Technical White Paper

揭开面向嵌入式系统的后量子密码学之谜

Walter Schnoor

摘要

什么是 PQC？

后量子密码学 (PQC) 是一类密码原语，旨在抵御同时拥有经典计算能力以及密码相关量子计算能力的对手。

内容

1 简介.....2
2 嵌入式系统中的安全目标.....2
3 密码原语.....2
4 CRQC 的潜在影响.....2
5 面向未来的新算法和标准.....3
6 CNSA 合规性.....3
7 对称加密与哈希函数.....3
8 PQC 数字签名：ML-DSA.....3
9 硬件加速是否必要？.....4
10 预哈希是否必要？.....6
11 其他 NIST 选定的 PQC 数字签名方案.....6
12 有状态数字签名方案.....7
13 密钥协议如何？.....7
14 面向未来保护嵌入式系统.....8
15 参考资料.....9

商标

Arm® and Cortex® are registered trademarks of Arm Limited.
所有商标均为其各自所有者的财产。

English Document: STDA060
Copyright © 2026 Texas Instruments Incorporated

1 简介

虽然后量子密码学听起来可能很神秘，但让您的嵌入式系统面向未来、抵御未来具备量子能力的对手，并不一定也要如此神秘。这是因为，虽然 PQC 改变了用于实现安全的底层算法，但并不会改变您的高层安全目标。换言之，“做什么”仍然相同——改变的是“怎么做”。本白皮书解释了 PQC 如何与嵌入式系统中的常见安全目标相契合、与 PQC 相关的行业标准，以及对使用微控制器的资源受限嵌入式系统的影响——即它对延迟、内存和通信时间意味着什么。

2 嵌入式系统中的安全目标

要了解 PQC 对嵌入式系统的影响，最好从安全目标本身开始。许多嵌入式系统共享一组共同的安全目标，包括：

- 安全启动，确保处理器上运行的软件或固件是真实的——换句话说，系统制造商授权了该软件或固件。
- 安全更新，为保护知识产权而使新的软件和固件更新保持机密。
- 证明，使设备或系统能够证明其知道某个秘密，从而向云服务器或系统中的另一个模块证明其是真实的且可信的。
- 安全通信，保护嵌入式系统中设备之间或嵌入式系统与远程服务器之间消息的机密性、完整性和真实性。

在密码系统中部署密码原语可实现这些安全目标。例如，电子控制单元中的微控制器可以通过包含不可变的信任根固件来实现安全启动，该固件使用数字签名算法在允许应用软件于 CPU 上运行之前对其进行认证，从而防止在 CPU 上执行未授权代码。如果底层原语将来被攻破，那么诸如安全启动之类的安全目标也会随之被攻破。

3 密码原语

密码原语是构建密码系统所需的密码学核心构建块。密码原语的一个例子是对称分组密码，它使用单个秘密密钥将明文加密为密文，并将密文解密回明文。另一个例子是数字签名算法，它使用两个密钥验证消息的真实性：一个用于将消息签名为真实的秘密密钥，以及一个用于验证消息是由匹配的秘密密钥签名的公钥。这些原语的安全性源自所谓的困难性假设。

当今使用的最流行的数字签名算法之一是椭圆曲线数字签名算法 (ECDSA)，其安全性基于椭圆曲线离散对数问题 (ECDLP) 的困难性。目前，没有已知算法能够使用当今计算机高效求解 ECDLP。因此，ECDLP 被认为“足够困难”，可以依赖其提供安全性。然而，量子计算机则是另一回事。

4 CRQC 的潜在影响

传统计算指当今的计算机，它们将定义好的操作作为一系列离散指令来运行。在某些情况下可以并行化这些指令，但计算仍限于顺序操作或并行的顺序操作集合。

另一方面，量子计算机可以利用同时既为“0”又为“1”的数据位，使它们能够以当今计算机无法做到的方式同时搜索解空间[1]。

密码学相关量子计算机 (CRQC) 理论上能够通过使用已知算法求解 ECDLP。如果这种量子计算机存在，它将打破使当今流行密码原语（如 ECDSA）安全的困难性假设。对手可能攻破该密码原语，从而破坏该密码原语所支撑的安全目标。攻击 ECDSA 的一个主要目标是恢复用于将新消息签名为真实的私钥，以便对手随后能使用该私钥签署欺诈性消息并产生有效签名。这要求在给定公钥 P 和公共生成点 G 的情况下找到私钥值 k，其中在椭圆曲线上 $P = kG$。对于传统计算机，在给定公开值 P 和 G 的情况下恢复 k 是困难的（已知方法以指数时间运行）。借助 CRQC，可能可以运行 Shor 算法：一种量子算法，它将攻击从指数时间降至多项式时间，使其可实际执行。在撰写本文时，尚不存在规模足够大且足够稳定、能够执行可攻破流行 ECDSA 实现（例如使用美国国家标准与技术研究院 [NIST] P-256 曲线的 ECDSA）的算法（如 Shor 算法）的量子计算机，但未来可能会存在。

尽管无人知道何时会出现足够强大、能够破解当今经典密码原语的量子计算机[1]，但许多行业正因以下原因开始保护其产品免受未来威胁：

- 构建可信的新密码学需要时间，将其集成到产品中也需要时间。
- 许多嵌入式系统在现场使用寿命长达 10 年或更久。现在应对 CRQC 可能被发明的情况可降低未来安全丧失的风险。
- 许多市场正在推动监管，尤其是企业和数据中心市场。

因此，我们看到越来越多的嵌入式系统设计人员转向 PQC 密码原语以实现安全目标。

English Document: STDA060
Copyright © 2026 Texas Instruments Incorporated

5 面向未来的新算法和标准

大多数嵌入式系统的安全目标与用于实现这些目标的密码原语（经典或量子）无关。采用 PQC 的动机不是改变嵌入式系统的安全目标；而是改变用于实现安全目标的底层运算，使困难性问题能够抵御具备 CRQC（密码相关量子计算）能力的攻击者。

6 CNSA 合规性

在美国，用于国家安全系统的产品必须符合美国国家安全局 (NSA) 的商业国家安全算法 (CNSA) 套件。2022 年，NSA 将 CNSA 套件从 1.0 版更新到 2.0 版；CNSA 2.0 的主要动机之一是纳入 PQC。CNSA 套件的新部署必须在 2027 年 1 月 1 日前符合 2.0 版，而已经部署的系统必须在 2030 年 12 月 31 日前符合要求，或被替换为符合 CNSA 2.0 的系统。

表 6-1 列出了 CNSA 1.0 批准的密码原语与 CNSA 2.0 要求批准的密码原语之间的对比。

表 6-1. 比较 CNSA 1.0 与 CNSA 2.0 批准的密码原语

应用用途	密码原语	通过 CNSA 1.0 认证	CNSA 2.0 批准 [2]
通用加密	对称加密	联邦信息处理标准 (FIPS) 出版物 (PUB) 197（仅限高级加密标准 [AES]-256）	
	安全哈希	FIPS PUB 180-4（仅限安全哈希算法 [SHA]-384）	FIPS PUB 180-4（SHA-384 或 SHA-512）
	数字签名	FIPS PUB 186-4（ECDSA P-384 或 Rivest、Shamir 和 Adleman 3072 位模数）	FIPS PUB 204（仅模块格基数字签名标准 [ML-DSA]-87）
	密钥协商	椭圆曲线 Diffie-Hellman P-384 或有限域 Diffie-Hellman（3072 位模数）	FIPS PUB 203（仅模块格基密钥封装机制 [ML-KEM]-1,024）
仅固件或软件签名	数字签名	-	NIST 专用出版物 (SP) 800-208（Leighton-Micali 签名 [LMS] 或扩展 Merkle 签名方案 [XMSS] [1]）
仅硬件完整性检查	安全哈希	-	FIPS PUB 202（SHA-3-384 或 SHA-3-512）

7 对称加密与哈希函数

CNSA 2.0 并未实质性改变对称加密原语和安全哈希函数的最低要求。例如，CNSA 2.0 对对称加密 (AES-256) 保留了与 CNSA 1.0 相同的算法和安全级别。这是因为目前 CRQC（密码相关量子计算机）对 AES-256 不构成重大威胁。尽管存在可以降低 AES 安全性的量子算法，但 CNSA 1.0 和 2.0 中选择的密钥长度（256 位）足够高，使该原语仍然安全。以 Grover 算法为例。Grover 算法是一种量子算法，可将对 AES 的暴力攻击时间从 2k 降低到 2k/2，其中 k 是以位为单位的密钥长度。在最理想的情况下，使用 Grover 算法时 AES-256 的安全强度会从 2^{256} 降至 2^{128}，这仍被认为是安全的。相比之下，使用 AES-128 的设计其安全强度可能从 2^{128} 降至 2^{64}，这可能是脆弱的。因此，在目前使用 AES-128 的应用中，建议切换到 AES-256，从而抵消量子计算机带来的优势。

CNSA 2.0 中一些最重大的变化与数字签名和密钥协商有关。

8 PQC 数字签名：ML-DSA

对于通用数字签名，包括固件和软件更新签名，CNSA 2.0 要求使用 ML-DSA。ML-DSA 的安全性依赖于模学习带误差问题的困难性，并且被认为能够抵御同时具备经典计算能力和量子计算能力的对手。ML-DSA 的设计基于提交给 NIST 的 Crystals-Dilithium 签名方案；经过三轮分析和评估后，NIST 选择将其标准化为 FIPS 204 [4]。

在使用 ECDSA 进行数字签名的嵌入式应用中（包括安全启动、证明及其他签名用例），ML-DSA 提供了一种具有类似编程接口的后量子安全替代方案。ML-DSA 还具有可扩展性；它支持三种不同的安全级别（ML-DSA-44、

English Document: STDA060
Copyright © 2026 Texas Instruments Incorporated

ML-DSA-65 和 ML-DSA-87），每种级别都在密钥和签名大小、计算延迟、计算所需静态随机存取存储器 (SRAM) 空间以及安全性之间进行权衡。CNSA 2.0 强制要求使用最高安全级别：ML-DSA-87。

对于嵌入式系统开发者而言，ML-DSA 与 ECDSA 在以下方面存在显著差异：

- 与 ECDSA 相比，ML-DSA 的密钥和签名尺寸要大得多。ML-DSA-87 的签名大小（4,627 字节）是 ECDSA P-384 签名大小（96 字节）的 48 倍以上，因此需要为闪存和 SRAM 预留更多空间，并且在执行证明功能或安装新的软件和固件更新时，线上传输时间也更长。
- 在比较基于 CPU 的实现时，ML-DSA 验证往往比 ECDSA 验证更快，从而缩短认证时间。例如，ML-DSA-87 签名验证操作执行所需时间可达 NIST P-384 曲线上 ECDSA 的十分之一，从而显著改善微控制器的启动时间。
- 与 ECDSA 相比，ML-DSA 实现通常需要更多 SRAM 来执行必要计算，并在工作内存和执行延迟之间具有不同的权衡。

表 8-1 比较了经典计算签名 (ECDSA P-384) 与后量子计算签名 (ML-DSA) 在 ML-DSA 各个安全级别下的情况。

表 8-1. ECDSA 参数大小和性能与 ML-DSA 的比较

参数	ECDSA	ML-DSA-44	ML-DSA-65	ML-DSA-87
安全级别	P-384	类别 2	类别 3	类别 5
密钥（字节）	48	2,570	4,032	4,896
公钥（字节）	96	1,312	1,952	2,592
签名（字节）	96	2,420	3,309	4,627
签名线上传输时间（秒）(1)	0.009	0.22	0.30	0.42
密钥生成时间（秒）(2)	1.12	0.084	0.14	0.23
签名时间（秒）(2)	1.23	≅0.4	≅0.8	≅1.6
验证时间（秒）(2)	**2.26**	**0.10**	**0.16**	**0.26**
验证所需 SRAM（千字节）(2)	2.2	21.9	29.1	37.6

(1) 在 100kHz 速率、每字节 9 比特条件下签名的传输时间
(2) 测试环境为运行 TI MSP 加密库的德州仪器 (TI) 80MHz Arm® Cortex®-M0+ 微控制器

9 硬件加速是否必要？

构建使用椭圆曲线密码学的嵌入式系统的开发者，如果应用对延迟敏感，可能会使用带有椭圆曲线计算加速功能的微控制器。然而，这可能导致所选微控制器比应用实际所需的更昂贵。

例如，在挑战-响应应用中，在 80MHz Arm Cortex-M0+ 处理器上为挑战-响应方案计算 ECDSA P-384 签名可能需要超过 2 秒，这远长于以 100Kbps（典型速率，假设采用标准 I^2C 或 115200 串行通用异步收发器等通信方式）在线上传输所得签名所需的 6 毫秒。

当转向 PQC 时，更大的数字签名和更短的计算时间会显著改变这种动态。例如，传输签名所需的线上时间增加了 48 倍，从大约 9 毫秒（对于 ECDSA P-384）增加到大约 420 毫秒（对于 ML-DSA-87）。签名验证计算时间从大约 2,200 毫秒（对于 ECDSA P-384）减少到大约 260 毫秒，这几乎是线上传输签名所需时间的一半。因此，诸如证明之类的应用往往更多受通信时间而非计算时间限制，从而使基于 CPU 的实现更具经济性。图 9-1 比较了 Arm Cortex-M0+ 和 Arm Cortex-M33 目标 MCU 上各安全级别的验证性能。

Copyright © 2026 Texas Instruments Incorporated

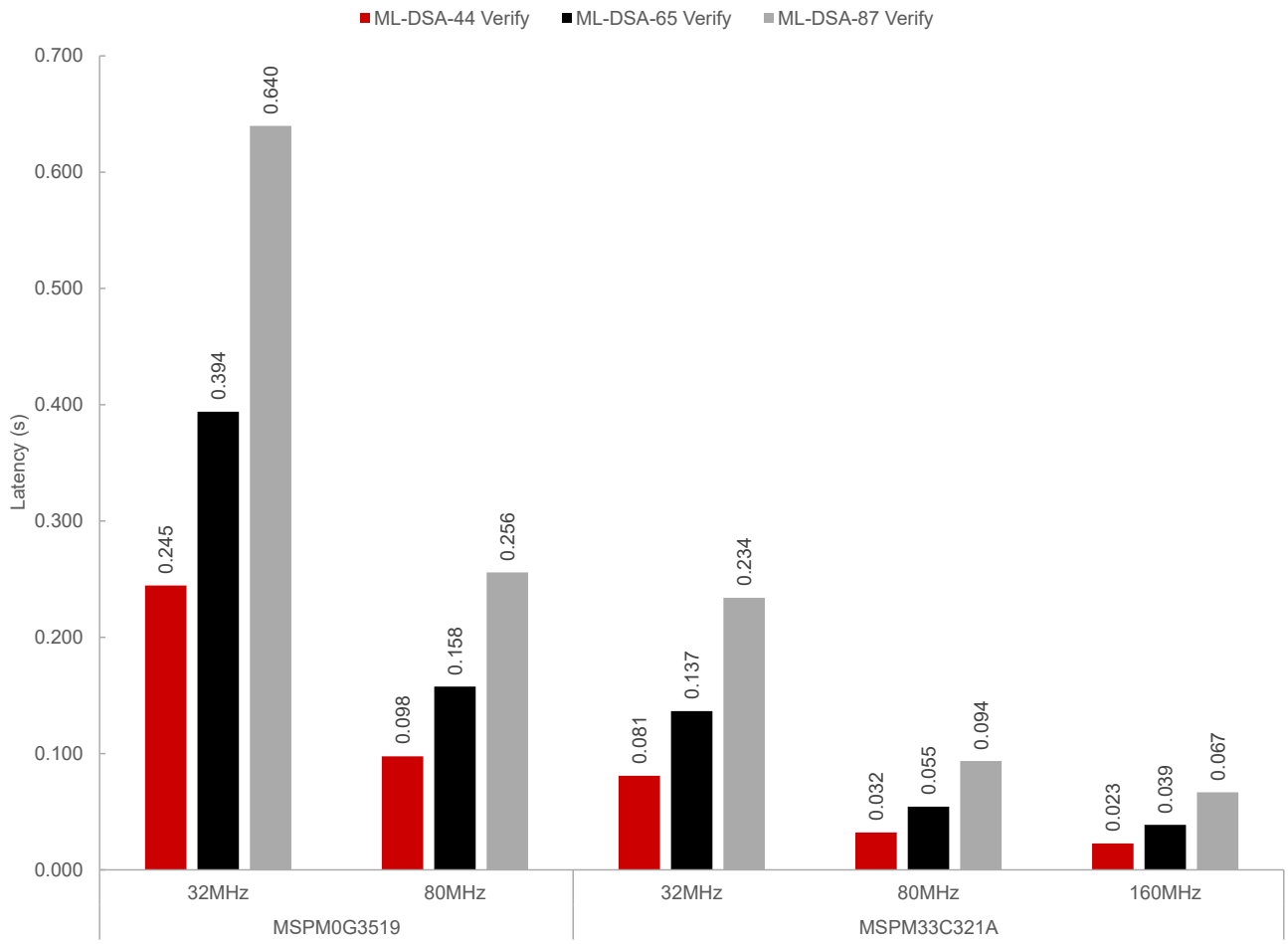


图 9-1. TI MSPM0、MSPM33 微控制器的 ML-DSA 验证性能

English Document: STDA060
Copyright © 2026 Texas Instruments Incorporated

10 预哈希是否必要？

在安全启动实现中，通常先使用 SHA-256 或 SHA-384 等安全哈希函数对应用软件和固件映像进行哈希，以创建固定大小的消息代表值，然后通过 ECDSA 对其进行签名和验证。ML-DSA 消除了这一“预哈希”步骤。事实上，ML-DSA 的设计者更希望该方案的用户将用于数字签名的整个消息内容直接传入签名算法本身。

在 FIPS PUB 204 中，此过程称为“纯”模式。FIPS PUB 204 还定义了一种称为 HashML-DSA 的预哈希模式，该模式包含有关计算中用于生成签名的哈希函数的信息（从而将哈希函数与签名绑定）。

除 FIPS 204 之外，还可以采用第三种选择：对消息进行预哈希，但使用纯模式下的 ML-DSA 对哈希进行签名，就像消息被直接传给 ML-DSA 一样，而不是使用 FIPS 204 的预哈希 HashML-DSA 模式。这省略了哈希函数与签名的绑定，但它与当今 ECDSA 与 SHA 一起使用的方式更具可比性。

ML-DSA 在生成和验证数字签名时在内部使用 Keccak 函数。Keccak 是 SHA-3 函数以及 SHA 和 Keccak (Shake) 可扩展输出函数内部使用的核心函数。ML-DSA 以多种方式使用 Keccak，包括对消息进行安全摘要以及在验证期间对公钥进行确定性扩展。

这很重要，因为在资源受限系统上，包含 ML-DSA 意味着也包含 Keccak，从而在同一代码占用空间内提供安全哈希函数。因此，使用纯 ML-DSA 操作模式意味着无需牺牲额外的非易失代码存储器来存储 SHA-384 或 SHA-512 等额外哈希函数。此外，Keccak 可用于将 SHA-3 作为通用哈希函数暴露出来，供应用中的其他用途使用。

目前，符合 CNSA 2.0 的系统可以使用经批准的安全哈希函数（SHA-384 或 SHA-512）对消息进行预哈希，但不能使用 FIPS PUB 204 规定的 ML-DSA 预哈希模式 (HashML-DSA)；相反，FIPS PUB 204 纯模式是默认模式。

11 其他 NIST 选定的 PQC 数字签名方案

除 ML-DSA 外，NIST 还选定了两种后量子计算数字签名方案进行标准化：

- 基于 Sphincs+ 签名方案的无状态哈希数字签名标准 (SLH-DSA)，标准化为 FIPS PUB 205。
- 基于 N 次截断多项式环单元格的快速傅里叶变换数字签名算法 (FN-DSA)，即 Falcon，正在被标准化为 FIPS PUB 206。

SLH-DSA 是一种基于哈希的签名方案。与 ML-DSA 一样，它是无状态的。与 ML-DSA 不同，其较高安全级别的签名尺寸非常大（达到数十千字节），这限制了 SLH-DSA 在嵌入式应用中的适用性，尤其是基于微控制器的应用。目前,CNSA 2.0 不允许基于 SLH-DSA 的签名,尽管 NIST 将 SLH-DSA 标准化。目前，CNSA 2.0 不允许基于 SLH-DSA 的签名，尽管 NIST 已标准化 SLH-DSA。

FN-DSA 有望提供比 ML-DSA 和 SLH-DSA 更小的签名，但代价是复杂性。FN-DSA 依赖浮点运算，而浮点运算在不同处理器架构上以恒定时间实现是出了名的困难。此外，并非所有嵌入式架构都包含浮点单元，从而抵消了性能优势。目前，CNSA 2.0 不允许 FN-DSA[3]，且 NIST 尚未标准化 FN-DSA。

由于这些原因，业界在很大程度上正转向将 ML-DSA 作为事实上的数字签名方案，因为它在性能、尺寸和安全性之间取得了良好平衡。

English Document: STDA060
Copyright © 2026 Texas Instruments Incorporated

12 有状态数字签名方案

ML-DSA 以及所讨论的方案都是无状态方案。对于给定私钥，在签名过程中不需要保持状态，这是嵌入式应用中备受青睐的特性，尤其是在资源受限环境中正常操作期间频繁发生挑战-响应签名的证明用例中。

在无状态方案被标准化之前，若干后量子安全的有状态签名方案为特定用例提供了选项，例如软件和固件映像的签名（安全启动），包括在 NIST SP 800-208 中标准化的 LMS 和 XMSS。

这些方案提供的较小公钥（小于 ML-DSA）对实现安全启动的嵌入式系统有益，因为它们在认证新固件时需要更少的嵌入式非易失性存储器（通常是闪存）。签名操作确实需要跟踪给定私有签名密钥的状态，以确保私钥的保密性。这对于固件和软件更新可能是可接受的，因为可以在受控的 IT 环境中为签名硬件安全模块维护状态，而在这种环境中固件和软件映像的更新并不频繁。然而，在嵌入式环境中这通常是不可接受的。

因此，LMS 和 XMSS 在符合 CNSA 2.0 的系统中是允许的，但仅用于固件和软件数字签名；它们不能用于其他用例，例如证明。相比之下，ML-DSA 可用于所有数字签名用例，包括安全启动和证明。

13 密钥协议如何？

密钥协商是在不安全信道上在两个实体之间建立共享秘密的必要步骤。与数字签名一样，依赖当今流行原语（如有限域 Diffie-Hellman (FFDH) 或椭圆曲线 Diffie-Hellman (ECDH)）的密钥协商算法也容易受到使用 Shor 算法的量子攻击。为解决这一问题，NIST 已在 FIPS 203 中标准化 ML-KEM（模块格密钥封装机制）。ML-KEM 是一种后量子安全的密钥封装机制 (KEM)，设计用于当今使用 FFDH 或 ECDH 的场景。

从握手双方随机性中派生共享秘密的 Diffie-Hellman 式设计不同，KEM 具有密钥生成步骤、封装步骤和解封装步骤。在客户端-服务器模型中，使用 ML-KEM 进行密钥协商的过程如下：

1. 服务器首先生成一个密钥对。封装（或“公共”）密钥分发给客户端。解封（或“专用”）密钥由服务器保留。
2. 当客户端希望与服务器建立安全信道并就对称密钥达成一致时，它基于随机种子执行 ML-KEM 封装操作，由此获得密文和共享秘密。然后将密文发送到服务器。
3. 服务器接收密文并执行解封装操作，得到与客户端相同的共享秘密。
4. 服务器和客户端现在可以使用共享秘密密钥进行通信。

ML-KEM 在软件中可以有良好性能，在 160MHz Cortex-M33 MCU 上封装时间低至 7ms。图 13-1 比较了 Arm Cortex-M0+ 和 Arm Cortex-M33 目标 MCU 上各安全级别的封装性能。

Copyright © 2026 Texas Instruments Incorporated

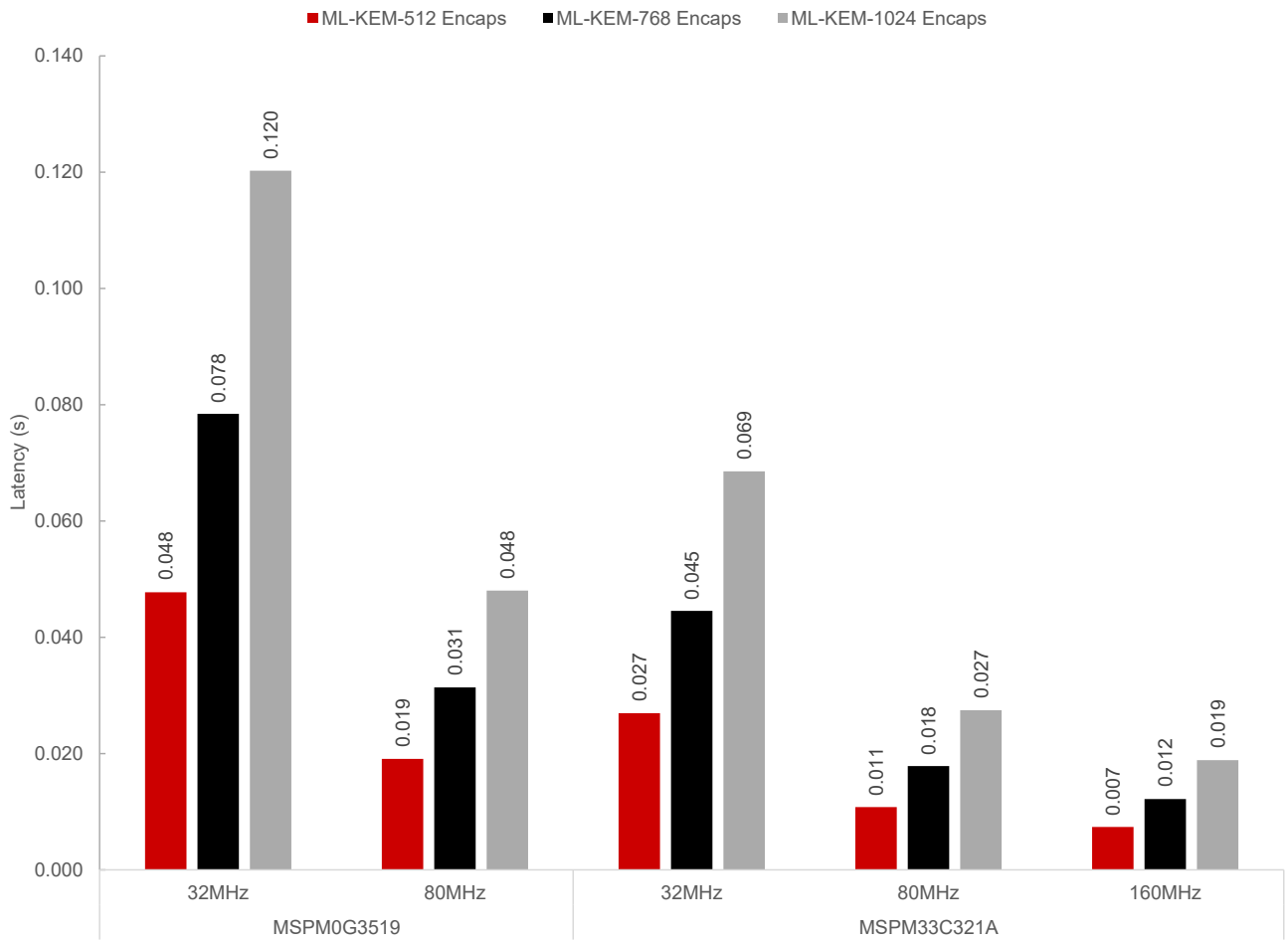


图 13-1. TI MSPM0、MSPM33 微控制器的 ML-KEM 封装性能

ML-KEM 除了被设计为量子安全之外，还具有其他理想的安全属性。值得注意的是，它支持 IND-CCA2（在选定密文攻击下的不可区分性）。这意味着，即使攻击者能够在从客户端到服务器的传输过程中（在封装与解封装之间）操纵密文，攻击者也不会获得优势或额外知识，因为解封装的输出始终与随机值不可区分。此外，ML-KEM 的输出可直接用作对称加密密钥，这为嵌入式应用带来了简便性，并可能消除额外密钥派生步骤的需要。

14 面向未来保护嵌入式系统

设计人员需要考虑这样一种可能性：攻击者未来可能拥有能力足够强的量子计算机，能够攻击用于提供基本安全目标（包括安全启动、证明和安全通信）的经典密码原语。通过 NIST 和 NSA，美国拥有一个框架，用于选择即使面对具备此类能力的攻击者也被认为安全的密码原语。

嵌入式系统中的挑战包括非易失存储和工作内存小、通信接口较慢以及存储状态的能力有限。TI 在超低功耗和低成本微控制器上提供对后量子安全密码原语的支持，这些微控制器可实现 CNSA 2.0 合规并平衡性能与内存，包括：

- 数字签名：完全支持预哈希形式和纯形式的 ML-DSA-44、ML-DSA-65 和 ML-DSA-87。
- 密钥封装：完全支持 ML-KEM-512、ML-KEM-768 和 ML-KEM-1024。
- 对称加密：支持 AES-256 及相关操作模式，包括计数器与密码块链接-消息认证码模式和伽罗瓦计数器模式。
- 安全哈希函数：完全支持 SHA-2 以及 SHA-3 和 SHAKE 套件。

English Document: STDA060
Copyright © 2026 Texas Instruments Incorporated

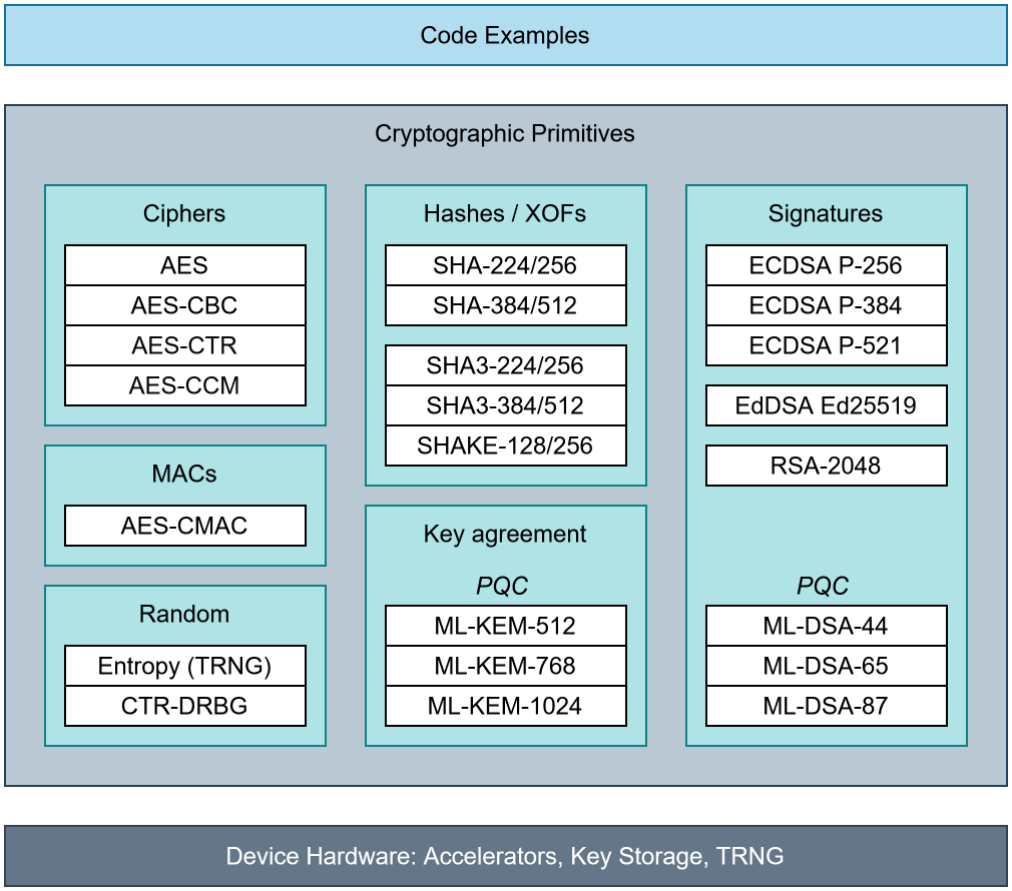


图 14-1. MSP-CRYPTO-LIB 特性

在基于 MSPM0 Arm Cortex-M0+ 和 MSPM33 Arm Cortex-M33 的微控制器上使用 TI MSP Crypto Library，可在不到 260ms (Cortex-M0+) 或 70ms (Cortex-M33) 内实现符合 CNSA 2.0 的 ML-DSA-87 数字签名验证，用于安全启动用例，即使在资源和成本受限的环境中也能提供快速启动时间，同时为量子计算成为现实的未来做好准备。

15 参考资料

1. NIST。2024 年 8 月 13 日。“什么是后量子密码学？”2026 年 2 月 3 日访问。
2. NSA。2022 年 9 月。“宣布商业国家安全算法套件 2.0。”2026 年 2 月 3 日访问。
3. NSA。December 2024.“商业国家安全算法套件 2.0 与量子计算常见问题解答。”2026 年 2 月 3 日访问。
4. NIST FIPS 204。2024 年 8 月 13 日。“基于模块格的数字签名标准。”2026 年 6 月 6 日访问。

Copyright © 2026 Texas Instruments Incorporated

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、与某特定用途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安全、安保法规或其他要求。

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。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 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。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

版权所有 © 2026，德州仪器 (TI) 公司

最后更新日期：2025 年 10 月