*Technical White Paper*

# *实现高性能实时控制系统的网络安全*

*Ibukun Olumuyiwa and David Foley*

**摘要**

随着现代汽车和工业产品的复杂性、性能和连接性不断增加，对强大的嵌入式网络安全解决方案的需求也越来越大。面对不断增长的威胁因素以及全球不断变化的法规要求，芯片制造商和 OEM 都必须在不影响性能的情况下调整和实现更强大的产品安全性。为了有效抵御嵌入式硬件和软件上日益复杂的攻击，需要采用多层方法。安全信任根、安全存储、加密加速、可信执行环境、安全密钥和代码配置以及运行时上下文隔离等要素是现代高性能实时微控制器中网络安全的重要组成部分。德州仪器 (TI) 的 AM26x 和 F29x 微控制器系列经过全新设计，可实现这些安全目标，同时在不影响性能的前提下为实时应用提供卓越性能。

## 内容



## 商标

# 1 引言

现代汽车和工业产品（从汽车和火车到伺服驱动器和服务器电源单元）的复杂性不断增加，需要具有更高性能的实时控制解决方案。与此同时，这些产品已高度互联，因此需要强大的网络安全解决方案来保持硬件和软件资产以及与其相关的系统的机密性、完整性、真实性和可用性。此外，应用程序在运行时的安全性受到了更大的关注，更大且更复杂的软件栈导致潜在威胁行为者的攻击面增加。

为了有效抵御嵌入式硬件和软件上日益复杂的新型攻击，需要采用全面的多层方法，有效地建立信任根、为加密密钥等关键资产提供安全存储、创建可执行安全敏感操作、安全密钥和代码配置以及运行时上下文隔离和内存保护的可信执行环境，以减轻恶意软件在系统中的潜在影响。本白皮书探讨了这些主题，以及安全微控制器架构如何在不影响性能的情况下尽可能实现这些网络安全目标。

# 2 需要全面的安全方法

嵌入式微控制器中的网络安全与所部署系统的安全性和功能直接相关。这些系统通常用于关键应用，如汽车系统、医疗设备和工业控制系统。这些设备容易受到各种类型的攻击，例如数据和知识产权盗窃、拒绝服务、恶意软件注入、远程控制和物理篡改。即使未直接连接到互联网的器件也可被这些类型的攻击利用，如 Stuxnet 和 Rowhammer 等高调攻击就证明了这一点。在汽车市场中，利用中间人攻击和 CAN 注入攻击实现的车辆盗窃在重要性和频率上都有所增加，凸显了对更强大运行时安全保护的需求。

这些系统中嵌入式软件复杂性的不断增加，也为攻击者提供了更多可乘之机。例如，虽然安全调试（“JTAG 锁定”）可以为芯片边界提供合理的强大保护，但 CAN、SPI 和 I2C 等通信接口可以为恶意软件或恶意输入提供潜在入口点。此外，旁路分析和故障注入攻击可用于绕过现有的安全保护，甚至提取机密和加密密钥。为了防御可能通过未经授权的固件更新引入系统的恶意软件，必须使用加密可靠的信任根（称为安全启动）。如今，各种嵌入式器件都具有某种形式的安全启动，无论代码存储在外部还是内部闪存存储器中，都能确保执行前的固件完整性。

但是，即使在嵌入式闪存微控制器中，仅靠安全启动是不够的。有效的上下文隔离和存储器保护有助于保护关键系统代码免受通过易受攻击的外部接口（如通信端口）引入的恶意软件的影响。

此外，加密算法的软件实现一直是许多侧通道攻击（如计时攻击）的主题，导致机密公开。软件加密算法由于计算复杂性也可能无法满足系统性能要求。硬件加密加速器可显著提升这些算法的性能，并提供内置保护功能以抵御常见攻击。

# 3 加密功能

加密是嵌入式安全的核心，也是保护机密性、完整性和真实性等网络安全属性的基本方法。机密性要求使用无法被未经授权的各方破解或破坏的强大算法进行加密。发送方使用加密算法和加密密钥将要传输的数据转换为不可解密的密文，接收方随后必须使用相应的解密算法和解密密钥将密文转换回原始数据。完整性和真实性是专注于建立信任的相关网络安全属性。完整性可确保数据在传输或存储过程中未被更改或损坏，而真实性可确保数据来自合法可信的来源。需要哈希函数和数字签名来确定代码或数据的完整性和真实性。为了实现网络安全目标并有效保护受保护资产的机密性、完整性和可用性，可以使用其中一种或多种加密功能来构建嵌入式网络安全解决方案。

## 3.1 加密和解密

有两种类型的加密密码：对称和非对称。对称加密依赖于单个共享密钥进行加密和解密，并且只要共享密钥保持机密且未被更改，就会保留机密性。但是，如果共享密钥暴露给第三方，受保护的信息可能会被解密为纯文本，并且保密会丢失。因此，保持对称加密中使用的共享密钥的机密性对于保持私人信息的机密性至关重要。高级加密标准 (AES) 是世界上使用最广泛的对称算法之一，密钥长度范围为 128 至 256 位。AES 已被美国国家标准与技术研究院 (NIST) 采用。

另一方面，非对称加密使用互补的密钥对，即一个公钥和一个私钥。由公钥加密的数据只能通过其关联的私钥来解密。非对称加密比对称加密慢得多，因此这些功能通常仅限于加密对称密钥。一旦由接收器解密，随后可使用对称密钥对消息的其余部分进行解密。Rivest-Shamir-Adleman 算法套件 (RSA) 和椭圆曲线加密 (ECC) 是两种最常见的非对称密码系列。

## 3.2 哈希、数字签名和身份验证

除了加密和解密外，安全系统还必须能够确认器件上存储的代码和数据资产的完整性。哈希算法支持这一目标，将任意长度的代码或数据块简化为独特的固定长度摘要。如果不使用密钥，加密哈希函数始终会为同一输入生成相同的输出摘要，并具有几个可确保其安全的重要属性：

1. 推导出用于计算给定哈希值的原始输入字符串是非常不可行的。
2. 对于原始报文，修改该报文的方式与原始报文产生相同的哈希值是非常不可行的。
3. 加密哈希函数不受冲突影响，这意味着两个不同的输入很难生成相同的输出。
4. 此外，输入的任何更改 ( 即使是 single-bit ) 都会导致输出发生大幅变化。这一过程称为雪崩效应。

通过计算要引导的身份验证证书或存储代码上的哈希摘要，并将其与已知参考进行比较，系统可以确认代码或数据自创建以来未被修改。安全哈希算法 (SHA) 和消息摘要 5 (MD5) 是常用哈希函数的示例。不建议使用 SHA-1 和 MD5 算法，因为已针对这些函数证明了成功的冲突攻击。可以改用 SHA-2 和 SHA-3 算法来提供强大的哈希函数。哈希摘要的长度与函数的加密强度有关。所有内容都相等，较长的摘要更安全，但代价是计算时间增加。

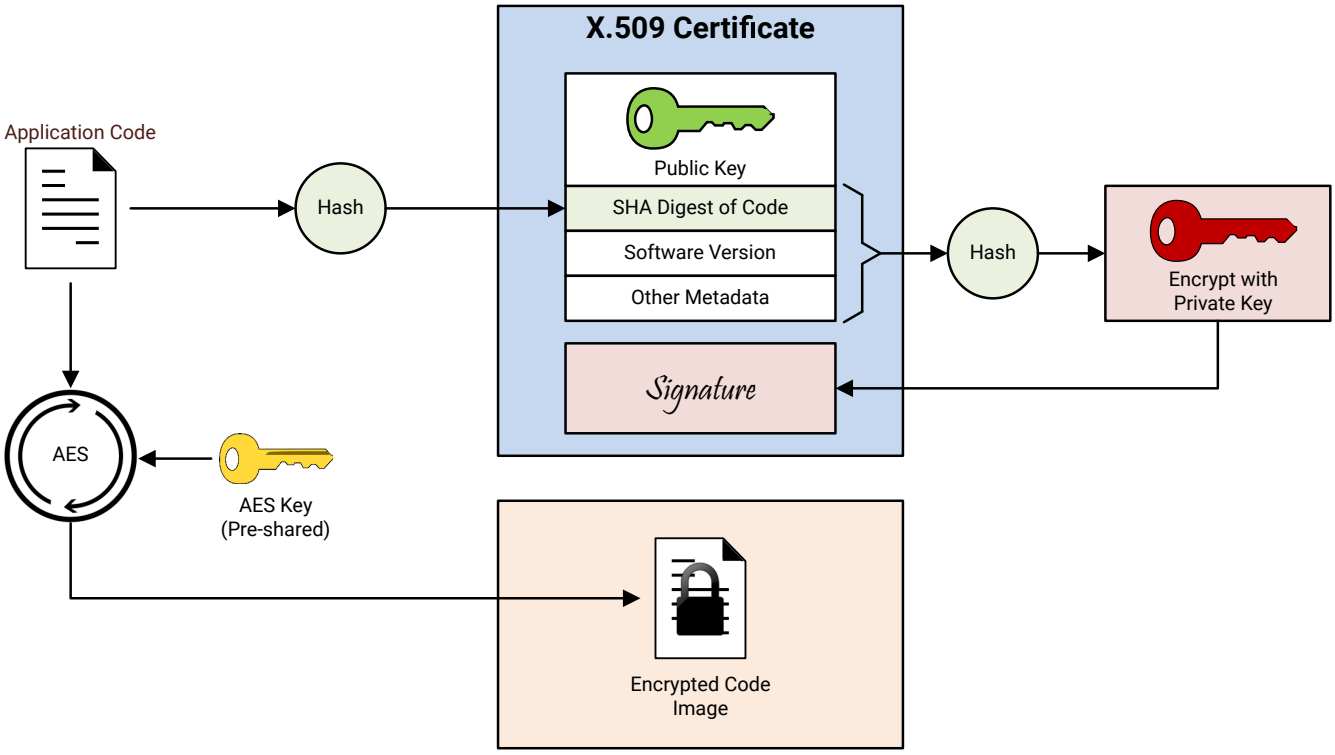


**图 3-1. 代码加密和数字签名示例**

非对称算法可用于使用私钥从消息创建数字签名。数字签名可用于确认已签名消息的完整性和真实性。但是，非对称算法比哈希函数慢得多。因此，通过解决大量数据的计算时间问题，哈希算法与非对称算法结合使用时效果很好。首先可以计算块的安全散列，而不是对整个块签名。然后使用发送方的私钥对生成的输出哈希摘要进行签名。接收方使用发送方的公钥反转此过程，计算接收到的数据块的哈希值，并根据原始哈希值对其进行身份验证。此过程可确定数据的完整性和真实性，同时节省计算时间，通常称为数字签名。这些项目通常与其他重要元数据一起存储在数字证书中，使用业界通用格式（如 X.509）。此过程对于出厂配置和固件更新至关重要，后续章节将对此进行讨论。

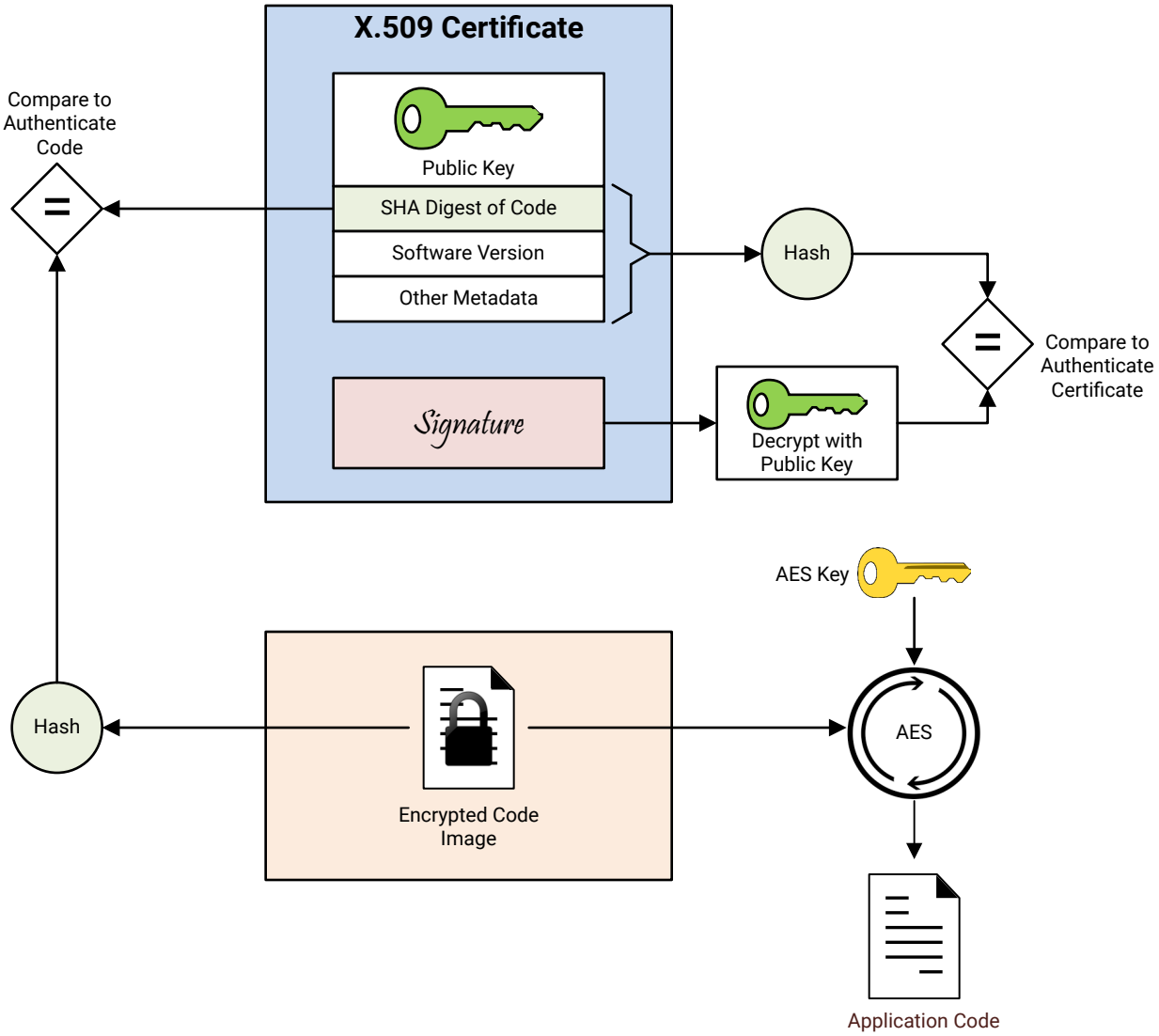


**图 3-2. 身份验证和解密示例**

加密身份验证方案还可用于应用本身，以实现运行时安全，验证在器件之间传输的数据的完整性和真实性。使用密钥、消息身份验证代码或 MAC 通过加密算法进行计算，然后附加到发送给接收器的消息中。然后，接收方可以使用相同的密钥和算法根据接收的消息计算 MAC，并将其与发送方发送的消息进行比较。如果两个代码匹配，则验证消息为真实且未更改。

两种最常用的加密身份验证方案是 CMAC 和 HMAC。CMAC 表示基于密码的消息身份验证代码，基于对称算法（如 AES）。HMAC 表示带密钥的哈希消息验证码，使用 SHA-256 等哈希函数。在为电子控制单元（即 ECU）之间传输的数据采用 AUTOSAR 安全车载通信 (SecOC) 架构的现代汽车系统中，可以找到加密消息身份验证的示例。SecOC 使用 CMAC 对通过车辆网络（如 CAN、FlexRay 或以太网）传输的网络消息提供端到端保护。在 SecOC 架构中，每个消息帧都包含一个安全标头和标尾，其中包含 MAC 和其他安全元数据。每个 ECU 都可以使用 MAC 来验证接收到的每个消息，共享密钥可以由中央主机管理和定期分发以保持新鲜度。此类方案可用于防范汽车窃贼经常使用的 CAN 注入等攻击方法。

## 3.3 随机数发生器 (RNG)

随机数生成是许多加密服务的重要元素。随机数用于初始化加密序列、生成密钥、创建身份验证质询等。但是，如果随机数来自可预测或缺少足够熵的来源，则可能会成为一个弱点，可以利用它打破加密并泄露机密。为了提高计算效率，许多现代系统都采用了假随机数生成器 (PRNG)，有时也称为确定性随机位生成器 (DRBG)。假随机数生成器使用数学算法生成确定性的数字序列，但依赖于初始随机种子。真随机数生成器 (TRNG) 使用随机度的

物理来源（例如噪声或量子现象）来生成真正随机和独立的位，但通常比 PRNG 慢得多。一种常见做法是使用 TRNG 提供高熵随机种子来初始化 PRNG，然后将其用于为加密应用生成随机数。

# 4 建立信任根

如前所述，表面上保护芯片边界不足以确保代码完整性，因为应用会以多种方式被恶意软件破坏或注入。建立信任根可提供两个重要好处来弥合这一差距：

- 确保器件始终使用可信代码启动
- 如果应用受到破坏，则使设备能够返回到受信任的代码，从而防止恶意软件继续运行。

信任根是嵌入式系统安全的一个基本概念。它是指一组执行关键安全功能并受到系统其余部分信任的硬件、固件和软件组件。这些关键功能可以包括加密密钥等机密信息的存储、辅助密钥和用户代码的身份验证和证明以及加密服务。信任根构成信任链中的第一个链接，可确保整个代码执行过程中应用软件的完整性。包含信任根功能的存储器元素必须是不可变的，即不可更改和不可修改。ROM、电子保险丝和一次性可编程或永久锁定的闪存就是不可更改存储器的示例。

## 4.1 机密的安全存储

安全信任根的一个重要组成部分是安全存储。安全存储可确保对设备的运行和安全至关重要的数据资产的机密性、完整性和可用性。这些数据资产可能包括加密密钥、证书、设备配置设置等。安全存储可保护关键资产免遭未经授权的访问或修改，防止可能危及应用系统整体安全性的数据泄露、篡改或损坏。为了建立安全的信任根，需要安全存储加密密钥和凭据。在安全微控制器中，这些资产通常存储在非易失性存储器（如闪存或电子保险丝阵列）中。此外，还需要阻止运行时应用程序软件直接读取或写入访问的硬编码保护，限制对不可变固件（如 ROM 代码或硬件加载程序）的访问。

## 4.2 保持密钥和代码安全性

许多用户在部署嵌入式系统时面临的一个挑战是如何在整个制造过程中保持代码、机密和知识产权的安全性。称为安全配置的过程涉及在不安全环境中将机密加密密钥编程到微控制器中。这些密钥随后用于对传入的应用程序代码进行身份验证和解密。在许多情况下，配置过程中由第三方制造和编程设施负责；即使签署了保密协议，恶意行为者仍有可能截取和窃取机密，或在微控制器上安装受到攻击的软件。因此，必须建立一个过程来在所有阶段保持密钥、证书和代码的机密性、完整性和真实性，从获取这些资产的服务器开始，到成功解密和对微控制器中的资产进行编程。

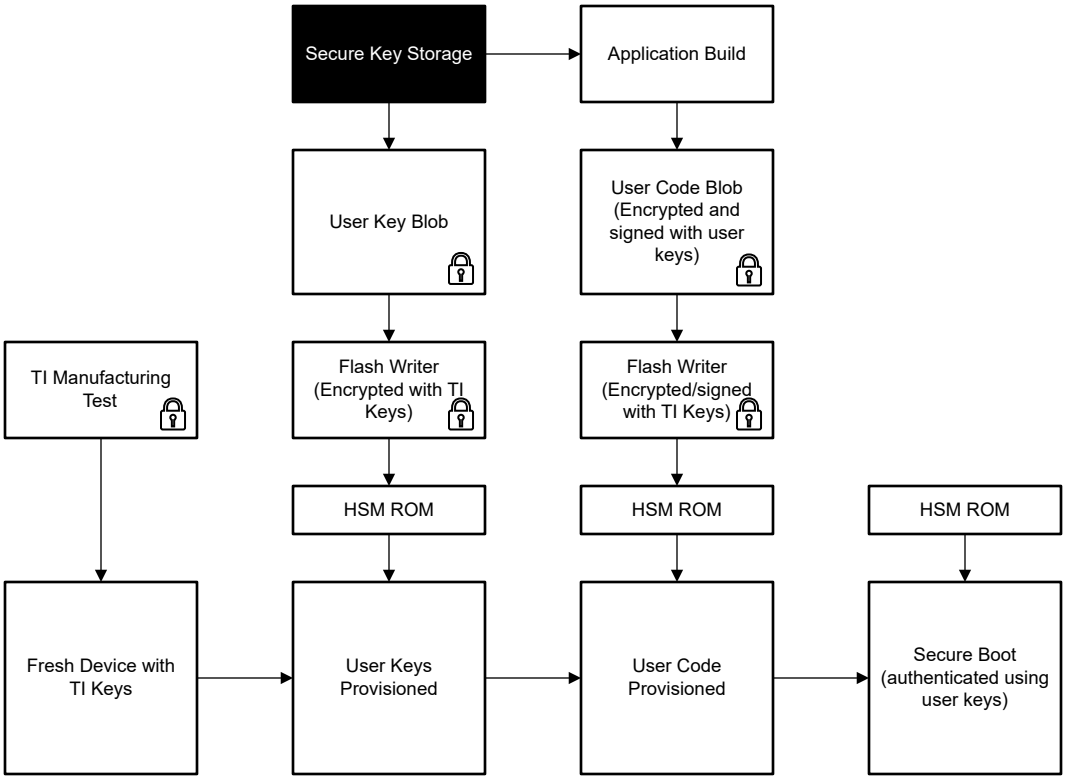


**图 4-1. 通过安全配置流程建立信任根**

安全配置流程从用于获取加密密钥和数据的 IT 基础设施开始。用于执行加密和解密的密钥必须存储在安全容器中，并在只能由密钥人员访问的可信执行环境中进行处理。在这种可信执行环境中，准备了用户密钥配置包，使用芯片制造商的密钥对用户密钥进行加密和签名。然后，将该软件包安全地传输到工厂并编程到器件中，从而在整个过程中对用户密钥保密。配置用户加密密钥后，可将应用程序代码编程到器件中。此过程与密钥配置过程类似，主要区别在于，用户代码现在使用之前编程到器件安全存储中的用户密钥进行加密和签名。对于具有内部闪存存储器的器件来说还有一个额外的安全优势：代码保持加密状态并且在被编程到器件中之前不可访问。然后可以对其进行解密并以纯文本形式存储，以更大限度地提高代码执行性能，因为使用器件级安全控制措施来保持机密性。另一方面，依赖于外部闪存芯片的器件可以使用每个器件特有的对称密钥来对外部闪存和内部 RAM 之间传输的代码进行加密和解密。为了加强安全性，在运送到客户编程设施之前，可以在器件中预先编程一个唯一的客户 ID。此唯一 ID 可进一步用于验证正版器件，并协助消除恶意第三方创建未经授权克隆的风险。

## 4.3 安全启动

嵌入式微控制器或处理器上的启动过程代表攻击者有机会破坏系统的安全性。因此，确保启动过程的安全对于为系统运行时操作建立信任根至关重要。借助使用配置流程安全地编程到器件中的加密密钥和证书，器件硬件和片上固件可以在开始执行应用程序代码之前验证编程到器件闪存中的应用程序代码、安全配置设置和其他数据的完整性。为了使此过程安全地建立信任根，用于执行安全启动操作的所有元素（包括片上固件）都必须是不可更改的。此外，芯片架构必须设计为在每次启动或芯片复位时始终执行安全启动过程。

在芯片启动时，器件首先从非易失性安全存储介质安全地将加密密钥和证书加载到受保护的存储器中。然后，片上启动代码的执行将在任何外部元件（例如调试器）无法访问和不可中断的隔离式可信安全环境中开始。此启动代码解密并验证代码证书的完整性，然后使用证书对应用程序代码和运行时安全设置进行身份验证。成功验证代码完整性会建立信任根，可以将控制权传递给应用程序，以便开始执行。相应地，应用程序还可以执行进一步的诊断和完整性检查，或使用片上加密服务通过质询响应方案对调试和外部通信接口进行身份验证。

## 5 安全执行环境

在器件上建立信任根通常需要创建一个安全的执行环境，有时也称为可信执行环境 (TEE)。安全执行环境背后的基本理念是创建单独的代码执行域（或“世界”），这些域对系统资源具有不同的权限和访问权限。安全执行环境可以保护正在使用的代码和数据免受恶意软件或硬件未经授权的修改、提取或篡改。硬件安全模块 (HSM) 是安全执行环境的一个典型示例，专用于为主机微控制器或处理器提供加密服务、安全存储、信任根和身份验证。TI 的 AM26x 和 F29x 微控制器包括一个具有这些特性的内置 HSM，可为应用程序实现安全密钥和代码配置过程、安全启动、调试身份验证和加密服务。

在许多情况下，仅建立信任根还不够，因为现代应用必须处理互联环境中的潜在网络安全威胁。外部通信接口（如 CAN 总线或 UART 端口）可能容易受到攻击，而攻击的结果会损害运行这些接口的软件任务的完整性。鉴于这些威胁的不可预测性，完整的网络安全威胁评估必然包括假设存在被攻破的代码模块及其对整个系统的机密性、完整性、可用性和安全性造成的风险。因此，运行时应用程序安全性是嵌入式系统中网络安全实现的一个重要方面。

由于所需的额外处理负担（通常意味着增加延迟和降低整体控制环路性能），基于软件的运行时安全措施在实时控制系统中是不可取的。此外，基于软件的解决方案是不可改变的；被攻破的代码可能导致整个运行时安全装置失败。基于硬件的解决方案通常从某种类型的存储器保护单元 (MPU) 开始。MPU 使应用开发人员能够定义固定存储器区域，并根据启动器尝试访问每个区域的情况来配置其访问权限（读取、写入和执行）。这些启动器可以包括 CPU、DMA 和调试器。在 TI F29x 系列微控制器中，内存和外设访问保护对上下文敏感。根据每个应用程序代码模块所在的地址范围，应用程序可以分为多个代码模块。然后，属于代码模块的任何数据范围或外设都可以与其他代码模块共享，并定义了单独的读取或写入权限。这些硬件存储器保护特定于执行存取的代码模块，因此无需由管理 MPU 的软件操作系统层，从而在不影响性能的情况下保持代码和数据安全。

除了存储器访问保护之外，CPU 内部数据的安全性对于建立安全的执行环境也至关重要。入侵者可能在 CPU 执行代码时窥探 CPU，从 CPU 寄存器和共享栈内存中读取机密。处理该问题的一种方法是基于软件：操作系统层中的任务调度器负责使不同的应用程序任务或线程相互隔离。在切换到新任务时，操作系统会通过保存、清除和恢复寄存器来维护线程上下文。这种方法的主要缺点是，对实时处理应用程序非常关键的中断服务例程不能使用任务调度器直接隔离。因此，帮助隔离 CPU 内应用程序上下文的硬件特性是在嵌入式实时系统中实现运行时安全的关键。通常，这涉及到硬件处理栈指针，在“安全”和“非安全”领域之间进行区分，或多个完全隔离的栈（在 C29x 等 CPU 架构中）。通常，这种类型的 CPU 指令集包括需要跨越栈边界的门指令。门指令可以触发寄存器归零等进程，并提供一个相关的访问保护方案来指定可以写入或读取哪些存储器。在 Arm® TrustZone®-M 等架构中，通常需要 trampoline 函数或 veneer 函数来在从一个上下文转换到另一个上下文期间处理这些保护更改。

另一方面，德州仪器 (TI) C29x 主要处理硬件中的上下文切换，实时应用新的保护以更大限度地提高控制性能。

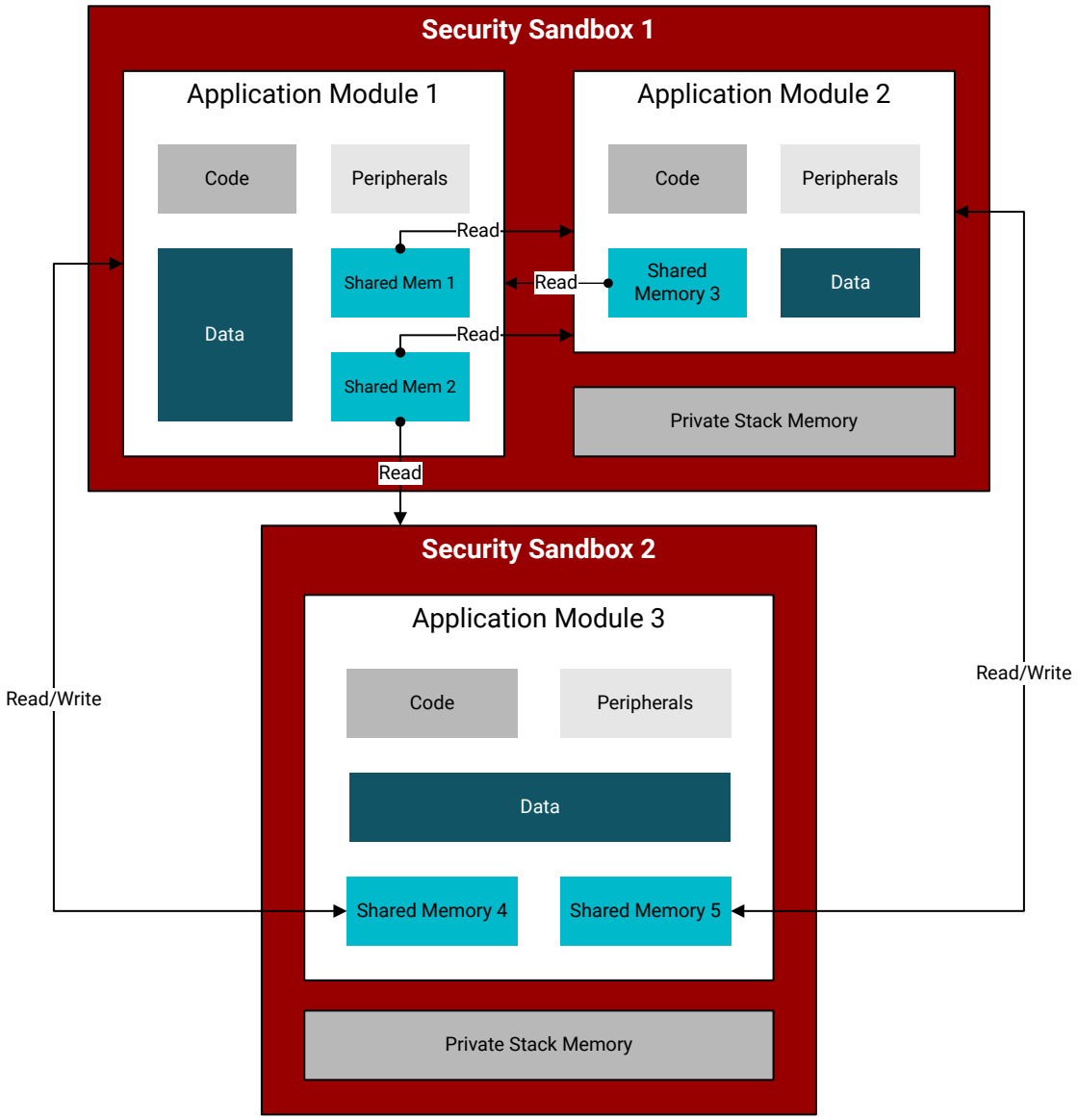


**图 5-1. 应用程序运行时安全分区示例**

通过在 CPU 中启用对代码和数据的硬件保护，嵌入式系统设计人员能够将应用划分为安全沙盒和应用程序任务模块，每个沙盒和应用程序任务模块都有自己的私有代码、数据和外设。此外，每个任务模块还可以选择与其他任务共享存储器区域。图 5-1 展示了一个安全分区方案示例。实施此类方案通常需要实时操作系统层，但是借助硬件中的隔离式栈和上下文感知存储器访问保护，系统设计人员可以实现完全隔离，同时避免软件操作系统层附带的额外延迟。

# 6 安全对策

为嵌入式系统实施网络安全策略的一个重要方面是确定系统中的漏洞，列举潜在的攻击类型和场景以及实施网络安全控制来减轻这些攻击。嵌入式控制系统往往更容易受到需要对器件进行本地或物理访问的攻击技术的攻击。其中包括与调试端口的连接、故障注入攻击（例如电源或时钟干扰）和其他侧通道攻击。可靠的嵌入式安全实施可识别这些不同的攻击情形并实施应对措施以减轻这些攻击。

原则上，故障注入攻击会尝试通过在器件的电源电压或系统时钟信号中引入临时异常来重定向 CPU 执行。例如，每个器件数据表都包括电源轨（例如内核、I/O 和模拟）的额定工作电压范围以及工作温度范围。这意味着器件被设计为可在这些电压和温度范围内正确运行并满足所有时序要求；在指定范围之外运行器件可能会导致未指定的行为。在实践中，违反这些规格通常会导致与时序相关的故障：欠压状态会导致违反设置时间，过压状态会导致违反保持时间。或者，可以使用可访问的输入时钟信号（如晶体振荡器输入引脚）在关键时间点直接注入时序故障。定时故障可能会导致在固件中的关键安全决策点跳过指令或重定向执行，从而导致未经授权的访问或暴露嵌入式机密。获取用于执行所需时序分析以成功执行故障注入攻击的工具越来越容易。针对这些类型攻击的对策可能包括内部电压和时钟频率监控电路。

为了防止故障注入攻击，即时检测单位或双位故障的能力是一种很有价值的工具。纠错码逻辑 (ECC) 就是此类保护机制的示例，能够自动纠正 single-bit 故障并检测存储器或总线上的双位故障。该机制使用预先计算的代码（通常随每个数据字一同写入存储器）而运行。ECC 通常用于功能安全环境，但也可用作安全对策，使系统拒绝注入故障或停止执行并发送错误信号进行响应。在 TI 的 AM26x 微控制器中，所有 ECC 保护适用于所有片上存储器和高速缓存，也适用于各种外设子系统和互连。F29x 微控制器具有直接内置于 C29 CPU 和器件互连中的 ECC 保护，在应用程序运行时针对所有存储器和外设的代码和数据故障提供强大的端到端保护。

# 7 调试安全性

攻击者破坏嵌入式控制系统的常用方法是使用 JTAG、SWD 或其他标准协议连接到微控制器的调试端口。如果调试端口不安全，攻击者可能很容易重定向执行、提取密码或探索其他方法来破坏系统。安全应用中使用的微控制器应根据基本要求对调试端口进行基于密码的锁定。为了提高安全性，如果系统设计人员希望避免使用密码（这些密码可能作为共享机密公开），则可以实施替代的身份验证机制，例如使用公钥加密的质询响应或基于证书的身份验证。

在某些情况下，需要将应用划分为多个域，可能将其中一个或多个域的开发外包给第三方或其他不可信的软件开发环境。在这种情况下，从安全角度来看，拥有一个在所有开发方之间共享的身份验证凭据并不理想。为了克服这一缺陷，器件架构可以使用独立的调试使能信号对芯片上的资源组进行分区。然后，可以为给定设备创建多个身份验证凭据，并对设备资源组进行不同级别的访问。例如，一个调试证书可授予对整个芯片的访问权限，而另一个证书会阻止对包含敏感数据的存储器区域的访问。

## 8 结语

随着汽车和工业实时控制系统变得越来越复杂且互连程度越来越高，网络安全威胁的范围和巧妙程度也在不断增加。因此，现代威胁形势推动了对实时微控制器产品中更严格网络安全功能的需求，而不会影响环路处理性能。德州仪器 (TI) C2000™ F29x 和 Sitara™ AM26x 微控制器满足此需求，使嵌入式设计人员能够为现代应用构建高度安全的实时控制系统。这些产品包含一个嵌入式硬件安全管理器 (HSM)，可提供固件保护、安全调试身份验证、安全存储和现代加密功能。用户加密密钥和代码的安全配置、安全启动和安全固件更新功能可用于建立信任根，并保持用户代码和资产的机密性和完整性。此外，TI C29x CPU 的独特架构可在运行时为应用实现动态上下文感知安全性，从而将安全执行环境的概念扩展到 HSM 的边界之外。借助这种运行时安全模型，通信栈等面向外部的应用功能可与关键系统功能隔离，从而降低由外部威胁对系统整体安全性带来的风险。借助此完整的平台安全解决方案，用户可以开发符合最高网络安全标准的高性能实时控制系统。

如需了解德州仪器 (TI) 的嵌入式微控制器（包括 C2000 和 Sitara 产品系列），请访问 ti.com/mcu。有关产品网络安全的详细信息，请访问 ti.com/security。

# 重要声明和免责声明

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 产品的应用。严禁对这些资源进行其他复制或展示。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