Establishing Security with Trusted Execution Technology Architecture

ABSTRACT: A key component in trusted computing is the Trusted Execution Technology architecture. By defining protocols for communication between partitions as well as application execution, Trusted Execution Technology can help ensure that security is enforced. As with all standards, though, Trusted Execution Technology only provides the protocols and framework to fulfill trust requirements. The responsibility lies with the creators and users of the system to properly implement and use the features inherent Trusted Execution Technology.

"Architecture starts when you carefully put two bricks together. There it begins." —Ludwig Mies van der Rohe

The architecture of Trusted Execution Technology (TXT) starts putting together the bricks available on the platform. The requirement is to protect software and use domain separation to provide the protection. Figure 1 shows the basic picture of TXT.

The architecture has three main features:
- The standard partition or the left-hand side (LHS), of the diagram is an operating environment such as one sees in non-TXT platforms. This part is the execution area of today.
- The protected partition or the right-hand side (RHS), of the diagram is the protected environment. This area is under the protection of TXT. All applications running in the protected partition have protection from software attack.
- The Measured Virtual Machine Manager (MLE) provides the controlling entity that manages all partitions.
The protected partition and the MLE reside in protected memory.

The Naming of the Sides

In the early days of Trusted Execution Technology project, a small task force of security architects was working on the design of TXT. The groups output was going to be a specification that defined TXT. The name of the specification was going to be something like “TXT Security Architect Specification.” As that moniker was much too large, the acronym of TXT SAS appeared which was finally shortened to SAS. The series of meetings to work through the various sections of the document are known as the SAS meetings.

The SAS continues to meet, albeit with a larger group now, defining the security properties of TXT. When the SAS started on the task of defining what worked with what with whom and when, it became very apparent that some sort of naming convention was necessary. As the conversation went back and forth, covering items like protected or unprotected, known and unknown, secure and insecure—they all got mixed up. Someone would ask, “Where is the item at any point in time?” The diagram in Figure 1 was up on the white board constantly, and one of us would point to the left or right side and say “there.” Figure 1 soon became “standardized” in that the standard partition was on the left side and the protected partition was on the right. The shorthand soon became to say simply left or right, and everyone in the room knew whether or not the information was under protection.

This shorthand worked very well for the security architects in the SAS. In fact, as the architects talked with others working on trusted computing projects, the terms left-hand side and right-hand side became the standard. For those working on these projects for any length of time, left-hand side always conjures up the image of something not under the protection of the platform, while conversely right-hand side always implies something receiving protection. The terms were so prevalent that the definition of the acronym of Left Hand Side (LHS) and Right Hand Side (RHS) are in the specifications.

Not surprisingly, marketing people feel that the terms LHS and RHS are not appropriate to use for a major new technology. Without the background of Figure 1, LHS and RHS do not really impart any information to a listener. When I gave the first presentation of TXT at the Intel® Developer Forum (IDF), marketing asked me not to use LHS and RHS as the terms were just too esoteric. The LHS became the standard partition and the RHS became the protected partition. While the terms are much more descriptive of the type of partition, they do not have the ease of use the LHS and RHS do. Having briefed many people on TXT, I still use LHS and RHS. When I communicate with other security architects who work for OS vendors or other hardware manufacturers, we continue to use LHS and
RHS. I am going to attempt to keep the marketing folks happy in this article and use the approved names, standard and protected. The reader must know though that when they hear or see someone talking about the LHS or RHS, the speaker is referring to Figure 1.

**Actual Use**

A funny thing happened with the architecture diagram: everyone realized that the diagram was really incomplete. One must come to grips with two realities when looking at Figure 1.

First, the diagram does not actually represent what most MLE vendors are likely to implement. Intel® Virtualization Technology allows for an almost unlimited number of partitions. Why should TXT limit the possibilities to only two? The answer is that TXT does not limit the number of partitions; the MLE can have one, two, or twenty partitions.

The second is that the difference between protected partition and unprotected partition is arbitrary. All partitions automatically have protection between each other that is a function of Intel® Virtualization Technology. The real protection attribute comes from individual page protection that prohibits DMA access. Certainly, the MLE would normally provide protection for itself from DMA, but individual partitions could easily have a mixture of protected and unprotected pages.

With these two issues, it becomes very difficult to differentiate between the partitions again. So, for the sake of sanity, the terms protected partition and unprotected partition continue to be used throughout this article. However, the reader is admonished to remember that the designation is only one of convenience and real partitions have a mixture of protected and unprotected physical pages.

**Measured Launched Environment**

The MLE is the controlling software in a TXT system. The MLE provides:
- Memory arbitration
- Resource assignment
- Communication channel
- Partition lifecycle

You must remember as we discuss the MLE that the MLE is just a Measured Launched Environment. With the assumption that the “normal” MLE is a Virtual Machine Monitor (VMM), the basic capabilities and responsibilities of a VMM do
not change when the VMM becomes a MLE.

Memory Arbitration

The MLE provides control of all memory and resources in the system. If an application needs access to a physical memory page, the application uses a memory address. Using the page table entries, the CPU evaluates the address and converts the virtual address into a physical address. The setup of the page table entries is under complete control of the MLE.

Memory arbitration is the basic building block for guest partition isolation. Ensuring that each and every memory access complies with the policy set by the MLE, and that the CPU enforces the memory access policy, is the crux of MLE protection capabilities.

Memory arbitration is an inherent feature of any VMM. The MLE adds no real additional properties to the memory arbitration feature of the VMM.

Resource Assignment

The MLE controls, in addition to physical memory, all resources of the platform. Disk drives, display adapters, USB devices, and any other device one uses on the platform are all under control of the MLE.

All memory-mapped resources are very easy for the MLE to control; they appear as memory addresses, so the MLE treats the address as a protected page. The entire MLE page controls map onto the controls for the resource.

Resources that are not memory mapped require specific support from the MLE. The MLE must be aware of how entities communicate with the resource and then the MLE must be able to intercept, or virtualize, the access.

Communication Channel

Guests are going to want to talk with each other. While it is certainly possible to simply require guests to have an Internet connection and use the Internet for inter-guest communication, the reality is that direct guest-to-guest communication is mandatory. The MLE has complete control of how each guest communicates with any other guest.

Partition Lifecycle

A major feature of the MLE is the ability to launch and terminate a guest partition. The MLE contains either explicitly as a standalone entity or implicitly as
part of the code, the policy on when and how to launch and terminate guest partitions. Entities wishing to rely on the MLE, and its guest partition lifecycle management policy, must have the ability to obtain validation of the currently enforced policy. For MLEs that have the policy implicitly embedded in the code, attesting to the MLE identity also confirms the guest partition lifecycle management policy. For MLE that have explicit policy statements, the MLE must be able to show which policy the MLE is enforcing and how the MLE loaded the policy.

Standard Partition

Defining the LHS as a standard partition implies that the LHS is the normal situation that a user expects. This is correct. The standard partition does run standard operating systems and applications. The OS and applications need no modification to run in the standard partition.

The standard partition provides protection from software attack. Information held in the standard partition has the exact same areas of exposure that the information does today on systems without TXT. If the application designer wants or needs protections, they will have to use some component that resides in the protected partition. The next section, “Protected Partition,” explains how to design an application that takes advantage of the protected partition.

Operating System

The OS is any standard operating system like Windows† XP, Linux, or a home-grown OS. The specification places no restrictions the type of OS that runs in the standard partition. If the user runs a particular OS today on their platform based on Intel architecture, the OS would still run in the standard partition. Certainly, if the OS in question is very old, its ability to run using new hardware may be questionable, but not due to TXT. The OS probably would not understand the newer hardware.

The OS does not need any knowledge of the MLE. Certainly if one were to run DOS 3.1, the OS would have no knowledge of the MLE. DOS 3.1 would run correctly, but a direct knowledge of the MLE would be impossible.

The OS can be aware that the MLE and protected partitions are operating and available. A new OS can understand the hardware nature of a TXT platform and provide the new services that the hardware allows.

Application
In the standard partition, you can run a normal application such as Quicken† or a user-written application. The application makes use of the OS services and executes without any knowledge of the MLE or the protected partition.

For applications running on an OS that has no knowledge of the MLE or protected partition, the application’s ability to use the protected partition is limited. Unless the protected partition provides a communication mechanism that is publicly available and discoverable by the application, the application is unable to use any services of the protected partition.

If the OS is aware of the MLE and protected partition, then the OS can expose to applications the services exposed by the MLE and protected partition.

**Protection Partition**

All software in the protected partition receives protection from software attack, with no difference in the protection whether the application is running at ring 0 or ring 3.

**Kernel**

The kernel provides the services for applications running at ring 3. No requirement states that a protected partition must use both ring 0 and ring 3. It is entirely possible to create a domain that only uses ring 0.

The kernel can provide a rich set of services or it can provide limited services. When more code is in the kernel, it becomes harder to evaluate the security properties of the kernel. From the standpoint of security, the kernel should be as small as possible. From a service standpoint, the kernel should provide a wide and deep set of services. Since these two views conflict with each other, the kernel designer must weigh these two issues and make the appropriate choice for the use model.

**Rich Service Kernel**

One way to provide a rich service kernel would be to use a current OS like Linux. While this certainly provides a rich set of services, most existing operating systems are very large and difficult to evaluate. Nothing in the TXT architecture prevents someone from using an OS as a kernel in the protected partition. The ability to show the properties of the OS is very difficult.

A rich service kernel is possible and desirable when the use model allows for user written applets that provide many user services. Carefully creating and adding the
necessary services to the kernel can result in a rich kernel with services that enable a rich set of applications.

**Limited Services**

A limited service kernel can be either general purpose or designed to support a special application, which would seem to be an oxymoron. However, by providing a controlled set of services, the limited kernel actually provides a better trust boundary. The rich service kernel described earlier provides lots of services that a special purpose application would not need. By providing only the services necessary for the application, the amount of code that requires validation is smaller.

Another way to provide limited services and keep the kernel small would be to select a set of services that the majority of applications relying on the kernel would need. For instance, providing memory management, trusted input, trusted output, and general thread handling could possibly provide services for a wide variety of applications. While the extra special fancy stuff may not be available, the ability to create trustable applications may be greater.

**Applet**

The term applet is a very specific label. The idea is that applications using ring 3 in the protected partition are not the rich applications that one sees in ring 3 for the standard partition. Protected partition applications are applets: small, single purpose, and capable of being evaluated.

Take an application that does online stock transactions. Most of the application could be a ring 3 standard partition application—that is, parts that display the choices, allow the user to determine what to buy, and create the buy order. The only portion of the application that must be in the protected partition is the component that allows the user to verify and digitally sign the order request. This type of requirement defines an applet very well. The applet does four things:

- Receives XML string
- Displays XML string using trusted output
- Receives user OK through trusted input
- Signs XML string using a digital signing key protected by the trusted platform

The application then uses a simple applet and the applet only requires a limited service kernel. While it is not the only use of an applet and kernel combination, this use does represent a valid combination.

**Application**
There is no requirement that entities in the protected partition must be applets. Applications are possible also. The difference between an application and applet is that an application is standalone not requiring any outside help, while the applet is a component of a larger application.

Applications will come in all of the varieties that are available today. Big large applications with 1,000’s of options and abilities and small single-purpose applications that do their job quickly and efficiently are all possibilities.

Partition Communication

No direct communication occurs between any ring 0 or ring 3 process in the standard partition and any ring 0 or ring 3 process in the protected partition. The communication vector is the MLE, which has the responsibility to expose interfaces to the standard and protected partitions such that the two partitions can communicate. The MLE is under no obligation to provide the communication path, and if it does not, the partitions must communicate using some outside path such as a network or files.

If the MLE does provide a communication path, it can take any form that the MLE wants to provide, using a standard protocol or creating a special protocol.

Communication between two processes normally occurs as an InterProcess Communication (IPC) or a Remote Procedure Call (RPC). The mechanism that an MLE could expose is closer to an RPC than an IPC.

IPC

The IPC mechanisms normally take into account the use of internal resources to make the transfer faster. The activity of two processes mingling some internal resources is the exact type of behavior that a trusted platform should prevent. It is possible for a MLE to create mechanisms that look like IPC interfaces but that do not mingle process resources. If the MLE wants to implement pipes, the MLE could create a system that allowed a partition to establish a pipe to another partition. The actual implementation of the pipe would be a pipe from the establishing partition to the MLE and then an MLE-managed pipe from the MLE to the destination partition.

The creation of a direct pipe between two partitions would be illegal.

Partition communication
The previous sentence highlights one of the main learning’s over the years with trusted platforms and TXT. The sentence, as originally written, reflects a view of the world that TXT controls all. While the sentiment is nice, the reality is that TXT only measures the MLE and the MLE provides control of the platform. While the architects of TXT can make assumptions about what the MLE will do, there is no guarantee that a MLE perform a certain way. In addition, consider a MLE that is creating two partitions that need a fast channel between the two partitions. A direct pipe might be the exact answer necessary for that use model. Calling that pipe illegal is attempting to force a set policy onto the users of the trusted platform and does not represent the reality of how the real uses of the platform.

With the new understanding of the promises made by TXT and the MLE, the above sentence becomes, “Any communication between partitions must be established and monitored by the MLE”. Note now that the MLE sets the rules for how to establish and monitor. The job of TXT is to measure the MLE such that outside entities can identify the MLE and know the policies the MLE enforces for partition communication.

For more information about trusted computing, please refer to the book *Dynamics of a Trusted Platform* by David Grawrock.

**About the Author**

**David Grawrock** is a Senior Principal Engineer and Security Architect at Intel. He serves as Chair of the Trusted Computing Group (TCG) Trusted Platform Module work group and is the Intel representative to the TCG Technical Committee. With 29 years in the computer industry, David holds 10 patents, and has held lead technical positions with Central Point Software, Symantec, and Lotus Development Corporation.

Copyright © 2008 Intel Corporation. All rights reserved.

This article is based on material found in book *Dynamics of a Trusted Platform* by David Grawrock. Visit the Intel Press web site to learn more about this book: [http://www.intel.com/intelpress/sum_secc2.htm](http://www.intel.com/intelpress/sum_secc2.htm)

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center,