安全元件的三大热门固件验证用例（第一部分）

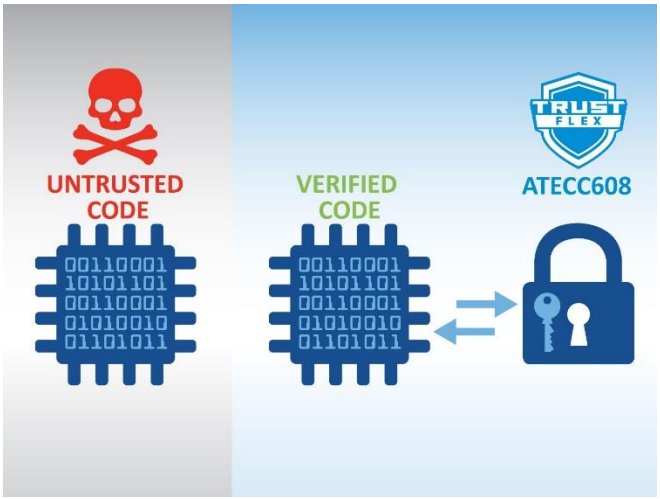
本系列分为两部分，第一部分将详细介绍在应用中实现固件验证的重要性以及所需的代码。

Microchip Technology Inc.

安全产品部市场营销经理

Xavier Bignalet

在当今市场上，嵌入式电子系统种类繁多，需要接受的认证也日益增加，如适用于工业应用的 IEC62443 安全标准，有时法规也取决于系统工作方式的关键特性。然而，要确保安全性，不仅需要依靠法规，还应该实施负责任的基本做法。今天我们来介绍一下小型和受限嵌入式系统的固件验证。每个嵌入式系统都基于其执行的代码（固件、软件和 RTL 等）运行。在本系列的第一部分中，我们将介绍当连接的心脏起搏器、工业网关、IoT 婴儿监视器，甚至工业机器电机的固件遭到攻击时，这些设备会发生什么。请留意第二部分，我们将在其中介绍如何在各种嵌入式系统上实现这些用例。



很多公司的知识产权（IP）处于产品的代码中。连接的心脏起搏器的代码可掌控人的生死，其重要性非同小可。同样，心脏起搏器也要接受认证。在工业泵中，代码可以处理速度和扭矩调整率，提高电机的性能，从而领先竞争对手。工业网关代码的架构能够以市场领先的速度处理复杂智能工厂网络中的海量有效载荷，但网关背后是一个覆盖多个机器和操作员的工业网络。现在，如果机器操作被更改，很快就会给工厂中的人员带来安全隐患。IoT 婴儿监视器代码可确保网络连接的稳健性，同时还能为父母提供与新生儿相关的隐私

信息。我们在下文中介绍了一家公司的代码对其 IP 至关重要的四个原因。公司应该考虑本文提到的各种现有标准所定义的威胁模式：

1. 远程数字攻击：漏洞是否在远程访问代码，以试图影响目标或整个产品群？
2. 远程逻辑攻击：漏洞是否在重点攻击代码中的某个错误，从而帮助黑客将攻击范围远程扩展到整个网络？
3. 本地物理攻击：如果黑客对产品有物理访问权限，可以对代码做什么（不一定要利用错误）？
4. 本地逻辑攻击：如果黑客对产品有物理访问权限，可以利用代码错误做什么？

近年来，为了解决上述问题，政府和许多行业已经着手制定 IoT 安全法规以创建相关标准。例如，在工业领域，ISA/IEC62443 标准载明了设计工业产品时要遵守的安全规范。欧洲的 EN303656 标准依据欧洲政府有关在当地消费市场销售 IoT 产品的安全法规和指南制定。UL2900 起初主要关注软件安全规范，现在大型公司正在将其用于消费市场。在所有的主要标准或法规中，您会发现，建议验证代码的真实性是其共同要求。

那么，可以采取哪些措施来保护这些代码？这些措施各有哪些优缺点？

代码将需要通过加密操作进行验证，以确保其真实性。验证可以在嵌入式系统操作过程中的不同时间点执行。

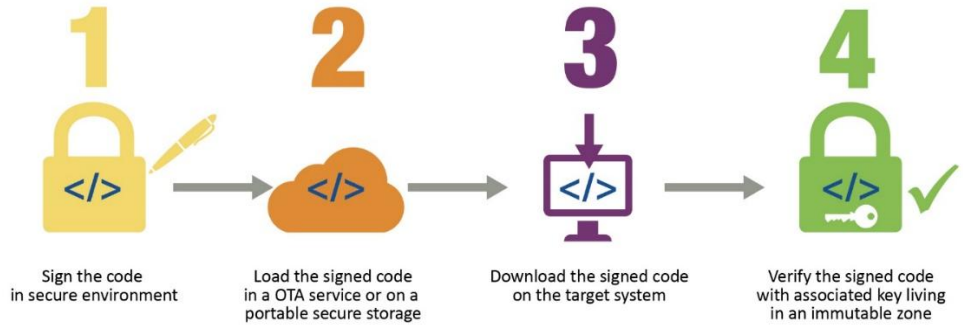
何时验证代码的真实性？

1. **引导时**：常见的技术术语是安全引导，它有多种实现方法：可以使用对称密钥，也可以使用非对称密钥，可以只验证摘要、签名，也可以验证整个代码映像。安全引导过程是为了确保只有真实代码可以在目标嵌入式设计上执行。
2. **运行时**：更通用的术语是 IP 保护。除了安全引导和无线（OTA）更新之外，固件工程师可以决定在系统运行期间的任何相关时间点执行代码验证，以确保代码没有被篡改。
3. **OTA 更新之后**：在 IoT 环境中，如果通过网络以“无线”方式推送新的代码映像来取代现有的代码，则在运行此新代码之前必须先验证其真伪。

上面描述的三种嵌入式安全功能总体上与代码验证规范有关。

现在，我们来介绍一下实现技术，然后再评价一下各自的优缺点。从根本上说，需要在安全性比较理想的企业环境中对代码进行“签名”，然后在嵌入式系统中对代码进行“验证”。这些“签名”和“验证”操作通过加密算法和对称或不对称密钥集实现。如下图所示，有四个主要步骤。第一步关注的是制造过程中发生的情况，以及代码加密和签名的处理方式。第

二步是如何将代码加载到嵌入式系统中（安全加载程序）。第三步涉及如何将代码下载到嵌入式系统中。第四步侧重于嵌入式系统制造完成后内部发生的情况，以确保在目标应用程序上运行的代码确实是真实的。



在制造过程中使用对称密钥进行签名操作

对称加密基于共享密钥架构，换言之，两个相同的密钥构成一个密钥对（对称密钥或也称为共享密钥）。主要缺点是，如果有人可以访问其中一个密钥，另一个共享密钥也是相同的，系统很容易遭到破坏。此外，基本做法要求为每个设备使用不同的对称密钥，这会产生逻辑悖论：如何在整个设备群中分配惟一对称密钥。因此，由于对称密钥易于实现和项目进度的限制，开发人员会在整个设备群中使用相同的对称密钥，因此这些密钥暴露的情况变得更加严重。接下来具体分析一下。第一步发生在公司环境中。

- 作为原始终端制造商（OEM），您的代码将通过“哈希”函数传递。我们在本文中以 SHA256 为例。此哈希函数的输出是代码映像的 32 字节摘要。
- 此哈希使用对称密钥（用于签名的 OEM 密钥）执行。
- 输出是“报文验证代码”（MAC）。
- MAC 提供给合约制造商（CM），合约制造商将在生产基地把 MAC 刷写到主控制器上。MAC 和对称密钥均由 CM 在此阶段加载。
- 请注意，这是供应链中容易出现漏洞的环节，因为无论是在制造过程中，还是在单片机中，密钥都不应该暴露。此时密钥会暴露给工厂内的操作员和将执行 MAC（如果 OEM 尚未完成）的设备。

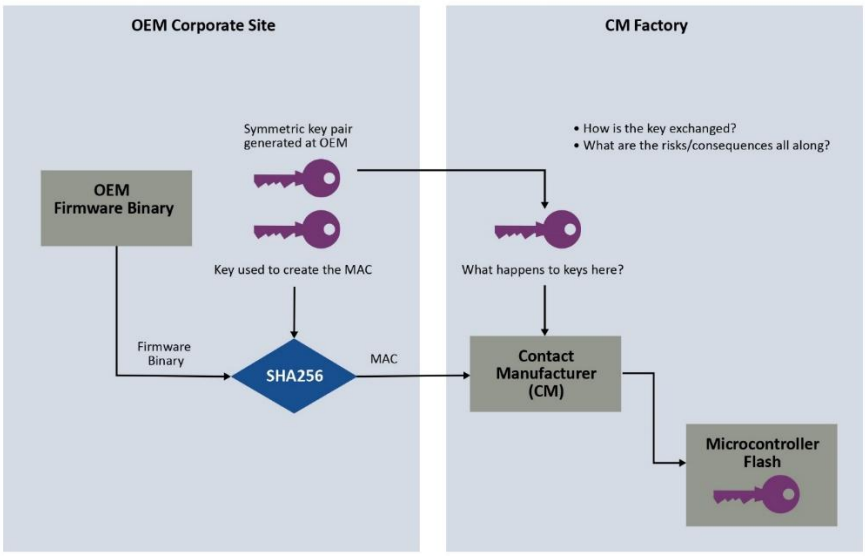


图1：在 OEM 生产基地进行对称签名

在制造过程中使用非对称密钥进行签名操作

这是一种更稳健且可扩展的方法，包括利用非对称密钥取代对称密钥。非对称密钥是由两个不同的密钥构成的一个密钥对，但两个密钥通过某种加密算法在数学上相关。我们将以 ECC-P256 为例，对于嵌入式系统来说，这是最常用，也是最有效的算法之一。私钥将用于对代码进行签名，公钥（根据私钥计算得出）将用于验证签名和/或摘要。

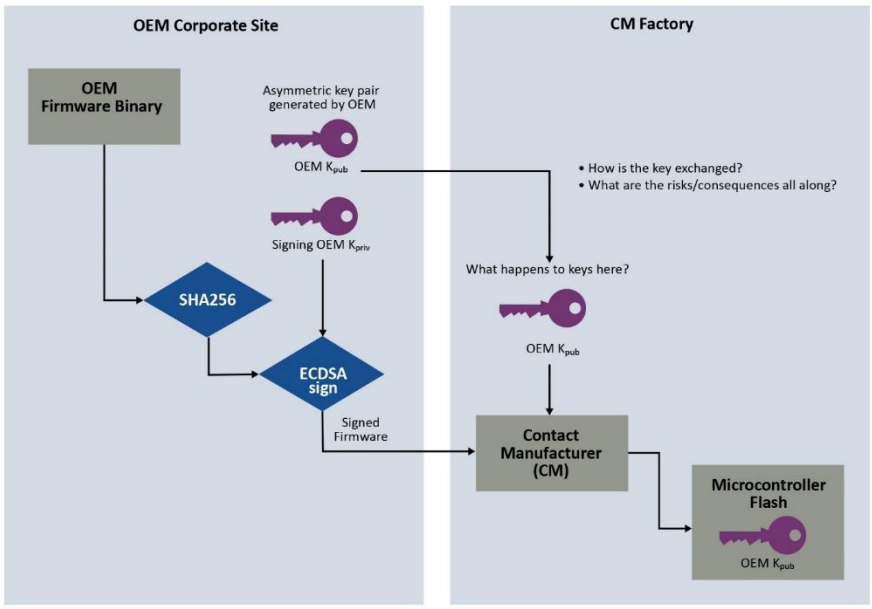


图 2：在 OEM 现场进行的非对称代码签名

想一想，在生产过程中，谁有密钥的访问权限？

无论选择的是对称架构还是非对称架构，都有几个重要的问题需要思考。由于需要将加密密钥加载到嵌入式系统以验证映像，请想一想：

- 能否相信掌握保护您已签名代码的密钥的合约制造商？切记，代码是公司的 IP。如果拥有密钥，他们就可以访问您的代码。
- CM 是否拥有签名密钥或用于验证的密钥的访问权限？如果您想更换一家或多家合约制造商，密钥怎么办？
- 您是否因合约制造商掌握了密钥而依赖他们？
- 如果雇用了多家合约制造商，如何管理各种密钥对集？
- 密钥在合约制造商那里有哪些保护措施？安全审计的结果会是什么？

涉及到制造过程中处理密钥的物流时，还有许多供应链问题需要考虑。签名密钥永远是最关键的密钥，要尽可能地与一切事物和人员隔离，最好是使用 HSM。但现在您面临的情况是代码验证成为非常重要的目标，因为它可以决定您对代码（记住这是公司的 IP）的保护程度。现在，密钥已交给合约制造商，此时漏洞暴露的可能性会增加。有几个问题可能需要您思考一下，包括：

- 您是否清楚密钥是否已妥善存储或在员工之间安全共享？
- 您的密钥是否因制造设备的网络保护不力，可以从工厂外部远程访问？
- 您的员工是否有可能简单地使用 U 盘将验证密钥带出工厂？如何对这种情况进行审计和确保员工可信？
- 如果掌管密钥的员工从合约制造商离职，密钥怎么办？

密钥位于嵌入式系统中的哪个位置？

如果将用于验证固件的公钥存储在传统单片机的闪存内，请慎重考虑。这将使加密成为代码二进制映像的一部分。传统工程师只是将密钥（公钥或对称密钥）加载到单片机或处理器的闪存中。我们来介绍一下您应该思考的问题和应该全盘考虑的答案。

密钥的价值有多大？攻击者能用它做什么？

时至今日，仍有大量设备将密钥存储在闪存中。攻击者的一些基本策略是尝试使用各种技术访问设备的存储器，如利用缓冲区溢出（示例：Heartbleed）、HEX 文件提取或其他方法来访问位于存储器中的密钥。这些都是非常真实的攻击，一些公司反映其系统中存在此类漏洞。在这种情况下，攻击者完全有可能开始发起可扩展攻击。如果每个密钥都像在二进制映像中一样容易访问，它们就会变得可更改、可复制，对大型机群进行远程访问的可能性也越来越高。如果我们对本文前面的内容还有印象，就会记得对于代码签名来说，使用对称密钥并不是最稳健的策略。现在，我们假设用于验证代码的密钥位于控制器的 OTP 内。OTP 方法是比较合理的初步措施，可以提高系统的稳健性，因为现在密钥位于不可变的存储区域。不可变并不意味着不可访问或不可读取，只是意味着不可更改。代码的价值决定对密钥的重视程度，具体视代码的价值或其中的 IP 含量而定。如果密钥可访问，它就可能被读取、复制和重复使用。现在任何恶意用户都可以合法地使用该密钥来验证他们自己的代码。

- 如果是对称密钥，情况将糟糕至极，特别是如果系统连接到云，更是雪上加霜。现在，当密钥被访问时，攻击者可以执行其恶意代码的 MAC，并且由于用于签名的对称密钥与用于验证代码的对称密钥相同，他们可以轻而易举地使用 checkMAC 验证其恶意代码。更糟糕的是，如果没有采取密钥多样化措施，那么十分有必要仔细考虑拥有整个设备群的密钥的人员名单。所有与婴儿监视器连接的工业机器都可能被伪造，所有 OTA 更新都可能遭到破坏，而且情况可能会恶化。

- 现在，如果被访问的是控制器闪存甚至是 OTP 中的公钥，那么恶意用户便可以随意复制和重复使用密钥以使用户授权合法化，并随后在目标单片机上安装和运行恶意代码。

与使用对称密钥架构相比，使用公钥架构要更加稳健，因为用于对代码进行签名的私钥与用于验证代码的公钥不同，但二者在数学上相关。也就是说，访问公钥允许攻击者查看并触发代码验证，以及在执行之前进行解密。此时，问题变成了："现在我看到了代码，要是我能更改代码并绕过公钥验证该怎么办？"要解决这一问题，除了适当的加密加速器（如 Microchip 安全元件的 ECC-P256）外，还需要单片机或处理器内的 BootROM 功能。BootROM 将确保控制器发出验证命令的存储区域也是不可变的，无法被绕过。然而，公钥被访问仍然是一个基本问题。

使用对称密钥与非对称密钥进行固件验证分别有哪些优缺点？

密钥	优点	缺点
对称	较少： - 加密简单 - 易于实现	很多： - 密钥被盗时非常容易被复制 - 可扩展攻击的风险较高 - 难以部署和管理惟一密钥群
非对称	很多： - 加密稳健 - 密钥架构可扩展 - 可使用 ECC 密钥优化存储器占用空间	较少： - 固件验证没有缺点 - ECDSA 的时间比 SHA256 长

在第一部分中，我们已经详细介绍了在应用中实现固件验证的重要性。应用程序代码将需要通过加密操作进行验证，以确保其真实性。验证可以在嵌入式系统运行过程中的不同时间点执行，如启动时、运行时和安全固件升级时。有许多技术可以使用非对称和对称的加密算法来实现固件验证。这些技术都有各自的优缺点。在本博客的第二部分，我们将介绍如何使用 Microchip 的安全元件和单片机实现固件验证。