
- How does the system administrator enable it?
- How does the datacenter take advantage of it?
- How does system software use it?
- How do others make use of it?

Then we will take a closer look at attestation and how it is currently used. To get a glimpse of the future, we will also discuss concepts that are being evaluated and prototyped.

What Intel TXT Does *Not* Do

Intel TXT does not measure trust, nor does it define levels of trust. These are subjective terms whose definitions will change over time. Rather, Intel TXT allows the datacenter/cloud service provider, cloud service client, OS, and other entities to make their own trust decisions with a new set of robust credentials. These trust decisions can be, but do not have to be, based solely on Intel TXT. In fact, since *defense in depth* refers to using multiple methods to protect your assets, Intel TXT assumes that other security mechanisms exist, new technologies will emerge, and together they will provide greater coverage. Intel TXT is flexible enough that it can be used to help verify proper utilization of other methods and their policies.

Intel TXT does not monitor runtime configuration. It only performs measurements at specific events, such as *Power On*, and upon request by the host OS to do a secure launch. Therefore, it does not degrade performance since it does not steal cycles from the operating system or applications running on the platform, nor does it consume memory bandwidth and other operational resources after the secure launch.

Currently, Intel TXT only measures the host OS or VMM and does not provide for secure measurements of a guest OS or its applications. This is a topic of discussion and interest throughout the industry, and it would be very surprising if this capability is not added in the future.

Enhancements for Servers

As mentioned at the beginning, Intel TXT was first developed for client platforms (desktop and mobile). So is it the same technology? There are some differences, primarily because servers are more complex than client machines. One of the most prominent differences is the set of server features known as Reliability, Availability, and Serviceability (RAS). The complexity of server RAS features such as memory mirroring, memory sparing, and error handling require capabilities that are very platform-specific, and thus must be performed by platform-specific code. This code must be trusted; in particular, the BIOS code that executes after a platform reset (to mitigate the reset attack) and the System Management Module (SMM) code, which may need to change memory configuration and/or platform configuration while the system is in operation.

This changes what the TCG refers to as the TCB (Trusted Computing Base), which, by definition, is the minimum amount of code that needs to be trusted for a platform. For server platforms, the TCB includes the processor microcode and the ACM, whose signature and integrity are checked before the ACM is allowed to execute. It must also include the BIOS (or at least a portion of the BIOS).

Including BIOS in the TCB

For client platforms, the BIOS does not need to be trusted to clear memory in defense of a reset attack, because memory cleaning is performed by the ACM before allowing the BIOS to access the memory. This is not possible for servers given the complexity of server memory configurations and RAS features. Thus in response to (or to detect) a reset attack, BIOS code integrity must be verified before the BIOS can be trusted to clean secrets from memory. Typically, this is transparent to the OS and end user, but it does have implications on how BIOS upgrades occur. For instance, upgrading BIOS causes its measurement to change, and if there is a reset attack after a BIOS update, the BIOS would not be trusted. To overcome this problem, Intel TXT allows for a signed BIOS policy that allows the ACM to validate that the BIOS is still trusted using signature verification.

Processor-Based CRTM

Because the BIOS must be trusted to clear memory after a reset attack, a trust decision must be made before BIOS is allowed to execute. This means that the BIOS has to be measured (or at least the portion of it that clears the memory) and that measurement has to be verified. This adds complexity to the BIOS because it must now provide a table that identifies the code that will be executed to clear memory and also specify the type of policy that the hardware uses to validate that code. Unless the verification fails, this will be transparent to the user and the software.

Trusting the SMM

Client platforms with Intel TXT can use an SMI Transfer Monitor (STM) to prevent SMM code from corrupting the platform configuration. The complexity of servers and the need for the SMM to handle RAS concepts such as memory and processor hot plug (a common term for the removal or replacement of these components while a server is operating) require that the SMM code be inside the trust domain. This means that the SMM code has to be measured as part of the BIOS code measurement, and thus included in the trust decision to allow the OS to perform a secure launch. Again, this is transparent to the OS and the end user, but it will show up in the logs that describe what is included in the BIOS measurement.

Other Differences

As alluded to previously, the complexity of Intel TXT increases on servers because of the server architecture complexity. This includes multiple processor sockets, enhanced I/O, base board management, and so on. Whereas these features make it harder on the BIOS developer, they tend to be transparent to the end user and the software using Intel TXT.

Impact of the Differences

The bottom line is that any operating system that functions with the client version of Intel TXT also functions with the server version. The platform credentials will be different, but each platform type will have a different measurement anyway. The OS measurements are performed exactly the same way for client and server platforms.

Even though the usage models differ greatly between client and server platforms, their Intel TXT behaviors have intentionally been kept consistent. This allows client operating systems to be used on server platforms and server operating systems to be used on client platforms—where this would be desirable.

Because the OS launches exactly the same and the integrity of the measurements does not change, applications using Intel TXT attestation work the same.

Roles and Responsibilities

So you have an Intel TXT–enabled platform—now what? What are your responsibilities? What precautions do you need to take? What about system software? What else is needed? Let's take a quick look at what is expected of all of the players. These roles and responsibilities will be explored in greater detail in subsequent chapters of this book.

OEM

The OEM puts together all of the components, and then packages them into an Intel TXT–capable platform. This design is thoroughly tested to assure that the platform complies with Intel TXT requirements and is able to perform a secure launch. The OEM often does the initial provisioning of the TPM before the platform ships. With the exception of occasional BIOS updates, the OEM's job is done once the platform leaves the factory.

Platform Owner

The platform arrives with Intel TXT and the TPM disabled. This is to prevent rogue software from taking over and setting its own trust policies, or launching a denial of service attack by consuming all of the TPM resources. What needs to be done next depends on the host OS that you are installing.

Before installing an Intel TXT–enabled OS, you have to enable the TPM and Intel TXT via the BIOS setup menus. Depending on the OS, you may need to establish TPM ownership and set the launch control policy (or the OS might do that for you). We will discuss how to do this later, as well as discuss tradeoffs for selecting a launch control policy.

Host Operating System

Operating system installation detects if a TPM is enabled and whether TPM ownership has been established. As mentioned earlier, the OS might require exclusive TPM ownership, and thus the OS performs the TPM "take ownership" operation itself, as well as sets the launch control policy. If so, it will do this as part of the install process on first boot. For this case, the OS provides the utilities for the platform owner to modify the policy.

Each time the host OS boots, it detects if Intel TXT is enabled, and if so, performs a secure launch. As a trusted OS, it continues measuring system code, configuration, and other entities into the platform configuration registers reserved for the OS. The host OS maintains a log that indicates what has been measured into each register.

The host OS provides remote access to certain TPM resources (such as platform configuration registers) and associated logs. This allows external management applications to make trust-based decisions using the Intel TXT measurements.

Other Software

As we progress through this book, we will see the various roles that are played by third-party software. We will discuss attestation services and trust authorities, and their importance to both the cloud service provider and the cloud service client.

There are no hard and fast rules for using Intel TXT attestation. Typically, third-party software uses Intel TXT measurements in the platform configuration registers to verify which host OS is in control (and which version). Knowing which host OS then provides insight into what the trusted OS has measured into the platform configuration registers reserved for the trusted OS.

For example, one application could use certain PCRs to verify which OS is in control and its version. Based on that information, another application could verify OS-specific values, such as geographic location measured into one of the platform configuration registers used by that OS. Other applications can use that information to make trust-based policy decisions.