Building trust: the journey towards security with open-source STM32 MPU solutions

Contents

5 Defining security
3 Role of standardization organizations
5 Protecting OEM assets has never been this simple thanks to STM32MP1 and STM32MP2 solutions
6 Secure boot: why is it important?
7 Public – private key authentication
8 STM32 MPU firmware signature process
8 Firmware authentication
9 The chain of trust and how it works
11 Security must be considered from the start
11 Secure secret provisioning
13 Hardware security certification options
14 How ST eases security certifications
14 High-security certifications for banking applications
16 The security and open-source software dilemma
18 The added value of open-source software
19 How you can benefit from the STM32 ecosystem and the open-source community
21 Managing the IP licensing scheme
21 ST partner program: benefit from a wide range of products and services
22 Conclusion
22 Useful resources

Standardized cryptographic algorithms offer numerous advantages over the concept of "security through obscurity." Standardized algorithms undergo rigorous public scrutiny by the cryptographic community, which helps to identify and rectify potential weaknesses, leading to stronger and more reliable security measures. This peer review process of open-source algorithms is critical because it leverages the collective expertise of cryptographers worldwide, ensuring that the algorithms are robust against various attack vectors.

In contrast, "security by obscurity" relies on keeping the details of a security mechanism secret, with the hope that if attackers are unaware of the system's inner workings, they will be less likely to find vulnerabilities. However, this approach is fundamentally flawed because it assumes that secrecy alone is sufficient to maintain security. Once the obscurity is compromised, the entire system's security can be jeopardized.

Furthermore, standardized algorithms are typically designed to be adaptable and scalable, allowing for seamless integration into different systems and applications. They also facilitate interoperability between various platforms and services, which is essential in our interconnected digital world. By using well-established and openly vetted cryptographic standards, organizations can ensure a high level of security that is based on proven principles and practices, rather than relying on the uncertain and risky strategy of keeping their security methods a secret.

This whitepaper addresses the various security challenges of embedded applications and how ST's open-source software implementation facilitates secure solutions without increasing costs. It introduces ST's OpenSTLinux distribution, which leverages the Trusted Firmware-A (TF-A), Trusted Firmware-M (TF-M), OP-TEE, and U-Boot open-source communities to establish a chain of trust, thereby helping companies to minimize development time and expenses.

DEFINING SECURITY

The increasing number of connected devices makes them an attractive target for attackers. OEMs must develop solutions that protect their products against a certain number of threats:

- Eavesdropping: Accessing information without authorization without any modification to the system
- Tampering: Modifying of the information or product without authorization from the manufacturer or end user

It is for these reasons that engineers must take into account certain security concepts in their designs:

- Confidentiality: To prevent data from being retrieved by unauthorized people
- Integrity: To ensure data has not been modified
- Authentication: Verify the data is coming unmodified from a known source
- Availability: To make data accessible whenever it is needed
- Non-repudiation: Assurance that a subject cannot deny a commitment or action after having performed it
- Anti-rollback: This feature prevents the loading of signed code that is older than the current version. Designed to prevent the exploitation of old vulnerabilities which have been fixed in newer versions, an anti-rollback mechanism prevents malicious people from reverting to a previous version.

ROLE OF STANDARDIZATION ORGANIZATIONS

Security in systems is one of the technologies which evolved the most in the past decade, moving from a “nice to have” to a “must have”.

Security started with “security by obscurity” where the security relied on the high-level skills of the architect. Now most security solutions are based on security standards. Still some systems continue to depend on “security by obscurity” due to certain constraints including footprint and performance.

Security standards rely on cryptography defined by standardization organizations, such as the United States’ National Institute of Standards and Technology (NIST), the French Cybersecurity Agency (ANSSI), and the German Federal Office for Information Security (BSI).

> “We must assume the enemy knows the system being used.
>
> Claude Shannon, the "father of information theory"

New cryptographic algorithms are based on Claude Shannon’s Communication Theory of Secrecy Systems that states “to make the problem mathematically tractable, we must assume that the enemy knows the system being used”.

Based on that principle, before standardizing a new cryptographic algorithm, NIST organizes an open public competition where candidate algorithms can be submitted. Proposals are evaluated by the community of experts and the NIST. Eventually NIST selects one or more proposals for standardization.

These steps are designed to ensure the selected cryptographic algorithm is robust from a mathematical standpoint that it is public.

The baseline of device security is the hardware as security levels increase with each generation of hardware, motivated by multiple parameters: market demand, regulation, threats, competition, certification, and geopolitics.

The first steps were to increase the security of the software as it was the easiest. Then the hardware was hardened to resist certain attacks.

> “It should not require secrecy, and it should not be a problem if it falls into enemy hands.
>
> Kerckhoffs's principle, Auguste Kerckhoffs (Dutch cryptographer)

Today, new hardware designers must consider security as a main requirement with several objectives depending on the targeted market vertical.

In order to demonstrate the security level of certain products or solutions, associations and standardization bodies have defined evaluation schemes based on the three parties involved: Developer, Laboratory, and Certification body. During these evaluation processes, products must comply with functional specifications and, depending on the certification level, penetration tests may be required. In some cases, the scope may be extended to include the development and manufacturing phases.

PROTECTING OEM ASSETS HAS NEVER BEEN THIS SIMPLE THANKS TO STM32 MPU SOLUTIONS

Regardless of the assets that need protection, security has always been difficult to implement. From end users to the engineering community and investors, the value chain has often shown some reluctance to accept and implement secure solutions.

As an analogy, implementing security in airports requires major investments in infrastructure, training personnel and customer acceptance. Likewise, deploying security within an IT network requires a dedicated development strategy with specific hardware and software features. Engineering capabilities must be developed to acquire strong expertise within the security domain. In order to stay one step ahead of threats, end users must accept the constraints of having slower networks and enhanced barriers to access different services.

Over the last 30 years, embedded security was snubbed in many domains as its added value was not completely understood. But the situation has changed and companies are now more eager to invest in security to obtain the right technical expertise because they realize that their return on investment is less jeopardized with a secure approach.

At another level, technical security in the embedded market has the same constraints as those of airport or IT network environments. The example that best illustrates the need for robust security in IoT connected devices (such as printers, IP cameras and others) is the Mirai malware. This malware and its many variants turned unprotected network devices into remotely controlled bots to conduct Distributed Denial of Service attacks on networks disrupting popular websites such as eBay, Netflix, GitHub, and PayPal to name a few.

Over the last decade, the continuous increase of various attacks drove IoT and Industry 4.0 companies to realize that security breaches might hurt their financial results and jeopardize their success. The time has come to drive investments on required infrastructure, IP development and deployment of industry initiatives. While the tide has turned, international standards and certification bodies dedicated to cybersecurity in the domains of IoT and Industry 4.0 among others have been created (PSA, SESIP, IEC 62443, per vertical organization, and many others) to make sure assets get properly protected against threats in the context of embedded devices, all along the life cycle of the devices. Indeed, in addition to the financial investments linked to development, OEMs must find innovative ways to accelerate the development phase in order to get their product to market more quickly and gain market share.

To help companies reduce development time and costs, ST proposes a solution based on open-source software, **ST's OpenSTLinux distribution**, providing a chain of trust built around the Firmware-A (TF-A), Firmware-M (TF-M), OP-TEE and U-Boot open-source communities.

While this proposal may seem surprising, there are many reasons why OEMs should leverage open-source software to protect their code while at the same time reducing the cost of the development phase. System designers will appreciate the level of security that can be easily added to their solution using STM32 microprocessors and their comprehensive development ecosystem leveraging years of experience of ST, partners and the community.

Among the different security features provided by the **STM32MP1** and **STM32MP2** platforms, this whitepaper focuses on implementing secure boot technology within embedded applications. While describing the different security challenges involved within the secure boot, the document explains how ST facilitates the implementation of secure embedded solutions without raising the bottom line thanks to an open-source software implementation.

SECURE BOOT: WHY IS IT IMPORTANT?

To ensure developers can have confidence in the security of their applications, the following sections address the key security requirements needed to protect the company's assets and end users' data at a minimum cost. These processes are also part of the **STM32Trust**, an innovative security framework combining ST's expertise, ecosystem and security services. To develop secure embedded solutions in a cost-effective manner, certain important concepts must be considered. When designing an IoT product with embedded security, both authenticity and anti-cloning concepts must be well understood in order to protect OEM and end user assets.

Secure boot provides the first stages of security in a device. Responsible for ensuring the global chain of trust of the system, it uses advanced cryptographic technologies to verify the integrity and the authenticity of the firmware before starting the main application. This is designed to thwart attacks that consist in modifying the firmware stored in external mass storage to inject malware, disable security features, dump software for product cloning, etc.

What is STM32Trust?

A robust multi-level strategy to enhance security in embedded systems, STM32Trust is a security framework that offers a complete toolset for code and execution protection and ensures IP protection, firmware authenticity and secure firmware updates, as well as secure data and the use of validated credentials.

STM32Trust helps protect your assets by identifying and analyzing threats and vulnerabilities to define protections and countermeasures and mitigating them with ready-to-use security functions and services.

To further ensure a high level of security, STM32Trust is based on two product certification schemes aligned with numerous national and application security standards:

- Security Evaluation Standard for IoT Platforms (SESIP) published by GlobalPlatform for IoT devices
- PSA Certified (Platform Security Architecture) by Arm® protecting IoT devices

PUBLIC – PRIVATE KEY AUTHENTICATION

The first step is to generate the private-public key pair for the secure boot. After generating the key pair, the private key needs to be stored in a highly secure environment in the OEM or subcontractor manufacturing site and never in the product nor at the third-party manufacturing site. If by some means, a malicious person gets access to the private key, they will be able to modify the code and re-sign it without being detected. The public key is used by the microprocessor at each boot to verify the signature of the OEM firmware located in the Flash memory and ensure its authenticity. The signature is generated only once by the OEM using the private key!

WHAT IS HASHING?

Based on a mathematical algorithm, hashing transforms the data in file into a fixed-size bit string. Changing even a single bit in a file will significantly change its hash value. This technique is often used to verify the integrity of a file after it has been transferred from one place to another; by comparing the hash values of both files, it is easy to determine if the files are different. The main advantages of hashing are:

- No way to determine the original message from the hash
- Resistance to collision of hash value: There is no practical algorithmic way other than brute force to find two modified messages with identical hash values.

WHAT IS PUBLIC-KEY CRYPTOGRAPHY?

This is a cryptographic scheme that uses pairs of keys (public and private) to prevent cloning and ensure the authenticity of their device. The private key is never exchanged with anyone, while the public key is designed to be shared. The mathematical foundation ensures it is not feasible to deduce the private key from the public key.

Messages are signed with the private key, and signatures can be verified by the receiving entity using the corresponding public key. This mechanism ensures a very high level of protection.

STM32 MPU FIRMWARE SIGNATURE PROCESS

The private key is used to generate the signature of the firmware while the public key will be hashed and stored in internal and one-time programmable (OTP) memory. These steps enable the authentication phase during the boot process. The use of **STM32CubeProgrammer** makes it easier to accomplish these steps. A two-step sequence is required to sign the firmware within a secure environment (a secured and protected PC in the OEM's or trusted subcontractor's site):

- A hash function hashes the product's firmware and generates a hash value with a fixed length.
- The Elliptic Curve Digital Signature Algorithm (ECDSA) is used to generate the firmware's signature by combining the hash value with the private key.

This initial code signature step paves the way to ensure the integrity and authentication of the boot images used during the initial chain of trust.

FIRMWARE AUTHENTICATION

The firmware authentication process starts by authenticating the public key stored in the external Flash memory in conjunction with the internal OTP content (Figure 1). The hardware (OTP) and boot-time computed hash values must equal each other. If the hash values do not match, this means that the public key has been tampered with. This may be the result of a malicious person trying to modify the product's firmware. If the hash values match, the public key is authenticated. The mathematical properties of public key cryptography (public and private keys) ensure the device boots only if the firmware is authenticated. Any attempt to break the integrity of the firmware layers breaks the chain of trust and prevents the device from booting. The digital signature process, including the above steps is summarized in Figure 1.

DIGITAL SIGNATURE DEFINITION

The digital signature process ensures the integrity and authenticity of the original firmware downloaded during the product manufacturing phase. A firmware authentication process is needed with asymmetric cryptography where a public key will be stored into the product while the private key is used to sign the firmware, stored in the product's non-volatile memory.

The private key must be well protected at the manufacturing site.

During the boot sequence, the firmware is hashed and compared to the signature to assess the integrity and authenticity of the OEM firmware.

The manufacturer's final objective may not be to protect the firmware's confidentiality, but rather to prevent malicious entities from either duplicating the product (anti-cloning mechanism) or adding malware to modify the product's purpose. Signing the code prior the manufacturing phase links this code to the final product.

By design, this prevents any potential malware added during the product's lifespan from persistently modifying the product and its use. The code-signing process ensures the integrity and authenticity of the firmware executed in the device at start-up, including even the customer applications.

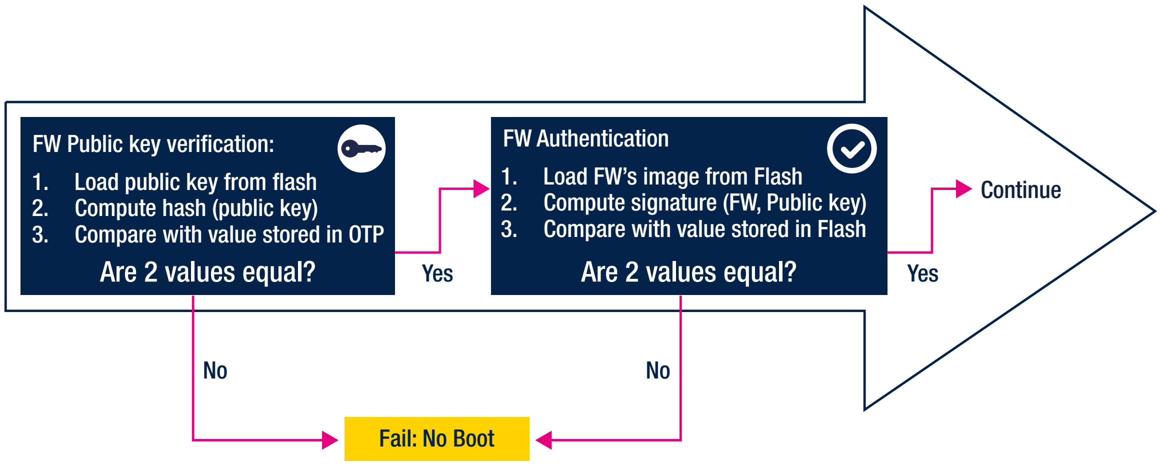

FW Public key verification:
1. Load public key from flash
2. Compute hash (public key)
3. Compare with value stored in OTP
Are 2 values equal?
Yes
FW Authentication
1. Load FW's image from Flash
2. Compute signature (FW, Public key)
3. Compare with value stored in Flash
Are 2 values equal?
Yes
Continue
No
No
Fail: No Boot

Figure 1: Firmware Authentication Mechanism on STM32MP1

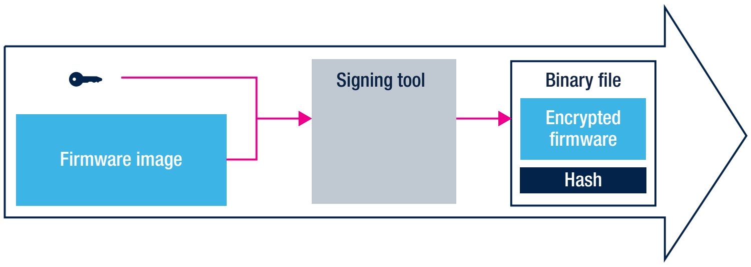
The STM32MP1 and STM32MP2 reference boot sequence starts with the authentication of the First Stage Boot Loader (TF-A) followed by the Secure OS (OP-TEE) securely initializing the STM32MP1 and STM32MP2, and finally the Second Stage Boot Loader (U-Boot) loads the Linux kernel.

The flexibility and implementation of the STM32MP1 and STM32MP2 along with the OTP allows the chain of trust to enable authentication from the ROM code up to and including OEM applications. STM32 MPUs are able to embed various authentication keys to support key revocation functions.

FIRMWARE ENCRYPTION

STM32MP13 and STM32MP25 microprocessors support an encrypted First Stage Boot Loader (FSBL) which ensures the confidentiality of the system's data from the ground up.

https://wiki.st.com/stm32mpu/wiki/STM32_MPU_ROM_code_secure_boot#Image_encryption



THE CHAIN OF TRUST AND HOW IT WORKS

During the boot-up sequence, the authentication process takes place between one stage and another to finally create the chain of trust sometimes known as the secure bootchain, for authentication and integrity purposes before reaching runtime.

The chain of trust, or secure bootchain, implements all the anti-rollback, authentication, anti-eavesdropping, and integrity mechanisms used to protect OEM and user data once the product is available in the field.

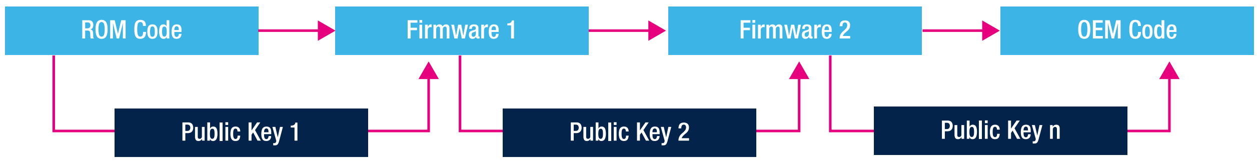


Figure 2: Use of public keys to ensure code authentication

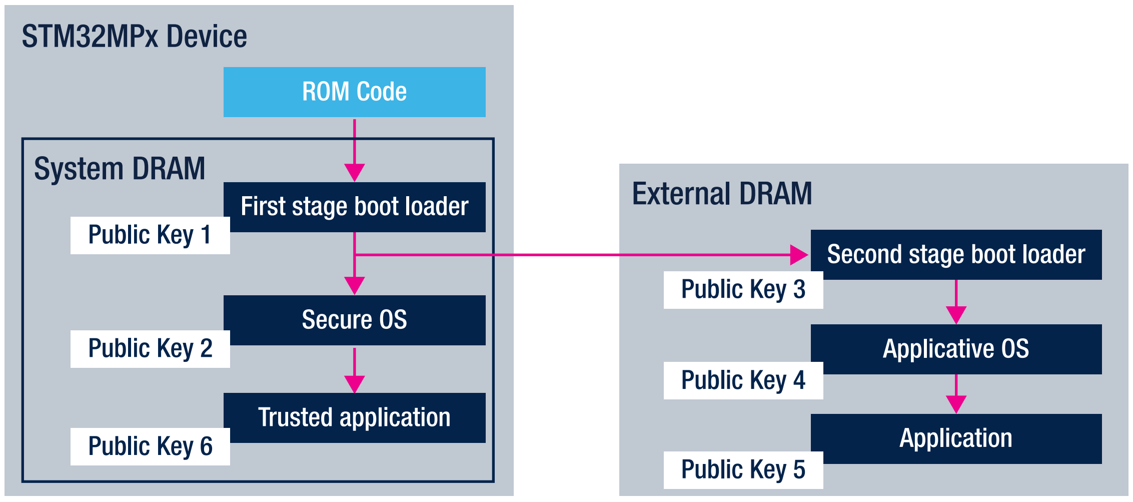


Figure 3: Trusted Foundation from ST - STM32MP1 reference bootchain sequence (chain of trust)

All boot stages up to the Second Stage Boot Loader are authenticated based on the OEM's public key. TF-A offers a PKI (public key infrastructure) starting from the OEM's public key to manage a chain of trust for the loaded binaries. Regarding the remaining firmware layers, the OEM has full flexibility to authenticate per its own requirements the remaining stages of the bootchain up to application levels as shown in Figure 3.

With this security infrastructure, a chain of trust (integrity and authentication) is created which allows the addition of extra services, once the kernel is accessible, by the Linux ecosystem to fully secure the entire OEM application.

By securely writing the hash value of the OEM's public key and secrets into the STM32MP1 and STM32MP2 OTP prior to the manufacturing phase (Secure Secret Provisioning mechanism), an OEM’s product (hardware) and its firmware becomes strongly interweaved with each other, preventing product cloning.

SECURITY MUST BE CONSIDERED FROM THE START

Considering security at the start of a project is important because it helps to identify potential security risks and vulnerabilities early on in the development process. This allows for the implementation of appropriate security measures and controls to mitigate those risks before they can be exploited by attackers.

Failing to consider security at the start of a project can result in security vulnerabilities being introduced into the system, which can be costly and time-consuming to fix later on. It can also lead to data breaches, system downtime, and damage to the organization's reputation.

By considering security at the start of a project, organizations can ensure that they are building secure systems that protect their data and systems from potential threats. This helps to minimize the risk of security incidents and ensures that the organization is compliant with relevant security regulations and standards.

Secured software
Secured hardware
Security by design
Certified security
Regulated security

- Secured software: Software provides a workaround for any possible hardware issues. Software, by nature, is easier to change than hardware. This is why when hardware does not implement security features or has a vulnerability, a workaround is developed with software.
- Secured hardware: Hardware includes security mechanisms. Software helps hardware ensure better performance for better resistance.
- Security by design: This approach involves integrating security considerations into the software and system development lifecycle from the beginning and throughout all stages of development. It is proactive, rather than reactive, to prevent security vulnerabilities and mitigate risks from the outset.
- Certified security: Standardization bodies publish security standards. Complying with security standards offers the advantages of ensuring interoperability and robustness. This assurance can be achieved through self-assessment or by engaging a third-party to test and certify that your solution adheres to specific standards, such as the NIST CAVP for cryptography or SESIP for security functions. Additionally, certain certifications involve a certifying authority that reviews the third-party results and issues a certificate of conformance.
- Regulated security: Regulations can, for given applications, define security requirements (i.e., the European Cyber Resilience Act). This is the strongest compliance requirement; if applications do not comply with these requirements, they can be fined.

SECURE SECRET PROVISIONING

Attacks can happen all along a product's lifecycle, not only when the product is in the field. They may happen for example during the production phase of the device by a malicious actor loading corrupted firmware (or stealing original firmware) while it is being programmed into chips. These are called "supply chain attacks", and since supply chains tend to be more and more complex, such attacks need to be considered by manufacturers and system providers.

The Secure Secret Provisioning (SSP) concept consists of protecting OEM information during the manufacturing phase, especially in the context of supply chains where it is difficult to assess the trustworthiness of the different players all along the production chain.

Prior to shipping the product to manufacturing facilities, confidential OEM data and hashes of the public keys are stored in a Hardware Security Module (HSM) card. Each genuine STM32 MPU is provisioned with a unique private / public key pair certified by ST during the chip manufacturing stage. The HSM uses this key pair to verify the authenticity of the MPU and protect OEM data all along the production process.

The HSM card is then sent to the potentially untrusted manufacturing site's owner and put in their production line. This guarantees the OEM's secret data and keys will be securely programmed only on genuine devices, and only accessible to OEM-signed firmware. Moreover, this procedure ensures the manufacture of a limited number of products (determined by the OEM) to better control inventory and prevent product cloning.

To help developers, our **STM32TrustedPackageCreator** utility packaged in the STM32CubeProgrammer tool can be used to create images and signatures required for loading the code in a secure manner, among other things.

When in-field firmware updates are required, STM32 MPUs have the correct secure boot implemented in the ROM code to prevent people from downgrading the firmware in order to exploit the firmware holes from the previous version and apply malicious techniques that can potentially change the behavior of the legitimate firmware. Indeed, the ROM code checks that the firmware version is higher or equal to the monotonic counter stored in OTP and automatically update it when a new valid version is detected.

HOW STM32CUBEPROGRAMMER CAN HELP

An all-in-one multi-OS software tool for programming STM32 devices, STM32CubeProgrammer (STM32CubeProg) offers a wide range of features for creating secure firmware and generating public – private keypairs.

Moreover, using the STM32 open development environment ensures easy portability to other STM32 devices.

HARDWARE SECURITY CERTIFICATION OPTIONS

Whether the software is open source or not, it is executed on hardware that needs to be secured at the correct level vis a vis its targeted usage.

Security certification can either be via self-assessment, an evaluation lab or a 3-party scheme:

- Self-assessment: you evaluate the security using your own scheme, without external support. This is quicker and less costly, but you need to hire people with the required skillset and there is no proof that your solution is secure as stated. Customers can be suspicious.
- Evaluation lab: you use an external lab to evaluate your solution. This is more costly and longer, but the lab can attest your solution is secure as stated. The evaluation report can be private or public; this is the developer's decision.
- 3-party scheme: The 1st party is the developer. The 2nd is the evaluation lab. The 3rd party is an independent entity called a certification body which defines the testing and method. The certification body qualifies (accredits) evaluation laboratories. This route is longer and more costly. This 3rd party can formally attest your solution is secure as stated. A public certificate is published on certification body's website.

The 3-party scheme is mainly used for SESIP, PSA, and PCI certifications. It is also worth noting that certain regulations mandate security certifications for various verticals such as automotive, industrial, and healthcare.

Never consider security as the last piece of the project, **but as the starting point**

HOW ST EASES SECURITY CERTIFICATIONS

ST's global security strategy is to ease efforts, complexity and costs for customers thanks to multiple certifications recognized by the industry. Additionally, it brings confidence and trust to the hardware thanks to a formal attestation of security from a recognized authority.

Each certification defines a scope of evaluation (list of security functions) that will be evaluated against defined tests.

Customers can apply these test results to their own security functions that rely on ST's hardware security functions.

For example, for STM32MP1 and STM32MP2 SESIP Level 3 certifications, the Hardware True Random Number Generator is tested against NIST SP800-90B. If a customer wishes to pass a FIPS 140-3 certification, they can re-use our SP800-90B certificate.

Another example is that SESIP Level 3 supports composition. Composition is similar to a Russian stacking doll: a security function based on a certified security function is certified. As a direct benefit, the customer mapping its security functions on underlying certified security functions would ease its device certification.

NEW IOT SECURITY STANDARDS: GET READY WITH A CERTIFIED MICROCONTROLLER AND ROOT-OF-TRUST

New IoT security standards: Get ready with a certified microcontroller and root-of-trust
Data Protection

Learn how IoT device makers can save time and cost to ensure compliance with the upcoming EU Cyber Resilience Act and revised Radio Equipment Directive.

Download our whitepaper

HIGH-SECURITY CERTIFICATIONS FOR BANKING APPLICATIONS

Due to the very nature of payment transactions, point of sale (POS) applications have their own 3-party security certification scheme.

The Security Standards Council manages the POS certification scheme and an information security standard named Payment Card Industry Data Security Standard (PCI DSS) is applied to payment systems.

The PCI-PIN Transaction Security Derived Test Requirements (PCI-PTS-DTR) document specifies all requirements that a POS solution must comply with.

Prior to deploying its solution, a POS vendor must pass the PCI-PTS certification process. As this involves all the hardware components included in the POS device and its related software, the scope is very large.

The STM32MP13x microprocessor series has successfully passed the PCI pre-certification: this is a PCI certification with a scope limited to the hardware. It cannot pretend to be a full PCI certification.

PCI certification is an attestation of a very high security level; as the POS handles transactions with banking cards, it should not be the weak point of the security chain.

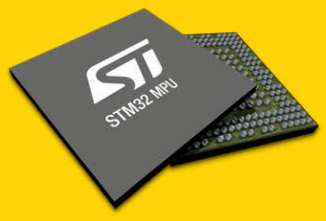
PCI is a 'monolithic' certification: even if a POS is composed of multiple components, the POS is to be certified as a whole component.

Thanks to STM32MP135 PCI pre-certification, POS vendors can more easily pass their PCI-PTS certification while reducing their efforts and time to market. With more confidence in the hardware security of the selected chip, companies reduce project risks while being able to focus more on other features.

ST's track record confirms positive customer experiences thanks to this PCI pre-certification strategy.

As STM32MP25 MPUs are in part based on the STM32MP135, they natively inherit its security features and are consequently ready for PCI pre-certification.

While non-POS vendors cannot use PCI pre-certification directly for their own certification, this PCI pre-certification is beneficial as it is proof that their application has a high security level.



Each STM32 MPU features a security option with an isolated secure boot with TrustZone® to protect applications from intrusions and more...

THE SECURITY AND OPEN-SOURCE SOFTWARE DILEMMA

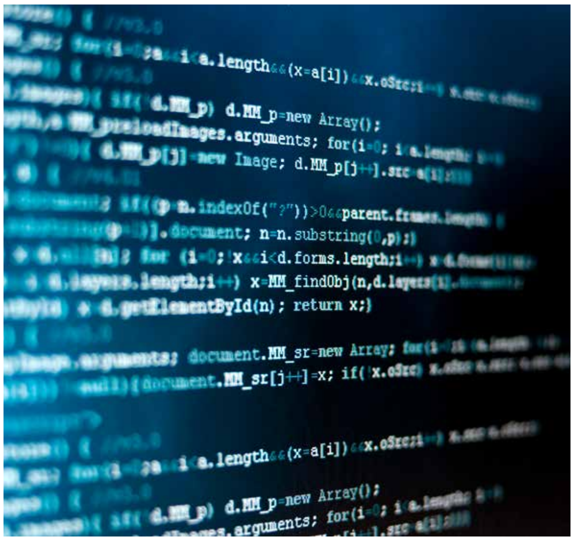
The perception of open-source software has evolved over time, and it can vary depending on the individual or organization. In the past, open-source software was often perceived as less secure than proprietary software, but this has changed as it now considers security as a key point.

Also, open source is no longer a small group of enthusiasts. The community has established rules and obligations to ensure the quality, security and interoperability of the software.

Security for open-source software is a difficult subject with benefits and drawbacks. When choosing between proprietary/commercial and open-source software, this duality is to be well balanced.

Benefits of open-source software include:

- Transparency: Greater transparency and visibility into the security features and potential vulnerabilities of open-source software are achieved by making the source code widely available. Communities are now using the DevSecOps framework and Product Security Incident Response Team (PSIRT) advisories.
- Support: The developer community assists in locating and fixing security vulnerabilities; important resources that an organization may not have at their disposition. Also, in general, these developers are security experts.
- Flexibility: Open source may be perceived as over-sized and over-engineered compared to specific security requirements. Open-source software can be modified, and certain functions can even be removed.



The supply chain is evolving: Yocto and Zephyr Embedded Projects are now automatically building SBOMs and signing binaries.

OPENSSF proposes tools for generating signatures and SBOMs as well as disclosing vulnerabilities.

But there are drawbacks to open-source software security as well, such as:

- Control: Compared to proprietary software, open-source software may give companies less control over software security because it is developed by a community of contributors.
- Complexity: Open-source software may have a large number of interdependent components, making it challenging to find and fix security flaws.
- Liability: As open-source software is by nature ‘open’, there may be some issues in terms of liability.
- Visibility: Unlike commercial software, open-source software may not have an extensive roadmap.

Open-source software can be a good option for organizations, but it is important to carefully evaluate the software and its security features before implementing it. Also, companies should ensure that they have the necessary expertise to properly customize, implement and maintain open-source software from a security perspective.

With the emergence of new regulations such as the European Radio Equipment Directive (RED) and Cyber Resilience Act (CRA), which require developers to disclose and fix their vulnerabilities, the open-source community has already put in place several initiatives:

Trustedfirmware.org has a Product Security Incident Response Team (PSIRT) and is publishing security advisories for each project (OP-TEE, TF-A, TF-M, and Mbed TLS)

OpenSSF OPEN SOURCE SECURITY FOUNDATION

The mission of the Open-Source Security Foundation is to sustainably secure the development, maintenance, and distribution of open-source software (OSS)

Explore STM32MPUs embedded software offer

Find your ST embedded software source code

THE ADDED VALUE OF OPEN-SOURCE SOFTWARE

To ease the development of software, mainlining in carefully selected good-quality projects with sufficient critical mass (such as Linux kernel, OP-TEE, etc.) provides the following benefits:

- Free-of-charge access to source code: Companies may inspect the available source code to select the desired features for their projects.
- Active maintenance: The open-source community can quickly fix known bugs and security vulnerabilities for further upgrade and enhancement.
- Ensuring software quality: Rules established in the selected open-source projects and code review by active community guarantee a high level of code quality.
- Software scalability towards different platforms: To reduce cost of software, a stable API between drivers and the application help developers migrate an application from one product to another.
- Certification: Depending on the topics and community members, open-source projects may obtain certain certifications and compliance with standards to help companies deploy their project.
- Support: By building on an open-source project, companies can reduce their development time and optimize their design efforts either with their internal software engineers or through authorized members of the **ST Partner Program** participating in the open-source community.

Open-source software is software that anyone can access, modify, and distribute, which can lead to greater collaboration and higher-quality code. At the same time, vulnerabilities like Log4shell have illustrated the downstream impact for flaws in widely used open-source code.

Cisa.gov, Sept 12, 2023

WHAT IS MAINLINING?

Leveraging the work of the many developers in the open-source community, modified code is uploaded (after validation) into the original, or mainline, source code. This means that the shared code benefits from the latest (and often innovative) modifications that add features, fix bugs or improve security.

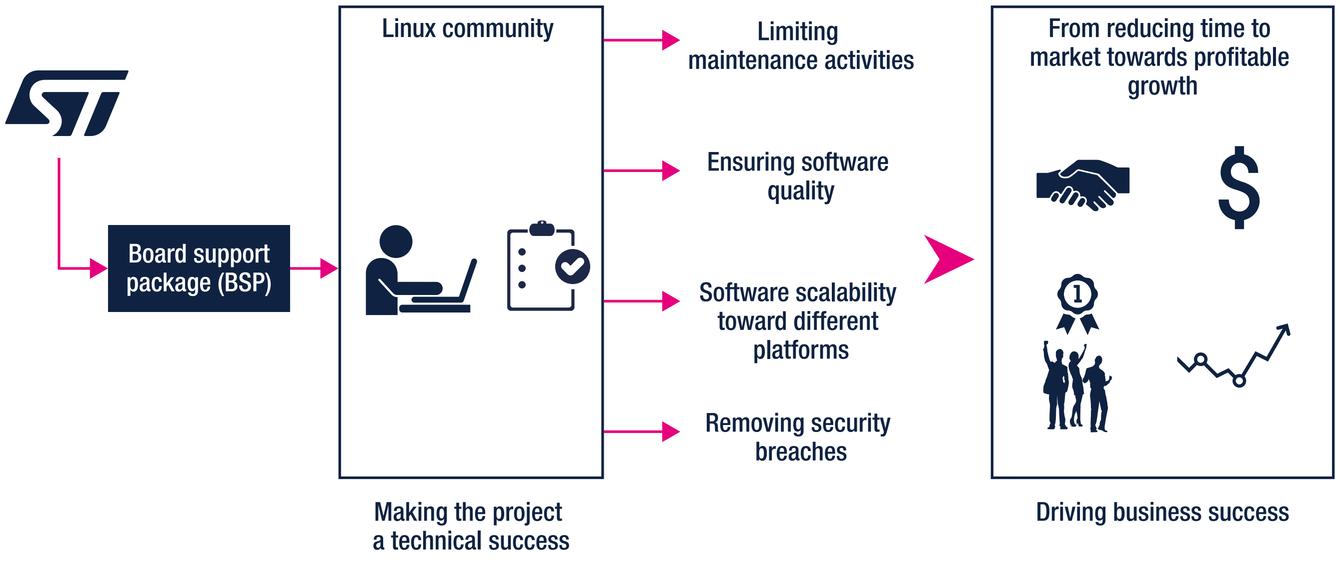

Board support package (BSP)
Linux community
Limiting maintenance activities
Ensuring software quality
Software scalability toward different platforms
Removing security breaches
From reducing time to market towards profitable growth
Making the project a technical success
Driving business success

Figure 4: Mainlining Advantages

HOW YOU CAN BENEFIT FROM THE STM32 ECOSYSTEM AND THE OPEN-SOURCE COMMUNITY

STMicroelectronics actively participates within the open-source community and the vast majority of the STM32 MPU drivers and board support packages are mainlined in all open-source projects that are part of the OpenSTLinux distribution.

By working with selected open-source communities (TF-A, OP-TEE, etc.), ST is continuously sharing proposals, requirements, and code enhancements to facilitate the application porting and long-term maintenance on top of the OpenSTLinux distribution.

These advantages of mainlining the open-source code help OEMs reduce engineering resources and development time, speed up time to market and, as a result, increase market share, while ensuring an excellent protection level with up-to-date patches and the most recent enhancements.

OEM developers can start their project using STM32 embedded software packages starting from the trusted bootchain based on Trusted Firmware-A (TF-A), Trusted Firmware-M (TF-M), OP-TEE and U-Boot. Each software component is authenticated before being securely installed in memory and executed by the processor's different execution contexts.

As a GlobalPlatform-compliant API, OP-TEE allows companies to develop their own trusted applications but also lets them use open-source ones or those provided by third party companies. Optionally and depending on the application, OP-TEE may load, authenticate, and isolate the real-time code running on the Cortex® M core.

The Universal Boot (U-Boot) is installed by TF-A and executed by the Cortex® A core into the non-secure context. The trusted foundation mechanism ensures the code’s integrity. The remaining open-source firmware stages can then be trusted.

In order to maintain an up-to-date Board Support Package (BSP) in terms of quality and stable environment (managing bug reports, fixing vulnerabilities, etc.), OpenSTLinux is based on long-term support versions and updated twice a year.

When correctly configured and initialized, different hardware mechanisms (memory protection for Cortex CPUs, TrustZone, Cortex-M core isolation, etc.) isolate the proprietary software from the rest of the software, thus protecting an OEM's intellectual property.

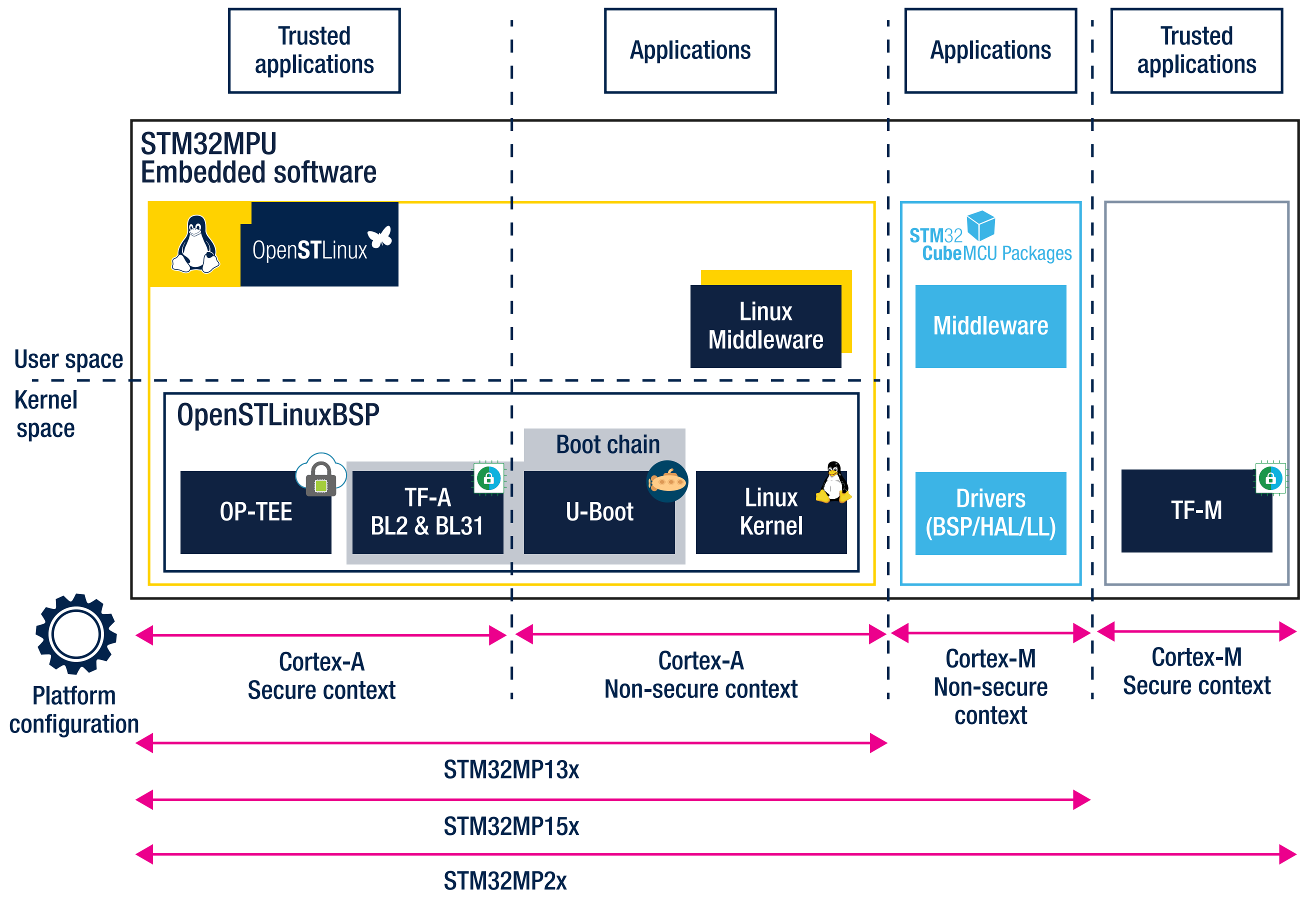
The hardware mechanisms listed above strengthen a defense-in-depth implementation. They provide separation mechanisms for the software stack with different levels of privileges. By applying the principle of least privilege, vulnerabilities can be confined to limited parts of the software stack, making it harder for malicious players to mount attacks.

By building a robust chain of trust, a secure bootchain ensures that attackers can neither corrupt nor tamper with an OEM's assets and end users' data.

Trusted Firmware-A (TF-A) is an open-source project by TrustedFirmware group, designed to run in the Secure Processing Environment (SPE) of Armv7-A and Armv8-A class processors.

Trusted Firmware-M (TF-M) is an open-source project by TrustedFirmware group, designed to run in the Secure Processing Environment (SPE) of ARMv8-M microcontrollers, following PSA Certified guidelines and offering services through secure partitions.

MCUboot is a secure bootloader for 32-bit microcontrollers, providing a common infrastructure for bootloader and system flash layout, enabling easy software upgrades, and is OS and hardware agnostic, relying on hardware porting layers from the OS it works with.



MANAGING THE IP LICENSING SCHEME

Another key consideration that must be taken into account when protecting OEM intellectual property (IP) resides in the licensing scheme management. There are many different licensing schemes that define how users can use, study, modify or improve open-source software, including the OpenSTLinux distribution, and as well as redistribute it in a modified or unmodified form. The most well-known schemes are the GNU General Public License (GPL), the MIT License (X11 License), Berkeley Source Distribution (BSD) licenses and some others.

By using TF-A, TF-M, and OP-TEE, OEMs can securely link and build their own proprietary code in a secure way since both environments are not copyleft and therefore are not contaminating. Furthermore, when releasing the OpenSTLinux distribution, ST provides legal licensing information for each code component helping customers to adapt their own licensing scheme strategy and implementation.

WHAT IS COPYLEFT?

A copyleft license, such as GPL, for example, lets software engineers use, study, change, and distribute modified or unmodified firmware and, more importantly, to make it fully available for further use by other software engineers.

Copyleft also includes the notion of contamination, meaning every piece of firmware becomes fully available for further use, thus preventing OEMs from keeping their own IPs confidential.

ST PARTNER PROGRAM: BENEFIT FROM A WIDE RANGE OF PRODUCTS AND SERVICES

ST Partner Program certifies and promotes the collaboration between ST and companies offering products and services that ease the adoption and use of ST devices.

When combined with the long-term maintenance strategy with ST Authorized Partners, ST ensures a high level of trust for projects based on the STM32MP1 and STM32MP2 platforms.

ST PARTNER PROGRAM

The ST Partner Program helps companies easily identify trusted partners able to supply expertise for their critical design projects; reducing their development efforts and accelerating time to market.

CONCLUSION

In the realm of embedded systems, the conversation has shifted from a mere focus on security to a broader emphasis on trustworthiness, which is increasingly grounded in transparency. The inherent transparency of open-source firmware is a significant factor in this evolution, as it undergoes continual refinement and assessment.

Regulatory measures are also contributing to this transformation by advocating for certification processes, transparency, and the ability to update systems. However, this evolution is not without its challenges. As the landscape evolves, so too do the strategies of attackers, who are now employing more sophisticated techniques such as side-channel and fault attacks, exploring new vectors of attack that range from remote to local, and leveraging emerging technologies like artificial intelligence to streamline their learning process. Additionally, the discussion around open-source hardware is gaining traction, with many initiatives already exploring its potential and implications.

This whitepaper demonstrates the trusted foundation behind our STM32MP1 and STM32MP2 platforms designed to ensure OEMs' products cannot be cloned nor compromised thanks to a robust chain of trust. In addition, using open-source software and benefiting from the shared contributions of the open-source community means OEMs can reduce development costs, lower the time to market, and gain market share.

RESOURCES

Product and development ecosystem pages

STM32MP1 microprocessor series from single Arm® Cortex®-A7 up to dual Arm® Cortex®-A7 and Cortex®-M4 cores [Product portfolio]

STM32MP2 microprocessor series with up to dual Arm Cortex®-A35 and Cortex®-M33 cores [Product portfolio]

STM32Trust: Raising the bar on security in embedded designs [Product ecosystem]

STM32 MPU software development tools [Product portfolio]

STM32 MPU embedded software [Product portfolio]

STM32CubeProgrammer all-in-one multi-OS software tool (containing the STM32TrustedPackageCreator utility) [Product page]

STM32 MPU OpenSTLinux Distribution [Product page]

STM32 MPU OpenSTLinux expansion packages [Product page]

Learning resources

STM32 MPU Platform Security overview [Wiki page]

STM32 MPU Platform boot overview [Wiki page]

STM32 MPU OP-TEE overview [Wiki page]

STM32MP1 Workshop or How to Appreciate the Complexities of MPU Design [Blog post and video]

STM32MP1 workshop [MOOC]

Enable secure, advanced edge AI in Industry 4.0 with the STM32MP2 MPU series [Webinar]

Technical documentation

AN5156: Introduction to STM32 microcontroller security [Application note]

AN5510: Overview of the secure secret provisioning (SSP) on STM32 platform [Application note]

New IoT security standards: Get ready with a certified microcontroller and root-of-trust [Whitepaper]

External websites

Open Portable Trusted Execution Environment (OP-TEE) [Website]

Open-Source Security Foundation (OpenSSF) [Website]

SESIP [Website]

ACRONYMS

API:	Application Programming Interface
CRA:	(European) Cyber Resilience Act
DRAM:	Dynamic Random Access Memory
ECC:	Elliptic Curve Cryptography
ECDSA:	Elliptic Curve Digital Signature Algorithm
eMMC:	embedded MultiMedia Card
FSBL:	First Stage Boot Loader
HSM:	Hardware Security Module
IP:	Intellectual Property
OEM:	Original equipment manufacturer
OP-TEE:	Open Portable – Trusted Execution Environment
OTP:	One Time Programmable
PSIRT:	Product Security Incident Response Team
RED:	(European) Radio Equipment Directive
ROM:	Read-Only Memory
SBOM:	Software Bill Of Material
SSBL:	Second Stage Boot Loader (U-Boot)
TF-A:	Trusted Firmware-A
TF-M:	Trusted Firmware-M

At STMicroelectronics we create technology that starts with You

Order code: **WP2408STM32MP1SEC**

For more information on ST products and solutions, visit www.st.com