第5章 可信PC

可信PC是最早开发，并已经得到实际应用的可信计算平台。在中国国家标准GB/T 29827—2013《信息安全技术 可信计算规范 可信平台主板功能接口》中，根据TPM是否早于PC主CPU先启动，TPM是否能对PC Boot ROM引导代码先进行可信度量，将可信PC分为兼容型和增强型可信PC。本章主要介绍可信PC的系统结构、中国可信PC实现实例和操作系统安全增强。

5.1 可信PC的系统结构

兼容型可信PC通常在普通PC的主板上利用已有总线接口，集成TPM，对主板硬件结构基本上不做调整。增强型可信PC需对主板硬件结构进行一定调整，以实现TPM早于PC主CPU先启动，完成对PC Boot ROM(BIOS/EFI)启动代码的完整性度量。兼容型和增强型可信PC在系统结构上有所差异。

5.1.1 兼容型可信PC的主板结构

TCG在其发布的可信PC规范里提出了兼容型可信PC的组成结构。兼容型可信PC的主要特征是在主板上嵌有可信构建模块(Trusted Building Block，TBB)，它包括可信度量根核(Core Root of Trust for Measurement，CRTM)和TPM，以及它们与主板之间的连接，如图5-1所示。基于TBB和信任链机制，就可实现可信性的度量、存储、报告机制，确保系统重要资源的完整性和平台的安全可信，为用户提供可信服务。在兼容型可信PC里，TPM通常通过LPC(Low Pin Count)、I^2C(Inter IC)、SPI(Serial Peripheral Interface)、EPI(Extended Peripheral Interface)等总线与ICH(IO Controller Hub)或PCH(Platform Controller Hub)芯片相连接。兼容型可信PC的TPM与PC主CPU同时启动，CRTM就是Boot ROM(BIOS/EFI)的起始代码。

5.1.2 增强型可信PC的主板结构

中国国家标准GB/T 29827—2013《信息安全技术 可信计算规范 可信平台主板功能接口》提出了增强型可信PC的主板结构。增强型可信PC的主板由TPM和其他通用部件组成，以TPM可信根为核心部件实现完整性的度量、存储、报告机制，并实现平台可信

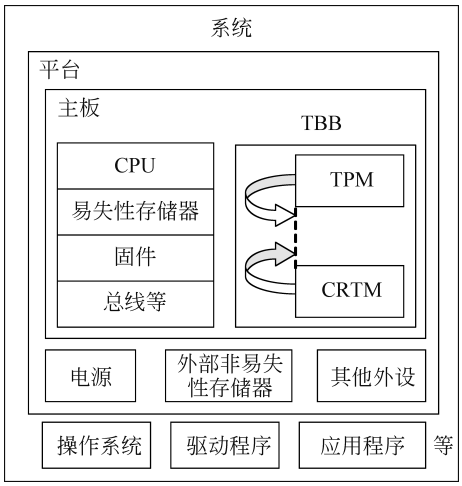

图 5-1　兼容型可信 PC 主板结构

引导功能，如图 5-2 所示。TPM 包括 TPM 硬件与固件系统，以及对外提供的驱动程序等实体。增强型可信 PC 主板是基于 TPM 模块的计算机主板，包括 CPU、动态存储器、显示控制器、总线、TPM 硬件设备、Boot ROM 固件层支撑模块及其设备驱动程序和 TPM 嵌入式系统等实体。计算机主板构建原则需确保 TPM 模块与主板的一一对应绑定关系。

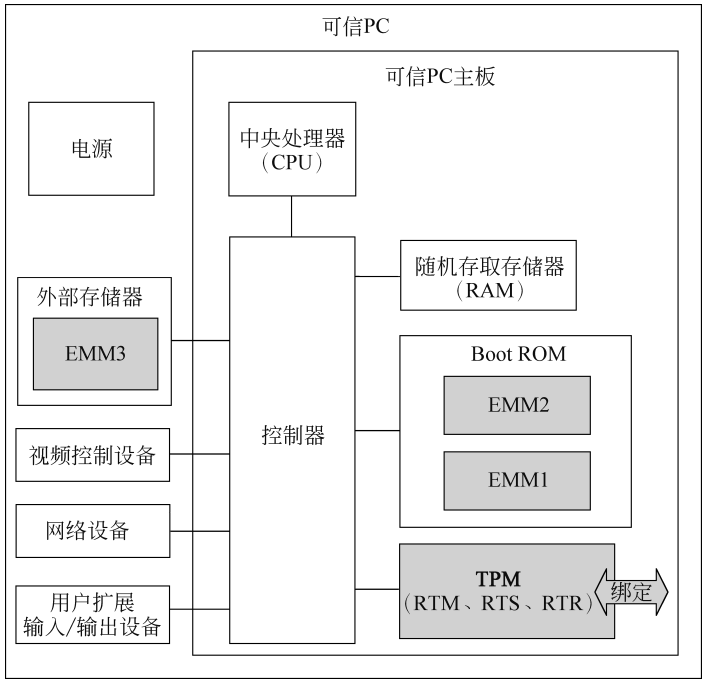

图 5-2　增强型可信 PC 的主板结构

增强型可信 PC 主板与兼容型可信 PC 主板的主要不同是，控制时序要确保计算机上电启动时 TPM 先执行最初的可信度量，然后主 CPU 再执行后续的度量。

增强型可信 PC TPM 与计算机主板其他部件的协作关系满足如下要求：

① TPM 先于计算机主板其他部件启动，包括主 CPU，为实现（Root of Trust Measure，RTM）度量 Boot ROM（BIOS/EFI）的初始引导代码 EMM1（扩展度量模块）构造必要条件。

② TPM 能够通过物理电路连接，可靠地读取主机 Boot ROM 的初始引导代码 EMM1，并对其实施完整性度量和 PCR 扩展存储操作。

增强型可信 PC 的 RTM 度量 EMM1 的完整性过程如下。

① TPM 可靠装载 RTM 程序代码。

② RTM 通过主板和 TPM 物理总线连接可靠地读取 Boot ROM 中指定地址区间的 EMM1 代码。

③ RTM 对 EMM1 执行哈希计算，并将结果扩展存储在 TPM 的 PCR 里。

④ TPM 确保能在 EMM2 请求下将 RTM 度量 EMM1 得到的结果发送给 EMM2 处理。

5.1.3 可信根

TPM 模块是 PC 的可信根（RT），包括可信度量根（RTM）、可信存储根（RTS）和可信报告根（RTR）。

① 可信度量根（RTM）：RTM 是对平台进行可信度量的基点。兼容性可信 PC 的 RTM 是 Boot ROM 的起始代码，用于对计算机系统资源进行最初的可信度量，它又被称为 CRTM。增强型可信 PC 的 RTM 是 TPM 嵌入式系统中用于度量 Boot ROM 程序起始代码的执行部件。

② 可信存储根（RTS）：RTS 是对可信度量值进行安全存储以及维护密钥树和保证数据安全的基点。它由 TPM 中的一组平台配置寄存器（Platform Configuration Register，PCR）和存储根密钥（Storage Root Key，SRK）或主存储密钥（Primary Storage Key，PSK）共同组成。

③ 可信报告根（RTR）：RTR 是平台向访问客体提供平台可信性状态报告的基点。它由 TPM 中的 PCR 和 EK 的派生密钥 AIK（Attestation Identity Key）或受限签名密钥（Restricted Sign Key，RSK）共同组成。

5.1.4 信任链和可信启动

无论是兼容型可信 PC 还是增强型可信 PC，在 RTM 完成对 EMM1 的可信度量后，均由可信扩展度量模块完成后续的信任链建立及可信启动过程。可信 PC 的 Boot ROM 主要有传统 BIOS 和扩展固件接口（Extended Firmware Interface，EFI）两种形式，下面分别进行描述。

1. BIOS 完整性度量

在使用 BIOS 的可信 PC 里，EMM1 是 BIOS 启动块，EMM2 是 BIOS Main Block，如图 5-3 所示。

PC 启动时，BIOS 启动块还不能使用 PC 的主内存，只能使用主 CPU 的寄存器。BIOS 启动块需要校验 BIOS Main Block 的完整性，若 BIOS 启动块直接用 PC 主 CPU 进行计算，由于不能使用 PC 主内存，效率会很低，校验 BIOS Main Block 的完整性会花费较长时间，整个可信 PC 的启动时间也会增加。为加快 BIOS 启动块对 BIOS Main Block 的完整性校验，BIOS 启动块将 BIOS Main Block 传送给 TPM（在 BIOS 中加入 TPM MA 驱动），由 TPM 完成哈希计算，如图 5-4 所示。

图 5-3　BIOS EMM1 EMM2 存储示意图

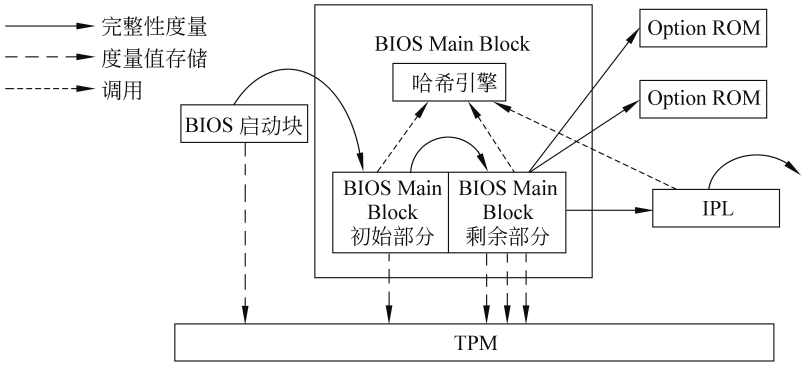

图 5-4　BIOS 完整性度量

与 PC 主 CPU 相比，TPM 的处理器的频率较低，不适宜进行大数据量的哈希运算，故 BIOS 启动块只将 BIOS Main Block 初始部分代码传送给 TPM 进行哈希计算（BIOS 中加入 TPM MP 驱动），该初始部分只包括 PC 主内存初始化模块、哈希引擎，以及调用哈希引擎对 BIOS Main Block 剩余部分进行完整性度量模块三部分。

BIOS 启动块利用 TPM 度量完 BIOS Main Block 初始部分的完整性后，便将平台主 CPU 的控制权转交给 BIOS Main Block 的初始部分。BIOS Main Block 初始部分就直接调用哈希引擎使用 PC 主 CPU 和主内存度量 BIOS Main Block 剩余部分的完整性，之后将 PC 主 CPU 的控制权转交给 BIOS Main Block 的剩余部分。BIOS Main Block 剩余部分再调用哈希引擎度量如下对象：PC 配置信息和配置事件、CPU 升级微码、Option ROM 代码、配置数据、初始程序装载器（Initial Program Loader，IPL）代码和配置数据等。在 PC 进行状态转换时，主 CPU 的控制权返回给 BIOS Main Block，完成对平台状态转换事件的度量。

BIOS 平台有关的度量对象以及该部分是否为必须度量见表 5-1。

BIOS Main Block 需要调用 TPM PCR 扩展等功能，需在 BIOS 中增加 MP（内存可用）驱动。BIOS 将 TPM 设备映射到一个基地址，负责与 TPM 进行通信。

MP 驱动的内存模式：MP 驱动的内存模型必须是保护模式。代码是浮动地址代码，不得对 EIP 的值做任何规定。

MP 驱动的基本函数如下。

表 5-1 BIOS 平台有关的度量对象以及该部分是否为必须度量

序号	度 量 对 象	必须/可选
0	BIOS Boot Block 版本标识符	必须度量
	POST BIOS 代码	必须度量
	SMM(系统管理模式)代码和建立系统管理模式的程序	必须度量
	ACPI Flash 数据	必须度量
	Embedded Option ROM：由主板厂商控制并维护的主板固件代码镜像	必须度量
1	CPU 升级微码	必须度量
	主机平台配置事件	必须度量
	ESCD、CMOS 和其他 NVRAM 数据	可选度量
	SMBIOS 结构	可选度量
2	被 BIOS 调用的 Option ROM Code	必须度量
	针对 BIOS 不可见的 Option ROM Code 部分	必须度量
	固化在主机平台的主板上，但由非设备制造商控制的 Option ROM Code	必须度量
4	Option ROM 的配置信息	必须度量
	Option ROM 的静态数据	必须度量
5	平台状态转换事件：系统从 S4(休眠)和 S5(关机)状态返回到 S0(全速运行)状态的状态转换事件	必须度量

① MPInitTPM，对 TPM 和驱动进行初始化。

② MPCloseTPM，关闭对 TPM 设备的连接。

③ MPGetTPMStatusInfo，从 TPM 设备读取当前错误和状态信息。

④ MPTPMTransmit，从输入缓冲向 TPM 传送数据，并向输出缓冲读取 TPM 的响应。

2. UEFI 完整性度量

在使用 EFI 的可信 PC 里，EMM1 是 Boot Core，为 EFI 执行的第一条指令到 TPM 驱动加载完毕之间的代码，通过插入在 EFI BIOS 中的 GUID 标识 EMM1 代码的结束位置；EMM2 是 EFI 主启动代码，如图 5-5 所示。

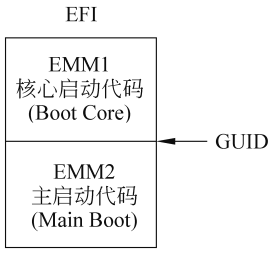

图 5-5 EFI EMM1 EMM2 存储示意图

EMM1 和 EMM2 之间包含 MAP1，MAP2，…，MAPi，EMM2 等多个度量代理点。EMM1 负责对后续加载的代码 MAP1 进行度量，然后将控制权传递给 MAP1，MAP1 再按照上述方法向后传递，信任链依此方式建立。MAP 完成对平台硬件配置信息、EFI BIOS 重要数据结构、核心代码等部件的完整性度量。各部分都度量完毕后，EMM2 度量 IPL(EMM3)，并传递控制权，如图 5-6 所示。

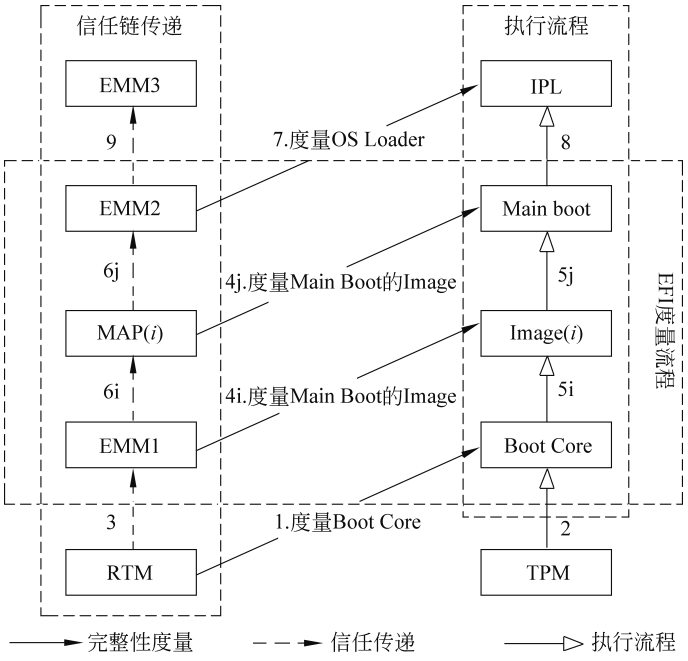

图 5-6　EFI 度量结构图

EFI 平台有关的度量对象以及该部分是否为必须度量见表 5-2。

表 5-2　EFI 平台有关的度量对象以及该部分是否为必须度量

序号	度 量 对 象	必须/可选
0	PC 厂商提供的 EFI 固件	必须度量
	嵌入在系统 ROM 中的 EFI 驱动程序	可选度量
	静态 ACPI 表	必须度量
1	其他数据表	可选度量
	影响系统配置的 EFI 变量	可选度量
2	适配器中 EFI 引导服务驱动程序	必须度量
	适配器或外部存储中 EFI 引导服务应用程序	必须度量
3	EFI 变量	可选度量
4	EFI 运行时应用程序(EFI OS 装载器)	必须度量
	EFI 运行时应用程序(HBA 系统配置工具 OS 装载器)	必须度量
	EFI 运行时应用程序(EfiChkDsk.efi 或 Diskpart.efi)	必须度量
5	EFI 引导变量 BootOrder	必须度量
	GPT 表	必须度量
	EFI 规范或私人定义的 EFI 静态变量	可选度量

EFI需调用TPM PCR扩展等功能,在UEFI里增加调用TPM的接口UEFI_TPM_PROTOCOL,该Protocol符合UEFI相关规范。

UEFI_TPM_PROTOCOL定义如下:

```
GUID #define UEFI_TPM_PROTOCOL_GUID
{0xf541796d, 0xa62e, 0x4954, 0xa7, 0x75, 0x95, 0x84, 0xf6, 0x1b, 0x9c, 0xdd}
typedef struct {
  UEFI_TPM_STATUS_CHECK            StatusCheck;
  UEFI_TPM_HASH_ALL                HashAll;
  UEFI_TPM_LOG_EVENT               LogEvent;
  UEFI_TPM_PASS_THROUGH_TO_TPM     PassThroughToTPM;
  UEFI_TPM_HASH_LOG_EXTEND_EVENT   HashLogExtendEvent;
} UEFI_TPM_PROTOCOL;
```

UEFI_TPM_PROTOCOL成员描述见表5-3。

表5-3 UEFI_TPM_PROTOCOL成员描述

数　据	描　述
StatusCheck	反映TPM当前状态的服务
HashAll	哈希计算服务
LogEvent	进行事件日志记录服务
PassThroughToTPM	访问TPM的命令通道服务
HashLogExtendEvent	服务:进行哈希计算、将结果扩展到TPM PCR中、记录事件日志

Event Log描述详细的完整性度量日志,在可信验证时需要使用Event Log。pre-OS阶段度量日志保存在ACPI表中。Event Log的一条记录的结构如下。

```
typedef struct
{
  TPM_PCRINDEX                     PCRIndex;
  TPM_EVENTTYPE                    EventType;
  TPM_DIGEST                       Digest;
  UINT32                           EventSize;
  UINT8                            Event[n];
} TPM_PCR_EVENT;
```

TPM_PCR_EVENT成员描述见表5-4。

表5-4 TPM_PCR_EVENT成员描述

数　据	描　述
PCRIndex	PCR序号
EventType	记录类型

续表

数　　据	描　　述
Digest	程序或数据哈希值
EventSize	记录内容的大小
Event[n]	所记录事件的内容

```
UEFI_TPM_STATUS_CHECK
typedef EFI_STATUS (EFIAPI * UEFI_TPM_STATUS_CHECK)
(
IN struct _UEFI_TPM_PROTOCOL            * This,
OUT TPCM_BOOT_SERVICE_CAPABILITY        * ProtocolCapability,
OUT UINT32                              * TPMFeatureFlags,
OUT UEFI_PHYSICAL_ADDRESS               * EventLogLocation
OUT UEFI_PHYSICAL_ADDRESS               * EventLogLastEntry
);
```

该服务用于获取 TPM 当前状况的信息以及 Event Log 相关内容。UEFI_TPM_STATUS_CHECK 函数参数描述见表 5-5。

表 5-5　UEFI_TPM_STATUS_CHECK 函数参数描述

数　　据	描　　述
This	指向 UEFI_TPM_PROTOCOL 的指针
ProtocolCapability	返回当前 TPM 的状态信息
TPMFeatureFlags	用于指示 Feature
EventLogLocation	指示 EventLog 位置的指针
EventLogLastEntry	指示内存中最后一个 Entry 的起始地址

```
UEFI_TPM_HASH_ALL
typedef EFI_STATUS (EFIAPI * UEFI_TPM_HASH_ALL)
(
IN struct _UEFI_TPM_PROTOCOL            * This,
IN UINT8                                * HashData,
IN UINT64                               HashDataLen,
IN TPM_ALGORITHM_ID                     AlgorithmId,
IN OUT UINT64                           * HashedDataLen,
IN OUT UINT8                            **HashedDataResult
);
```

该服务用于提供哈希操作。UEFI_TPM_HASH_ALL 函数参数描述见表 5-6。

表 5-6　UEFI_TPM_HASH_ALL 函数参数描述

数　　据	描　　述
This	指向 UEFI_TPM_PROTOCOL 的指针
HashData	指向将要被哈希的数据缓冲区的指针
HashDataLen	用于哈希计算的数据缓冲区的长度
AlgorithmId	哈希算法标识
HashedDataLen	哈希结果的长度
HashedDataResult	指向哈希结果的指针

```
UEFI_TPM_LOG_EVENT
typedef EFI_STATUS (EFIAPI * UEFI_TPM_LOG_EVENT)
(
IN struct _UEFI_TPM_PROTOCOL            * This,
IN TPM_PCR_EVENT                        * TPMLogData,
IN OUT UINT32                           * EventNumber,
IN UINT32                               Flags
);
```

该服务用于提供 Log Event 功能的操作，并将结果添加到 Event Log 的入口。UEFI_TPM_LOG_EVENT 函数参数描述见表 5-7。

表 5-7　UEFI_TPM_LOG_EVENT 函数参数描述

数　　据	描　　述
This	指向 UEFI_TPM_PROTOCOL 的指针
TPMLogData	指向内存中 TPM_PCR_EVENT 数据结构的首地址的指针
EventNumber	刚刚被 Log 的 Event 的数目
Flags	标志位

```
UEFI_TPM_PASS_THROUGH_TO_TPM
typedef EFI_STATUS (EFIAPI * UEFI_TPM_PASS_THROUGH_TO_TPM)
(
IN struct _UEFI_TPM_PROTOCOL            * This,
IN UINT32                               TpmInputParameterBlockSize,
IN UINT8                                * TpmInputParameterBlock,
IN UINT32                               TpmOutputParameterBlockSize,
IN UINT8                                * TpmOutputParameterBlock
);
```

该调用用于以字节流形式向 TPM 发送命令，以字节流形式返回 TPM 的结果。UEFI_TPM_PASS_THROUGH_TO_TPM 函数参数描述见表 5-8。

表 5-8　UEFI_TPM_PASS_THROUGH_TO_TPM 函数参数描述

数　　据	描　　述
This	指向 UEFI_TPM_PROTOCOL 的指针
TpmInputParameterBlockSize	TPM 输入参数块的大小
TpmInputParameterBlock	指向 TPM 输入参数块的指针
TpmOutputParameterBlockSize	TPM 输出参数块的大小
TpmOutputParameterBlock	指向 TPM 输出参数块的指针

```
UEFI_TPM_HASH_LOG_EXTEND_EVENT
typedef EFI_STATUS (EFIAPI * UEFI_TPM_HASH_LOG_EXTEND_EVENT)
(
IN struct _UEFI_TPM_PROTOCOL            * This,
IN UEFI_PHYSICAL_ADDRESS                HashData,
IN UINT64                               HashDataLen,
IN TPM_ALGORITHM_ID                     AlgorithmId,
IN OUT TPM_PCR_EVENT                    * TPMLogData,
IN OUT UINT32                           * EventNumber
OUT UEFI_PHYSICAL_ADDRESS               * EventLogLastEntry
);
```

该函数用于提供哈希以及 Extends Event 的功能。UEFI_TPM_HASH_LOG_EXTEND_EVENT 函数参数描述见表 5-9。

表 5-9　UEFI_TPM_HASH_LOG_EXTEND_EVENT 函数参数描述

数　　据	描　　述
This	指向 UEFI_TPM_PROTOCOL 的指针
HashData	指向将要被哈希的数据缓冲区的指针
HashDataLen	将要被哈希的数据缓冲区的长度
AlgorithmId	哈希算法标识
TPMLogData	指向物理地址中包含 TPM_PCR_EVENT 结构起始位置的指针
EventNumber	刚被 Log 的 Event 的数目
EventLogLastEntry	指向物理地址中刚刚放置的 Event Log 首字节的指针

TPM EFI 驱动，由 TPM 厂商提供给 EFI BIOS 厂商。

该驱动实现的 PROTOCOL 的 GUID 定义如下。

```
#define EFI_TPM_TDD_PROTOCOL_GUID \
{ \
0xe5d3c9db, 0x21b1, 0x4d0f, 0x93, 0xe6, 0xb3, 0x80, 0x68, 0xef, 0x5f, 0x1b \
}
```