Application Note

ZHCAEW7 – 12. 2024

C2000 安全特性在服务器电源上的应用

Shaoxing Ke

摘要

服务器电源是数据中心不可或缺的一部分，专门为服务器提供稳定可靠且安全的电力供应。随着 AI 数字化时代到来，云计算、大数据等技术迅速发展，人们需要应对日益复杂的网络安全, 信息安全, IEC62443 信息法规或 MCRPS 服务器电源标准加密需求等，从而云上服务器数据加密转变为本地终端设备数据存储, 身份验证且数据加密。为此, 本应用报告将详细介绍 C2000 器件上的安全特性来实现信息安全，并且通过安全启动 Secure Boot 在实际用户代码执行之前进行身份验证来确保芯片代码的安全, 此外还通过 JTAG Lock 功能来阻止芯片调试口的外部访问，防止代码的泄漏。

目录

图

1 引言

服务器电源产品要求应使用相应上位机命令 (即 0xxxh) 通过 PMBus 上的安全协议和数据模型 (M-CRPS 规范中 SPDM, Security Protocol and Data Model) 来支持安全认证, 其中包括了固件认证, 设备身份验证和代码保护, 详情可参考图 1, MCRPS 服务器电源 SPDM 信息安全需求。针对安全认证最好的实现方式是使用密钥并且进行握手应答。通常来说, 握手应答有两种方式：1). 基于**对称加密算法**的安全认证, 即加密和解密使用同一个密钥，常见算法如 SHA-256, CMAC-128 等，参考如图 2 所示；2). 基于**非对称加密算法**的安全认证, 即安全认证会依赖于私钥和公钥。只有被认证的设备才知道私钥, 而公钥可以公布给对设备进行安全认证的任何一方，常见算法如 ECDSA, RSA 和 SM-x 等，参考如图 3 所示。

12.9.2 Firmware Attestation

The BMC may attest the firmware of the power supply through the use of the SPDM GET_MEASUREMENTS request. The M-CRPS shall respond to the BMC with the measurements for each area requested. If M-CRPS is ready when the GET_MEASUREMENTS request is received, then it shall respond with the MEASUREMENTS response otherwise it shall respond with an ERROR response with the error ID set to "ResponseNotReady".

12.9.3 Device Authentication

To support device authentication the M-CRPS shall support the ECDSA Sign/Verify protocol using the NIST P-384 algorithm. The BMC may request to authorize the device through the use of the CHALLENGE request. The M-CRPS shall respond to the CHALLENGE request with an ERROR response with the error ID set to "ResponseNotReady". Once the challenge has been signed according to the SPDM spec the M-CRPS shall respond to the BMC's next RESPOND_WHEN_READY request with the CHALLENGE_AUTH response.

12.9.4 Code Protection

All microcontrollers within the M-CRPS shall enable code protections such that the FW on the device may not be read/written/erased or otherwise altered via any method other than authenticated signed images. This includes all debugging/programming pins.

Figure 1. MCRPS SPDM 信息安全需求

对称加密算法, 通过握手应答加密认证始终要求被认证对象本身持有密钥, 该密钥是主机与设备之间的共享密钥。这就要求加密芯片或者 MCU 器件本身拥有安全存储空间或者加密的 OTP 区域来存放该密钥。参考如图 2 所示。

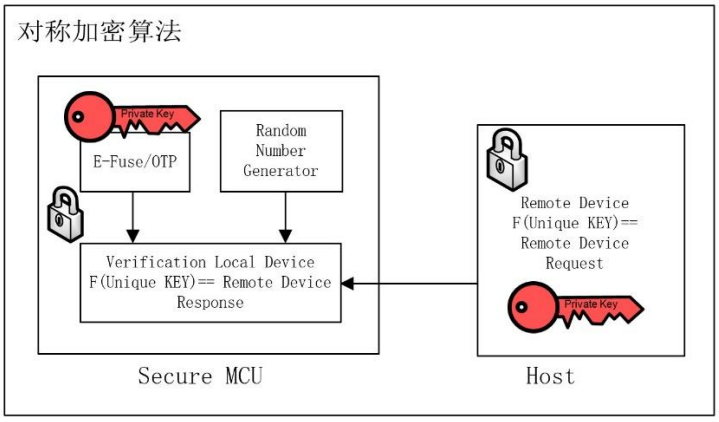


Figure 2. 对称密钥安全认证架构图

非对称加密算法，通过两个密钥即公钥和私钥来实现加密认证，其中**公钥是用来处理加密的消息**，只能由与其对应的私钥来恢复或解密；**私钥是用来处理或签名信息**，只能由与其对应的公钥来验证签名。私钥一般自己本地保存，公钥可以分享给任何设备的。通俗来讲，你需要和任何一个设备互发信息，首先利用该设备的公钥进行信息加密；其次，该设备方收到了加密信息并且利用自己本地的私钥进行解密，那么同时也实现签名的认证，最终实现固件的安全认证。参考如图 3 所示。

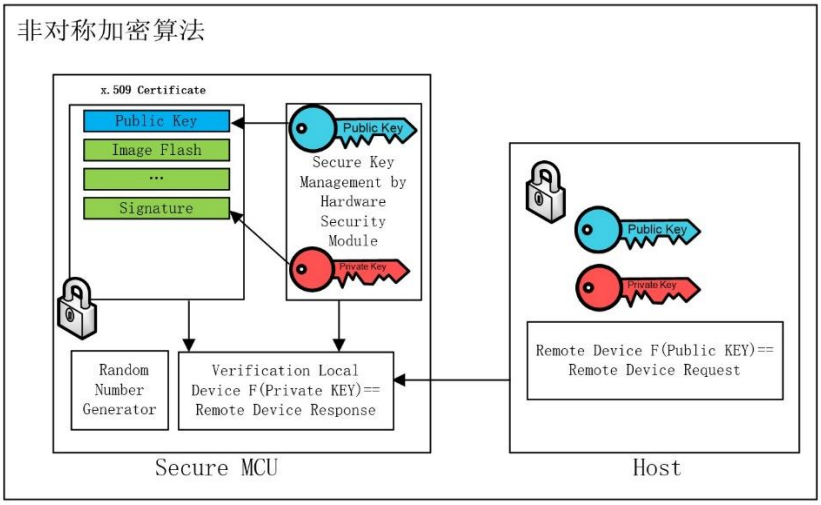


Figure 3.　非对称密钥安全认证架构图

针对固件认证，可使用 C2000 系列器件中的 Secure Boot 来进行固件验证；针对设备身份验证，可以使用 C2000 系列器件中的 Unique ID 来进行验证；针对代码保护可以使用 C2000 系列器件中的 DCSM 双代码加密模块来对芯片内存 RAM 和 Flash 进行读写保护，此外新一代的 C2000 器件如 F28P55x/F28003x/F28001x 系列器件等还可以通过 JTAG Lock 功能来阻止芯片调试口的外部访问，防止代码的泄漏，详情参考如图 4 所示。

Enabler	Feature	F2838x	F2837xD	F2837xS / F2807x	F28004x/ F28002x	F280013x/ F280015x	F2800 3x	F28P65 x	F28P55 x
Cryptographic Acceleration	AES : 128, 192, 256 engine	✓				✓	✓	✓	✓
Device Identity	UID programmed into OTP	✓	✓	✓	✓	✓	✓	✓	✓
Secure Boot	User programmed OTP setting	✓				✓	✓	✓	✓
	128b AES-CMAC based	✓				✓	✓	✓	✓
	Partial authenticated boot (16KB each core)	✓				✓	✓	✓	✓
	One time programmable Flash								✓
Secure Firmware Update	User programmed OTP setting	✓				✓	✓	✓	✓
	128b AES-CMAC based authentication	✓				✓	✓	✓	✓
Debug Security	Password controlled / permanent JTAG lock	✓				✓	✓	✓	✓
	Security-aware debugging	✓	✓	✓	✓	✓	✓	✓	✓
Trusted Execution Environment	Via multiple cores with IPC communication	✓	✓						
Secure Storage	Via multiple cores with IPC communication	✓	✓						
Software IP Protection	Secure memory zones	✓	✓	✓	✓	✓	✓	✓	✓
	Execute only memory	✓	✓	✓	✓	✓	✓	✓	✓

Figure 4.　C2000 不同系列器件加密特性图

此外, C2000 下一代的新型 C29x 内核 **F29H85x** 器件采用 64 位 CPU 架构, 同时支持多个并行流水线指令操作且允许对 CPU 执行的内存任务进行硬件级别安全隔离, 提供运行时安全性且不影响实时信号链运算性能, 最重要的是芯片内部集成硬件安全模块 HSM, 可实现最高等级 EVITA-Full 的信息安全。HSM 模块架构可参考如图 5 所示，F29H85x 安全特性可参考如图 6 所示。

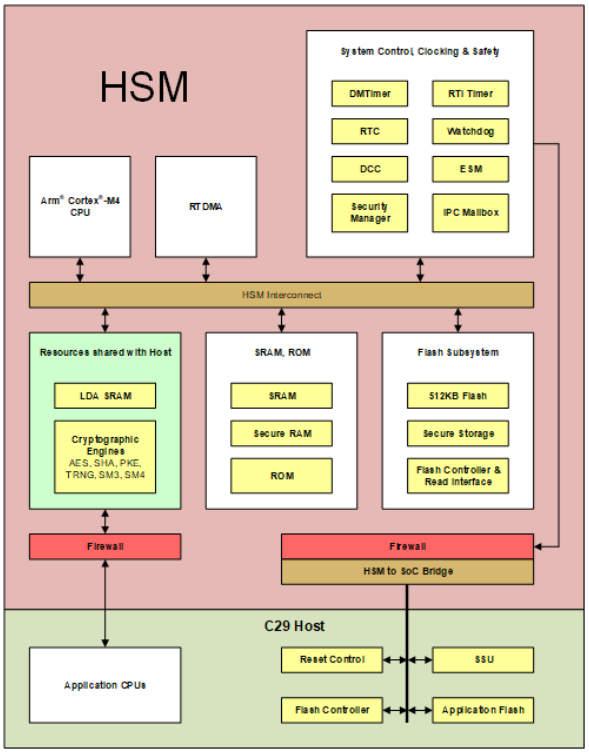
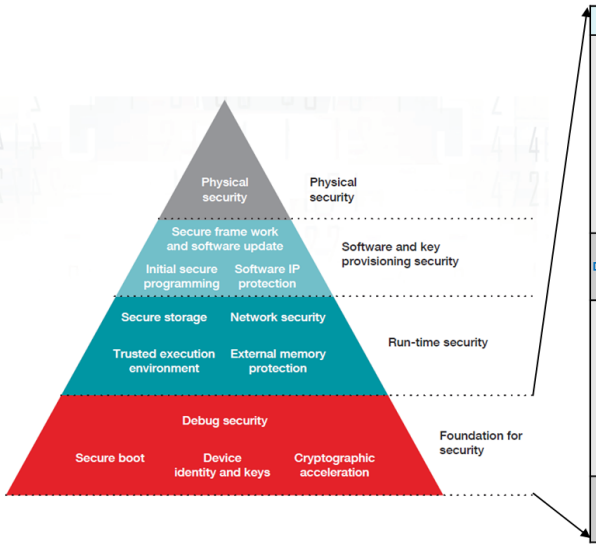


Figure 5.　F29H85x 系列器件硬件安全模块 HSM 示意图



Enabler	Feature	AM26x	F29x
Cryptography Acceleration	AES : 128,192, 256	✓	✓
	SHA2-224, 256	✓	✓
	SHA2: 384, 512	✓	✓
	True RNG	✓	✓
	DRBG	✓	✓
	PKA: RSA 2048 acceleration	✓	✓
	PKA: RSA 4096 acceleration	✓	✓
	PKA: ECC acceleration	✓	✓
	DMA support for cryptographic engines	✓	✓
Device ID and Keys	Device Public ID via ROM API	✓	✓
	Booting keys (MPK/SMPK, KEK, MEK/SMEK)	OTP	Secure Flash
	General OTP bits and other OTP (MSV, SWRV)	✓	✓
Secure Boot	Customer programs own keys	✓	✓
	RSA 2048-based		✓
	RSA 4096-based	✓	✓
	ECDSA (NIST Curve)		✓
	Authenticated boot	✓	✓
	Encrypted boot	✓	✓
	Model ID check	✓	✓
	Anti-roll back check	✓	✓
Debug Security	SW-controlled JTAG access	✓	✓
	Permanent disable JTAG via efuse	✓	✓
	Security-aware debugging	✓	✓

Figure 6.　F29H85x 系列器件安全特性列表

2 信息安全的常见几种方案

一般来讲, 有如下几种方案对 MCRPS 服务器电源产品中固件或设备进行安全验证：1). 通过外置安全加密芯片对主控 MCU 进行固件的安全验证, 但这会导致 BOM 增加, 成本也相应增加，即安全启动分立芯片方案; 2). 通过对主控 MCU 应用程序启动之前通过纯软件加密算法的安全验证, 这不仅有可能导致安全启动时间过长，而且该方案芯片硬件本身没有信任根, 有可能存在被攻击风险，即安全启动集成芯片纯软件方案 (例如纯软件方案 F2837xS/F2837xD Safe Copy Code and Safe CRC function);3). 通过芯片上硬件加密算法模块来缩短 CPU 软件计算时间, 并通过芯片安全存储空间 Secure OTP 区域来存放密钥防止密钥泄漏, 以及配合相应的软件加密算法来实现固件的验证即安全启动集成芯片部分硬件+部分软件方案; 4). 通过芯片上集成的 HSM 硬件安全模块(HSM, Hardware Security Module)对主控 MCU 的安全启动, 安全升级与身份签名进行认证, 且实现最高等级的安全认证即安全启动集成芯片纯硬件方案。

以上方案中, 方案 1 是安全启动分立芯片方案简单且架构灵活；方案 2 安全启动纯软件方案实现起来风险高且并不可靠；方案 3 安全启动集成芯片部分硬件+部分软件方案, 成本较低但软件实现复杂；方案 4 安全启动集成芯片纯硬件方案实现灵活且安全可靠。

本应用报告主要对方案 1 安全启动分立芯片方案 (Gen2 C2000 + Security Cortex-M0 MCU **AUTH1013**) 和方案 3 (Gen3 C2000 F28P55x/F28003x/F28001x)进行介绍, 实施与验证, 详情如下:

2.1 安全启动分立芯片方案介绍

通过外置安全加密芯片对主芯片 MCU 的安全启动, 安全升级与身份签名进行认证, 不仅可以使得原先软件架构改动较少, 而且实现方案简单, 参考如下图 7 所示。通过低成本的 M0 加密 MCU 芯片实现对固件的安全认证, 可以通过 I2C 外设接口和主机进行握手应答, 并且如果验证成功允许通过 UART 或 SPI 接口与主控 MCU 芯片进行信息交互, 否则验证失败则直接不允许主控 MCU 芯片正常启动。当然服务器基板 BMC 会将代码存放外部 Flash, 通过 Secure IC 若认证成功则允许数据进行交互, 进而主控 MCU 代码执行在外部的 Flash 芯片上,图 7 要求 C2000 MCU 本身 DCSM 对片内 Flash 进行加密处理。安全启动的分立芯片方案架构, 如图 7 所示。

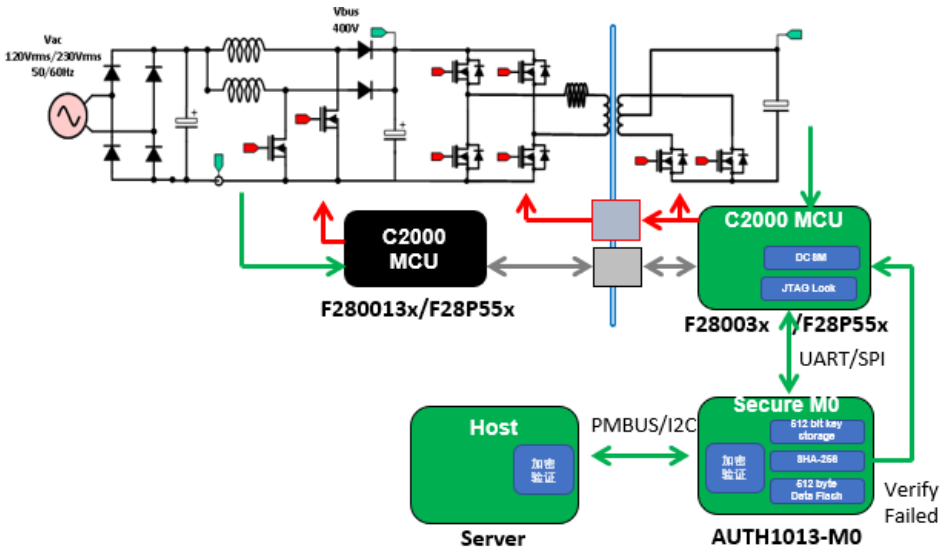


Figure 7.　安全认证的传统分立芯片方案架构图

AUTH1013 是低成本的 M0 安全认证加密芯片, 拥有**专用的 256bit 密钥安全存储空间**, 真实随机数发生器 TRNG, 超低功能运行模式睡眠模式下最低 5uA 消耗电流, 支持软件 SHA-256 库进行加密/解密等， 从而满足产品代码安全, 固件安全和身份验证的功能需求。

针对如 F2803x/F2802x 系列器件, 芯片拥有代码安全模块 CSM (Code security module, CSM)，以及仿真保护逻辑模块, 也可以支持上电默认从芯片加密区域启动即 Zero Boot Mode Selection Pin。其中代码安全模块 CSM, 允许程序和代码的操作权限级别, 参考如图 8 所示。此外, 如果芯片内部存储空间是处于加密状态的, JTAG 仿真器若尝试连接芯片则会触发芯片内部的仿真代码保护逻辑 ECSL, 从而**触发禁止仿真调试的功能会导致 JTAG 断开连接**。参考如图 9 所示。

由此, 针对老一代的 C2000 如 F2803x/F2802x 系列器件, 芯片必须要保证上电进入代码安全区域, 需要修改上电引导模式为 Flash Boot Mode 模式, 如此芯片才可以阻止外部 JTAG 仿真器的连接与访问, 避免代码的泄漏。ECSL 仿真保护逻辑流程图, 参考如图 10 所示。然而, 针对新一代的 C2000 器件如 F28003x/F28P55x 等, 可以直接利用 JTAG Lock 功能阻止仿真器的连接。

Table 1-9. Security Levels

PMF Executed With Correct Password?	Operating Mode	Program Fetch Location	Security Description
No	Secure	Outside secure memory	Only instruction fetches by the CPU are allowed to secure memory. In other words, code can still be executed, but not read
No	Secure	Inside secure memory	CPU has full access. JTAG port cannot read the secured memory contents.
Yes	Not Secure	Anywhere	Full access for CPU and JTAG port to secure memory

Figure 8.　代码安全模块 CSM 安全加密等级介绍图

5.2.2 Emulation Code Security Logic (ECSL)

In addition to the CSM, the emulation code security logic (ECSL) has been implemented using a 64-bit password (part of existing CSM password) for each zone to prevent unauthorized users from stepping through secure code. A halt in secure code while the emulator is connected will trip the ECSL and break the emulation connection. To allow emulation of secure code, while maintaining the CSM protection against secure memory reads, you must write the correct 64-bit password into the CSMKEY (0/1) registers, which matches the password value stored in the USER OTP of that zone. This will disable the ECSL for the specific zone.

When initially debugging a device with the password locations in OTP programmed (secured), the emulator takes some time to take control of the CPU. During this time, the CPU will start running and may execute an instruction that performs an access to a protected ECSL area and if the CPU is halted when the program counter (PC) is pointing to a secure location, the ECSL will trip and cause the emulator connection to be broken.

The solution to this problem is to use the Wait Boot Mode boot option. In this mode, the CPU will be in a loop and will not jump to the user application code. Using this BOOTMODE, you can connect to CCS and debug the code.

Figure 9.　ECSL 仿真保护逻辑说明

然而, 针对固件的认证, 可以通过低成本的安全认证加密芯片 AUTH1013 来实现, 可以通过对称加密算法将密钥存储于专用的 256-bit 安全存储空间, 通过软件上调用 SHA-256 软件库来对代码进行加密/解密, 从而将受信任的信息存入 512 Byte 数据 Flash 空间。如果来自上位机的“挑战-质询”模式是验证成功的, 则将上位机的命令信息传递给主控 MCU, 进而去执行相应的功能操作或者查询信息。针对身份验证, 可以通过 TRNG 真实随机数发生器来自动生成特有的 ID 号, 进而将该信息包含在验证信息的头文件信息中, 从而来区分不同服务器电源产品的特有身份证。

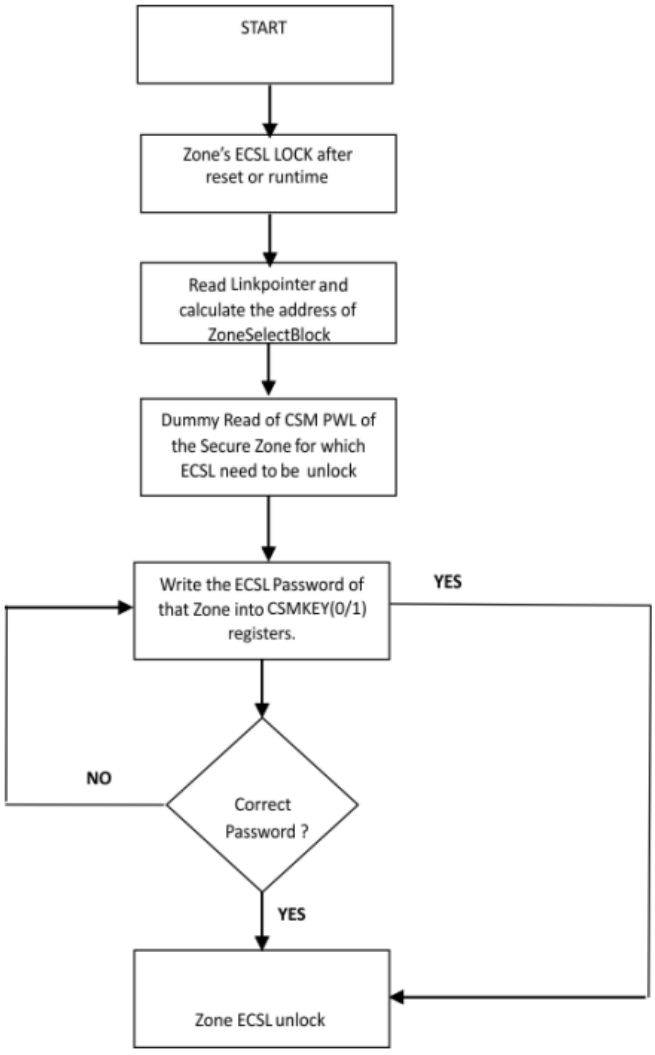


Figure 5-4. ECSL Password Match Flow (PMF)

Figure 10. ECSL 仿真保护逻辑流程图(所有 C2000 器件都有该仿真保护逻辑)

2.2 安全启动集成芯片方案介绍

新一代 C2000 如 F28P55x/F28003x 器件本身支持 Secure Boot 安全启动, 如参考图 4 所示。由此, 可以直接依赖主控 MCU 器件的安全加密特性, 来满足服务器电源产品的固件认证, 身份认证和代码保护的需求。安全启动的集成芯片方案架构, 如图 11 所示。备注,图 11 并不指代具体的前级 AC-DC 整流拓扑和后级 DC-DC 降压拓扑。（如常见 AC-DC 有无桥图腾柱 PFC, 无桥 Boost PFC 和三相维也纳 PFC 等, 常见 DC-DC 有全桥 LLC，DAB 双有源桥和移相全桥 PSFB 等）本应用报告中并不指定具体的拓扑, 一般来讲安全加密认证一般会是在后级 DC-DC 侧, 这将共用后级主控 MCU 与 Server 服务器的 PMBus 通讯协议。除此双 MCU 架构, 当然也会拥有单芯片 MCU 架构, 但一般由于隔离成本和控制复杂度应用较少。下一章节会详细讲解安全启动集成芯片方案的具体实施。

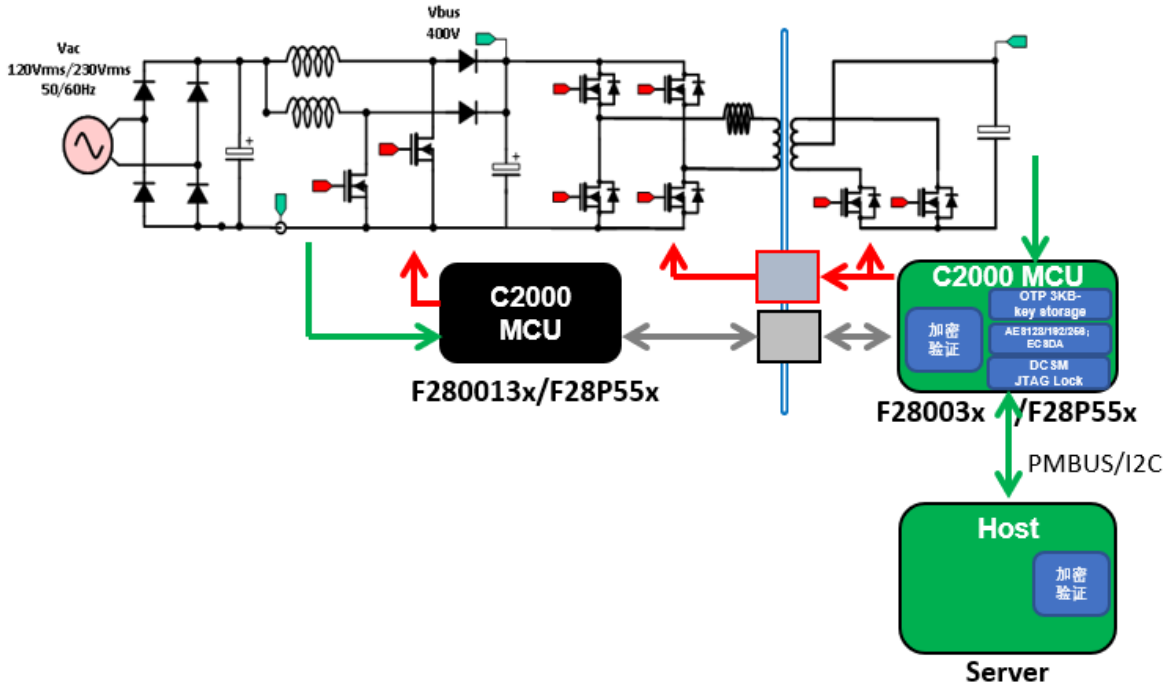


Figure 11.　安全认证的集成芯片方案架构图

2.3　安全启动集成芯片方案实施

如图 11 所示，固件安全验证是在应用程序执行之前进行的身份验证。其中，安全闪存启动 Secure Boot 是使用 128 位 AES-CMAC 认证算法实现的，该算法在应用程序代码内容上运行，返回通过/失败状态，并且仅在认证成功时继续执行应用程序代码。其主要功能是在执行之前对闪存中的用户应用程序代码进行认证的能力。这通过确保应用程序代码在编程到闪存中后未被篡改来确定其完整性。基于密码的消息认证码 (CMAC) 是一种基于 AES 的认证算法，它从输入数据块构建认证标签输入数据块一次以 128 位的形式输入到加密引擎/软件（基于 CPU 子系统）中，以及 128 位 CMAC 密钥。密钥位于 CPU1 USER OTP 中，并由设备所有内核上的 CMAC 认证算法使用。加密引擎/软件对整个输入数据块运行 CMAC 算法并生成最终的 128 位认证标签，并存储于 Flash 头 128bit 位置。如参考图 12 安全内存启动 Boot 流程所示。

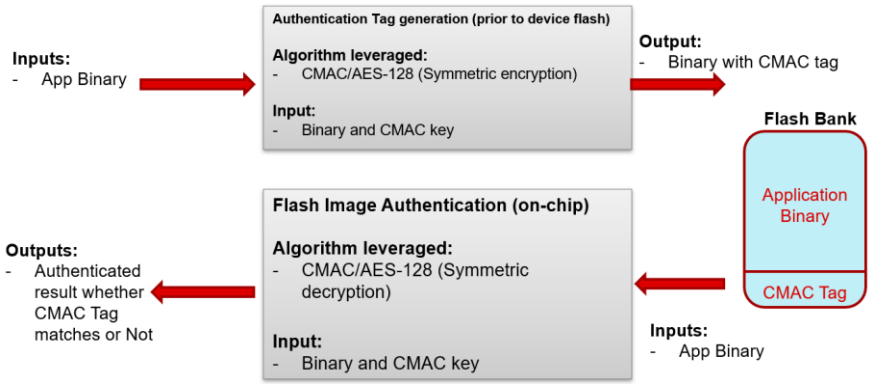


Figure 12.　安全内存启动 Boot 流程图

如参考图 12 包括两个步骤：

1) 创建并生成身份验证标签方法即密钥：其中可以通过 On-Chip flash tool or UniFlash Tools 编程或者 SysConfig DCSM Tool, 本应用报告如下章节会进行详细介绍(本文密钥示例 **0x0011 2233 4455 6677 8899 AABB CCDD EEFF**)；

2) 对 Flash 闪存中的应用程序执行前进行身份验证, 确保调用 Secure ROM 安全内存中的 **CPU1BROM_calculateCMAC** 来进行身份验证，本应用报告第 3 章节进行详细介绍, 如图 13 ROM Secure Boot 流程图。

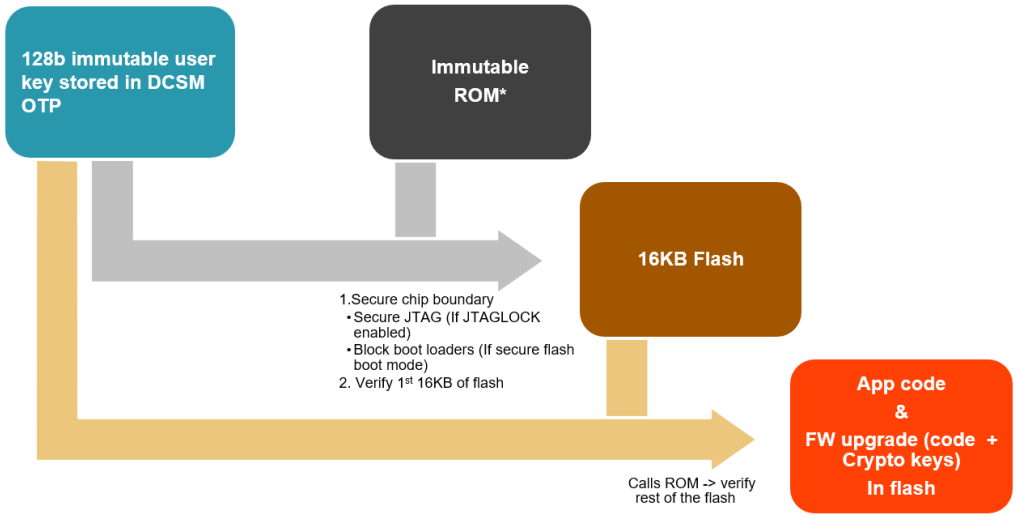


Figure 13.　安全认证的传统分立芯片方案架构图

创建并生成身份验证标签方法。可以通过 On-Chip flash tool or SysConfig DCSM Tool 对 128 bit CMAC Key 进行 OTP 区域的编程, 参考如图 14 所示。CMAC Key 是属于对称密钥, 需要器件本身 OTP 存放该密钥, 同时代码本身 HEX Binary 包含该密钥。除此之外, C2000 器件提供 ECSDA 签名库对签名进行验证, 提供 AES 硬件加速器对代码进行加密, 从而也可以实现非对称加密的算法。

use the hex conversion utility as follows to apply the CMAC algorithm to regions in allocated memory:

- Use the --cmac=file option. The file should contain a 128-bit hex CMAC key. The CMAC key in the file specified by the --cmac command-line option must use the format 0xkey0key1key2key3 in order to access the device registers for CMACKEY0-3. For example, the following file contents represent CMACKEY registers containing key0= `0x7c0b7db9`, key1= `0x811f10d0`, key2= `0x0e476c7a`, and key3= `0x0d92f6e0`.

```
0x7c0b7db9811f10d00e476c7a0d92f6e0
```

- Use either the --image option or the --load_image option when using the --cmac option. If you use the --image option, set both --memwidth and --romwidth to 16.
- If you use the --boot option (and other boot table options described in Section 12.11.3) with the --cmac option, the CMAC algorithm assumes that a fill value of 1 is used for gaps between boot table regions. Because of this assumption, you should also set --fill=0xFFFF when using the --boot and --cmac options together.
- Specify a HEX directive with one entry that represents all the allocated flash memory. Use a 128-bit aligned length and specify the optional fill value. (The default fill is set to 0's.)
- Define the global CMAC tags in C code.

Figure 14.　CMAC 128-bit Key 密钥演示图

Flash 闪存中的应用程序执行前进行身份验证。首先，我们通过存储 OTP 区域的 128 bit CMAC Key 对芯片 Secure ROM 启动进行身份验证，一旦验证成功，则允许我们调用 Secure ROM 中的 CPU1BROM_calculateCMAC 对芯片片上 16KB Flash 内存之外的进行身份验证。详细的验证流程参考如图 16 所示。

The CMAC calculate and compare function (Table 4-53) allows calculation of the CMAC signature of a Flash memory block and compare against a golden signature. This is used in the secure boot mode to authenticate the boot image.

Table 4-53. CMAC Calculation Function

CPU	Function Prototype	Function Parameters	Function Return Value
CPU (C28x)	uint32_t **CPU1BROM_calculateCMAC**(uint32_t startAddress, uint32_t endAddress, uint32_t signatureAddress)	**startAddress**: Starting address of memory for which CMAC has to be calculated **endAddress**: Ending address of memory for which CMAC has to be calculated **signatureAddress**: Address of location where golden CMAC signature is stored	**0xFFFFFFFFU**: Calculated CMAC signature did not match golden signature (fail) **0xA5A5A5A5U**: Memory range provided is not aligned to 128-bit boundary or length is zero **0xE1E1E1E1U**: AES Engine timed out **0x00000000U**: No Error

Figure 15. CPU1BROM_calculateCMAC API 函数说明

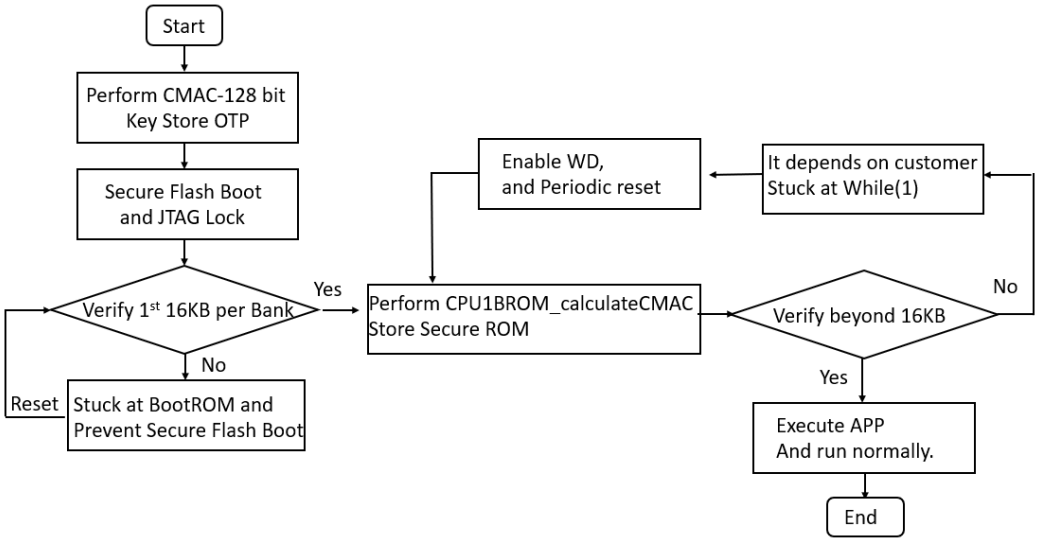


Figure 16. 安全启动固件验证的流程图-对称加密算法 CMAC

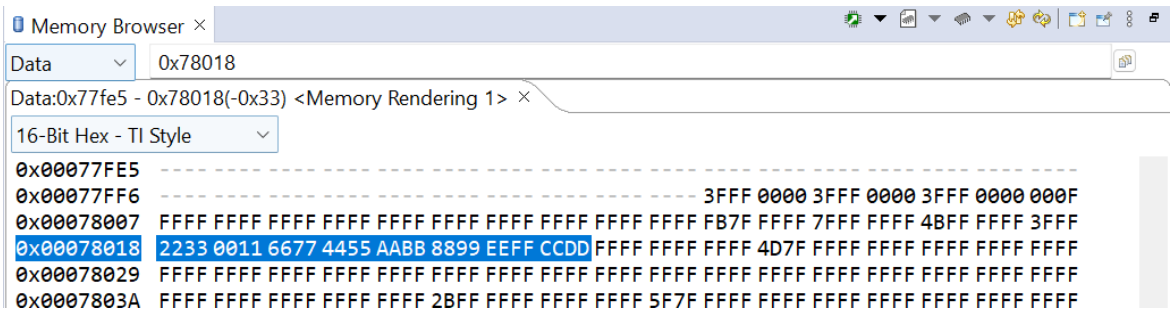


Figure 17. 存储 OTP 区域的对称加密密钥, 地址(0x78018 ~0x7801E)

参考如下：如果验证失败, 则芯片无法正常跳转到 **Secure Flash Entry Point Address (**如图 20 所示),芯片代码卡顿在芯片 Boot ROM, 此时 cpu1_SuccessfullyBooted 返回值为默认 0; 如果验证成功, 则芯片正常跳转 Secure Flash Entry Point Address, 且紧接着通过调用 Secure ROM 中的 CPU1BROM_calculateCMAC 对芯片 16KB Flash 内存之外的进行身份验证，验证成功 applicationCMACStatus 状态返回 1, 如图 18 和图 19 所示。

```
181     //
182     // Call CMAC Authentication API to verify contents of all of
183     // device flash.
184     //
185     // Upon failure, enter infinite loop.
186     //
187     applicationCMACStatus = CPU1BROM_calculateCMAC(CMAC_AUTH_START_ADDRESS,
188                                                    CMAC_AUTH_END_ADDRESS,
189                                                    CMAC_AUTH_TAG_ADDRESS);
190
191     if(CMAC_AUTH_PASS != applicationCMACStatus)
192     {
193         //
194         // Application will get stuck here upon failure
195         //
196         while(1);
197     }
198     else
199     {
200         cpu1_SuccessfullyBooted = true;
201     }
202
```

Figure 18. CPU1BROM_calculateCMAC 测试代码

Variables | Expressions | Registers

Expression	Type	Value
DcsmZ1Regs	Register Group	
CpuSysRegs.RESC	Register	0xC0000004
DcsmZ1Regs	Register Group	
cmac_all	struct CMAC_TAG	{tag=[0xDFCE '\xdfce',0xC17A 'z',0x05CD '\x5cd',...
cmac_sb_1	char[8]	[0xAC75 'u',0x20BB '\x20bb',0x450F '\x450f',0xB...
cpu1_SuccessfullyBooted	unsigned char	0 '\x00'
GpioDataRegs.GPADAT.GPIO12	pointer : 1	0x0

Figure 19. CPU1BROM_calculateCMAC 验证失败结果

Table 4-20. Secure Flash Boot Details

Details	Location Address
CMAC Signature Address	Flash Entry Point Address + 0x2
CMAC Key Address (128-bit key)	DCSM Z1 OTP CMACKEY0/1/2/3
Flash Entry Point (Bank 0, Sector 0)	0x0008 0000
Flash Entry Point (Bank 0, Sector 8)	0x0008 8000
Flash Entry Point (Bank 0, End of Sector 15)	0x0008 FFF0
Flash Entry Point (Bank 1, Sector 0)	0x0009 0000
Flash Entry Point (Bank 1, End of Sector 7)	0x0009 7FF0
Flash Entry Point (Bank 1, End of Sector 15)	0x0009 FFF0
Flash Entry Point (Bank 2, Sector 0)	0x000A 0000
Address Range for CMAC Calculation	Start: Flash Entry Point Address End: Flash Entry Point Address + 16 KB

Figure 20. Secure Flash Boot 信息列表

身份安全验证是通过存储于 OTP 区域的 256bit 设备唯一标识号 Unique ID, 可以通过 UID_REGS 寄存器来获取身份标识码。参考如图 21 和图 22 所示。

Table 4-1. Unique ID Locations on C2000 Devices

Device Family	MSW Location	LSW Location
F24x and F240x	N/A	N/A
F280x	0x000809	0x000808
F2802x	0x000901	0x000900
F2803x	0x000901	0x000900
F2805x	0x3D7FDB	0x3D7FDA
F2806x	0x000901	0x000900
F2823x and F2833x	0x000901	0x000900
F2807x(1)	0x0703CD	0x0703CC
F2837x(1)	0x0703CD	0x0703CC
F28004x(1)	0x0703CD	0x0703CC
F2838x(1)	0x07020D	0x07020C
F28002x(1)	0x0701F5	0x0701F4

(1) See the device data sheet and technical reference manual for more information regarding this UID_UNIQUE register.

Figure 21. Unique ID 设备唯一标识号列表

3.15.16.7 UID_UNIQUE Register (Offset = Ch) [Reset = X]

UID_UNIQUE is shown in Figure 3-281 and described in Table 3-308.

Return to the Summary Table.

UID Unique 32 bit number

Figure 3-281. UID_UNIQUE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UniqueID																															
R-X																															

Table 3-308. UID_UNIQUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	UniqueID	R	X	Unique portion of the UID. This identifier will be unique across all devices with the same PARTIDH. Reset type: N/A

Figure 22. Unique ID 寄存器示意图

代码安全是通过芯片 DCSM 双代码加密模块对片上存储空间 Flash 和 RAM 进行读写的保护, 以及访问权限的设置, 参考如图 23 所示。

针对 DCSM 代码加密/解密, 可以通过在工程代码中写入对应的密码或者 C2000 SysConfig GUI 界面中写入对应的密码; 可以通过 CCS On chip Flash 或者 UniFlash 烧录工具写入加密/解密密码; 还有可以直接代码中调用或者目标配置启动文件 GEL 中写入对应的加密/解密密码, 参考如图 24 所示。当然, 若解密之后为保证程序执行前是处于加密状态, 可以通过 Z1_CR.FORCESEC 软件强制使能 DCSM 进行如图 23 所示的流程, 此时若 CPU 读取 CSMKEY 寄存器状态则让芯片处于再次上锁的状态。

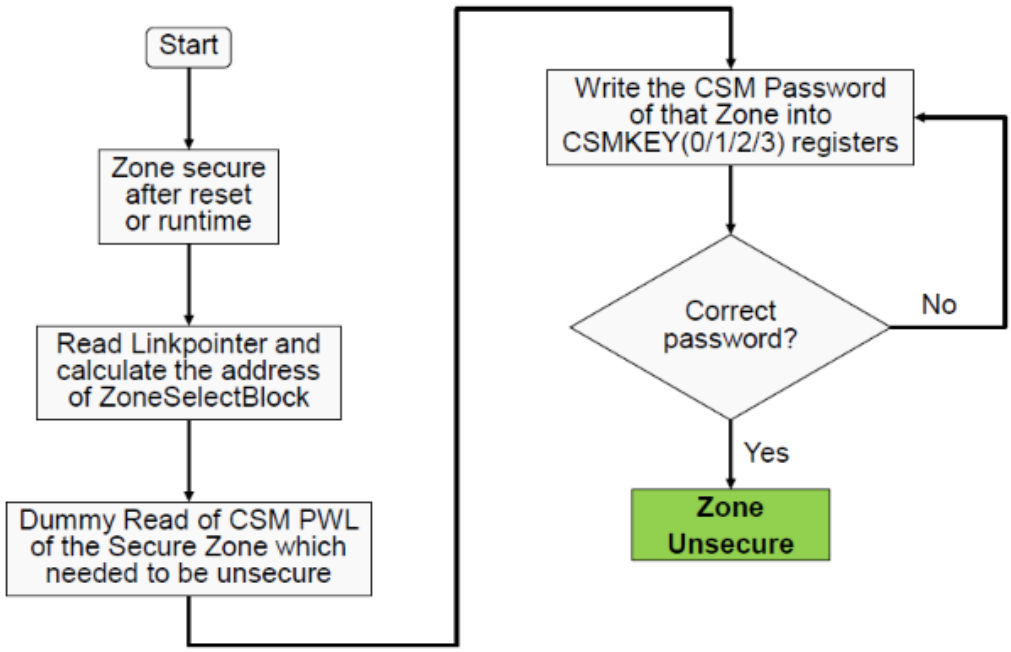


Figure 23. DCSM 代码安全流程示意图

```
if ((*(unsigned long *)(DCSM_Z1_BASE + DCSM_O_Z1_CR) & DCSM_Z1_CR_UNSECURE) == 0U)
{
    //LOCKED State --> ARMED State
    *(unsigned long *)(DCSM_Z1_BASE + DCSM_O_Z1_CSMKEY0) = 0x55555555; //Zone 1 CSMKEY Loads
    *(unsigned long *)(DCSM_Z1_BASE + DCSM_O_Z1_CSMKEY1) = 0x4d755555;
    *(unsigned long *)(DCSM_Z1_BASE + DCSM_O_Z1_CSMKEY2) = 0x55555555;
    *(unsigned long *)(DCSM_Z1_BASE + DCSM_O_Z1_CSMKEY3) = 0x55555555;
}
```

Figure 24. DCSM 代码加密/解密示意图

2.4 安全启动集成芯片方案验证

本应用报告只是通过应用 F28P55x 芯片上 Secure Boot 安全启动特性对产品进行对称加密的验证。其中安全启动的执行时间也是大多数客户关注的其中一个参数, 数据中心无法容忍长时间的启机时间, 因此基于 F28P55x 开发板进行实测有如下安全启动的时间, 参考如下表 1 所示。

Table. 1 F28P55x 安全启动时间列表

内存大小 /Flash	正常启动	安全启动对称加密(AES – CMAC)
Entire Flash (256KB)	6.54 ms	9.58 ms (1st 16KB of Flash) 55.18 ms – 70 ms (verify beyond 16KB Flash)

3 总结

无论是分立芯片的安全验证方案, 或是单芯片集成的安全验证方案都是可以满足服务器电源 MRCPS 加密的固件认证, 身份认证和代码安全的需求。可以通过芯片本身的 Secure Boot 安全启动对固件运行前进行验证, 避免执行错误或者恶意攻击的伪代码; 可以通过芯片的 256 bit Unique ID 来标识唯一身份; 可以通过芯片的 DCSM 双代码加密模块来对芯片的内存存储空间进行读写保护, 避免代码泄漏或者篡改; 最后, 通过锁定一种 Flash Boot Mode 保证芯片上电的安全执行, 和利用 JTAG Lock 功能来阻止外部调试口的访问。总之, C2000 器件的安全特性是可以满足 MCRPS 服务器加密的需求。

4 参考文献

1. TMS320C28x Assembly Language Tools v20.2.0.LTS User's Guide
2. Secure BOOT on C2000 Device
3. Enhancing Device Security by Using JTAGLOCK Feature
4. TMS320F28003x Real-Time Microcontrollers Technical Reference Manual
5. TMS320F280013x Real-Time Microcontrollers Technical Reference Manual
6. TMS320F2855x Real-Time Microcontrollers Technical Reference Manual
7. C2000™ Unique Device Number (Rev. B)

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。 严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265