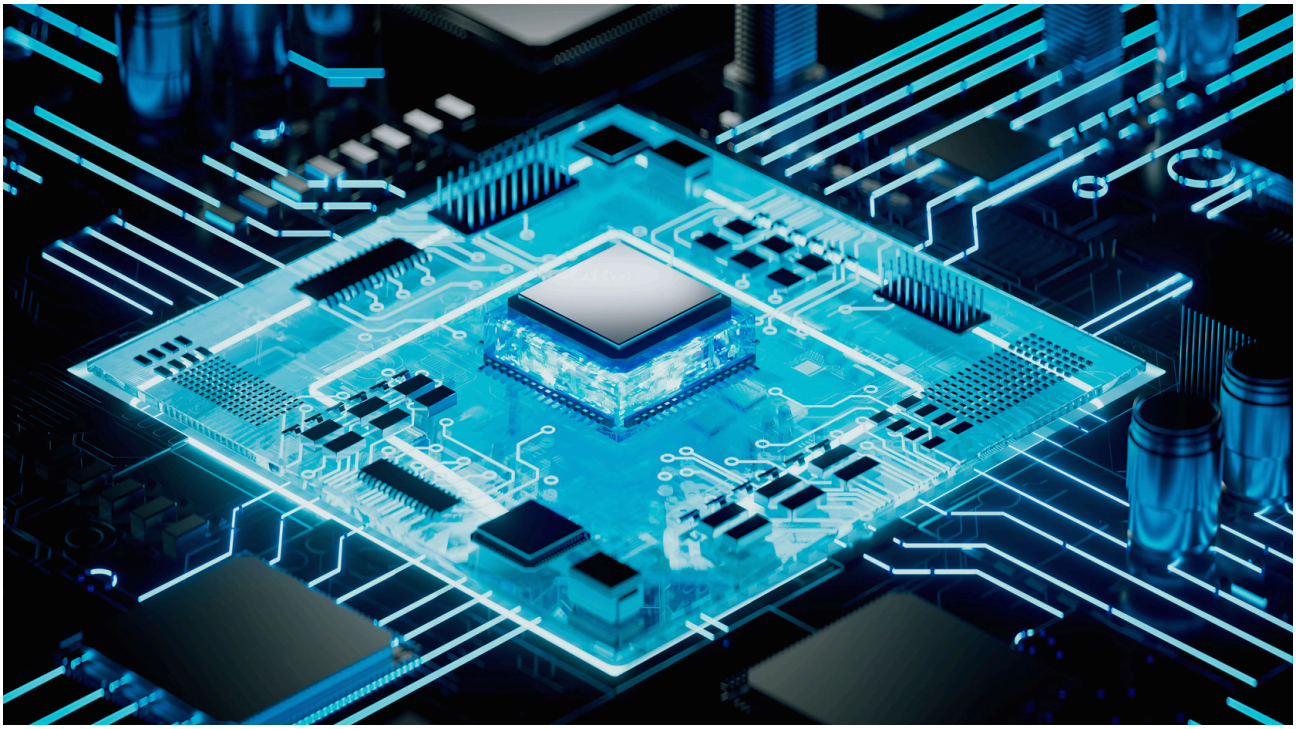
EPTECH

Essential Guide Securing IoT & Embedded Devices

Protect your devices with **Embedded Secure Boot** Solutions

Embedded Secure Boot Introduction

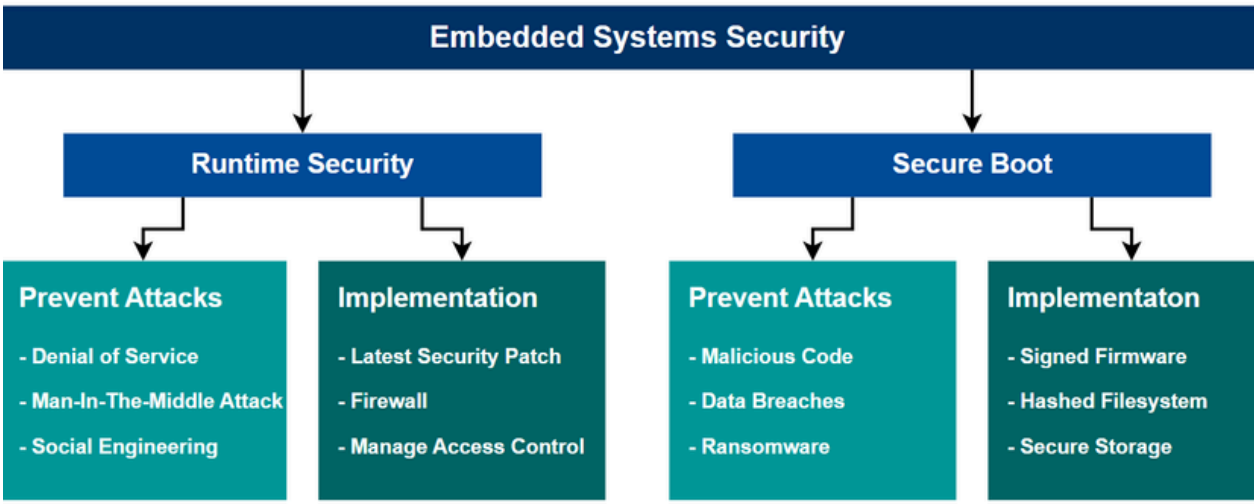


With the increase in Network-Connected IoT devices, cyberattacks are also increasing day by day. A survey says that 94% of IoT device vendors have reported at least one harmful cyberattack in the last 2 years. According to IoT Security Landscape Report in 2024, home network devices receive an average of 10 network attacks daily. To prevent losses in cyberattacks, we must implement foolproof cybersecurity solutions in our embedded and IoT devices.

Cyber Security in IoT & Embedded Devices

To achieve comprehensive cybersecurity within a system, two types of security measures are required.

- **Runtime Security**
- **Secure Boot**



Runtime Security

Runtime security primarily refers to the security of systems while they are operational, particularly in relation to network threats. It helps protect against attacks originating from networks while the system is running. By implementing security measures, runtime security aims to mitigate these attacks as effectively as possible. Network security plays a crucial role in preventing such threats.

Most Common Cyber Attacks

Denial of Service (DoS)

An attack that overwhelms a system or network, making it unavailable to users by consuming resources.

Man-in-the-Middle (MITM)

An attack where an attacker intercepts and potentially alters communication between two parties without their knowledge.

Social Engineering

A technique used to manipulate individuals into divulging confidential information or performing actions beneficial to the attacker.

Measures to reduce these Attacks

Closing Unused Network Ports

Reduces the attack surface by disabling network ports that are not needed, preventing unauthorized access.

Implementing Firewalls

Controls incoming and outgoing network traffic based on security rules, acting as a barrier to block unauthorized access.

Applying Latest Kernel Security Patches

Fixes vulnerabilities in the operating system's kernel to protect against newly discovered threats and exploits.

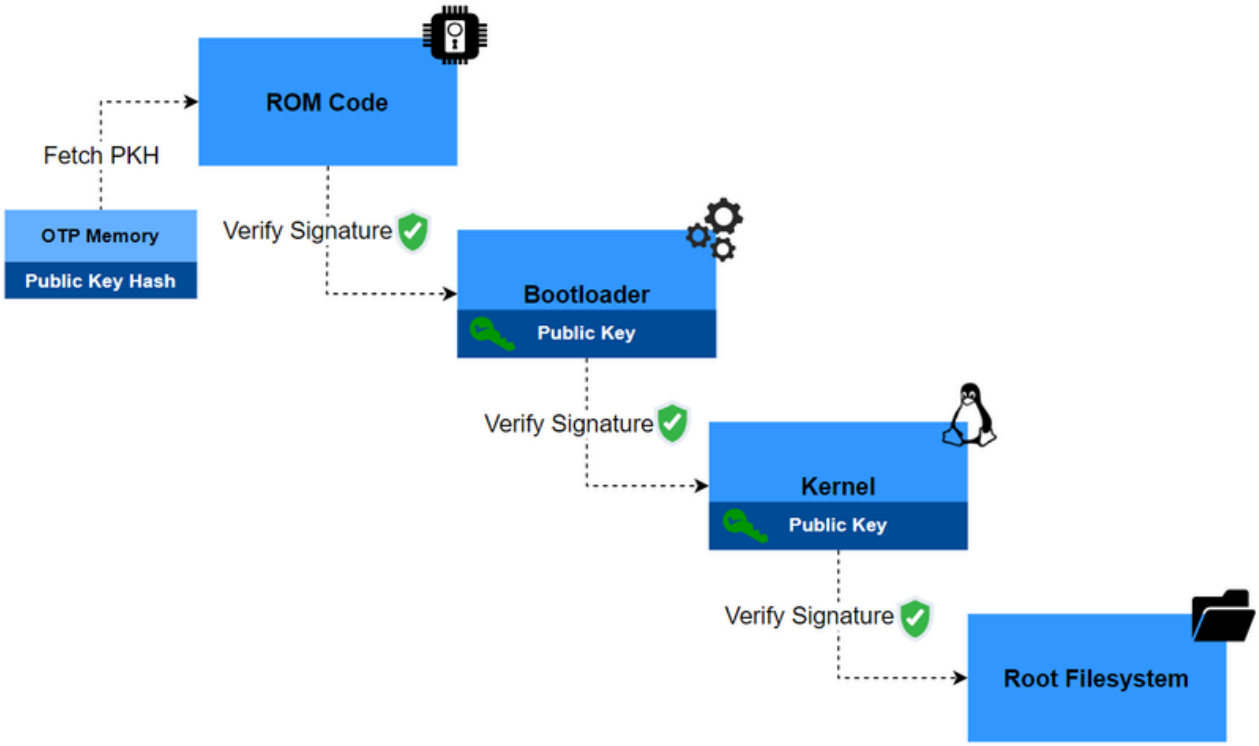
Runtime Security without Secure Boot?

Runtime security is crucial for defending against various network attacks, but it is insufficient without proper Secure Boot. Secure Boot protects against threats such as malware injection, bootkits, and rootkits—attacks that runtime security alone cannot address. If applications in an embedded system are compromised, network security becomes ineffective. This is where **Secure Boot becomes vital**, as it ensures the integrity of the system at its foundational level.

As a result, Secure Boot serves as the **base line** of all security measures within the system.

Secure Boot in IoT & Embedded Devices

Secure Boot is a technique that ensures only trusted and authorized firmware is executed on network-connected IoT devices. In Secure Boot, each piece of firmware must have a "**secure passport**" to run on the hardware, which is known as a digital signature.



Secure Boot begins its process immediately after the "Power On" button is pressed. It establishes a **chain of trust**, where each boot component validates the next one using a digital signature before execution during the boot-up process. This ensures strong protection against malicious code, ransomware, rootkits, and bootkits.

Embedded Systems Boot Process

ROM Code 01

It is the first code that runs on embedded hardware, usually provided by Hardware Vendors. Its purpose is to initialize basic peripherals at startup and load bootloaders.

Bootloader 02

The role of the bootloader is to load the kernel from different mediums like eMMC, SD Cards, Network and Nand/NOR Flashes. The components are SPL, ATF, Main Bootloader and OP-TEE (optional).

Kernel 03

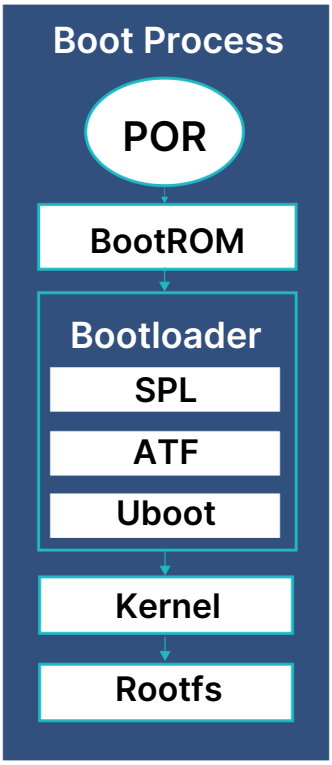
The kernel is the core, responsible for task scheduling and hardware management in embedded systems. It stays in the system RAM until the device remains Powered ON.

RootFS 04

All our files, folders and application binaries reside in the Root File System (RootFS).

Embedded Systems Boot Process with Secure Boot

In Secure Boot, each boot component validates and authenticates the next component before execution. The Original Device Manufacturer (ODM) provides signed binaries for the IoT device, and these digital signatures are verified at runtime before the components are executed.



1. When the device is powered on, the **ROM** code begins execution. This code resides in Read-Only Memory and cannot be modified, so it does not require authentication.

2. Next is the **Bootloader Binary**, whose signature is authenticated by the ROM code. The specific authentication techniques for the bootloader vary by vendor, which will be discussed later.

3. The bootloader is then responsible for loading the **Kernel**. Before execution, it validates the kernel binary's signature using public keys.

4. Once the kernel is loaded, it mounts the **Root Filesystem (RootFS)**, where all files and applications are stored. The kernel ensures the integrity of the rootfs by verifying its hash value.

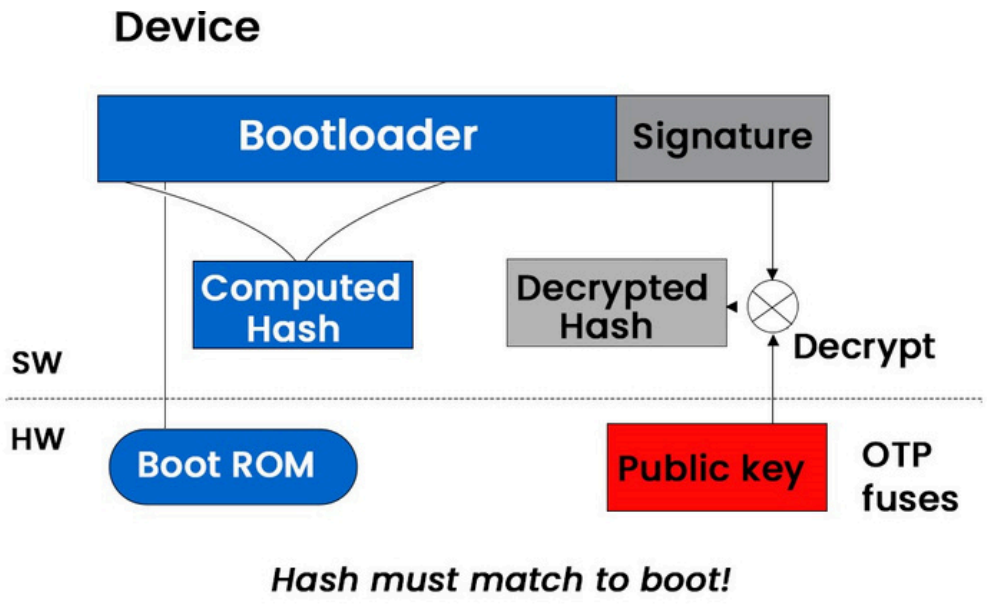
This process establishes a **Chain of Trust**, ensuring that no unauthenticated software can run on the hardware. This greatly reduces the risk of cyberattacks and protects devices from malware injection, bootkits, and rootkits.

Bootloader Verification

Bootloader verification methods vary across vendors. The ROM code is responsible for bootloader verification, and this code is typically provided by hardware vendors like STM, NXP, and Xilinx.

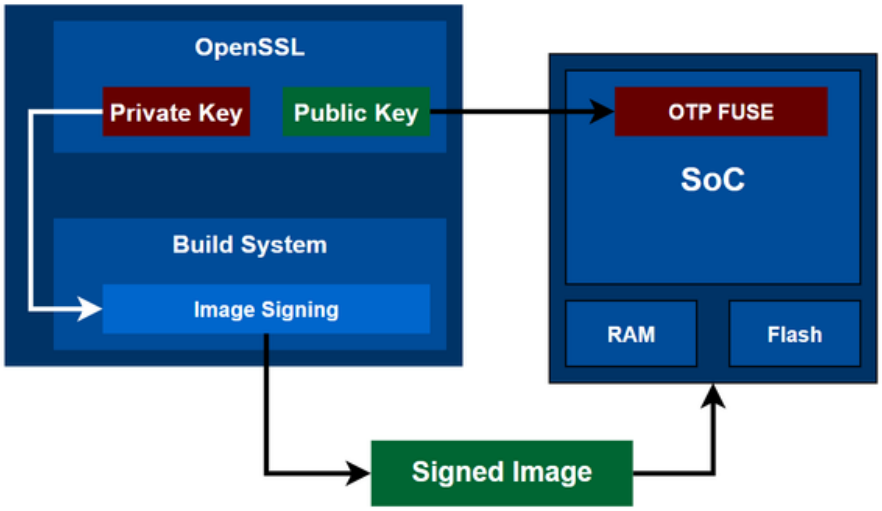
NXP offers **High Assurance Boot (HABv4)** and **Advanced High Assurance Boot (AHAB)** for bootloader authentication. Both HAB and AHAB use PKI (Public Key Infrastructure) hashes to authenticate the bootloader. NXP provides the Code Signing Tool (CST) to sign the boot components. The PKI hashes of these keys are stored in the device's One-Time Programmable (OTP) memory. The ROM code then uses these PKI hashes to validate the bootloader's signature.

Similarly, STM utilizes PKI hashes stored in OTP fuses within the hardware to authenticate the bootloader image. STM provides the **STM32 Signing Tool** for key generation, bootloader signing, and managing PKI hashes on STM boards.



Kernel Image Verification

The kernel image is authenticated using a public-private key pair technique. A pair of keys is generated on the host machine using OpenSSL, with the private key used to sign the kernel image. The corresponding public key is embedded in the bootloader image. During boot, the bootloader verifies the kernel image's signature against the public key stored in the bootloader. This ensures that only **authenticated kernel images** are loaded onto the device.

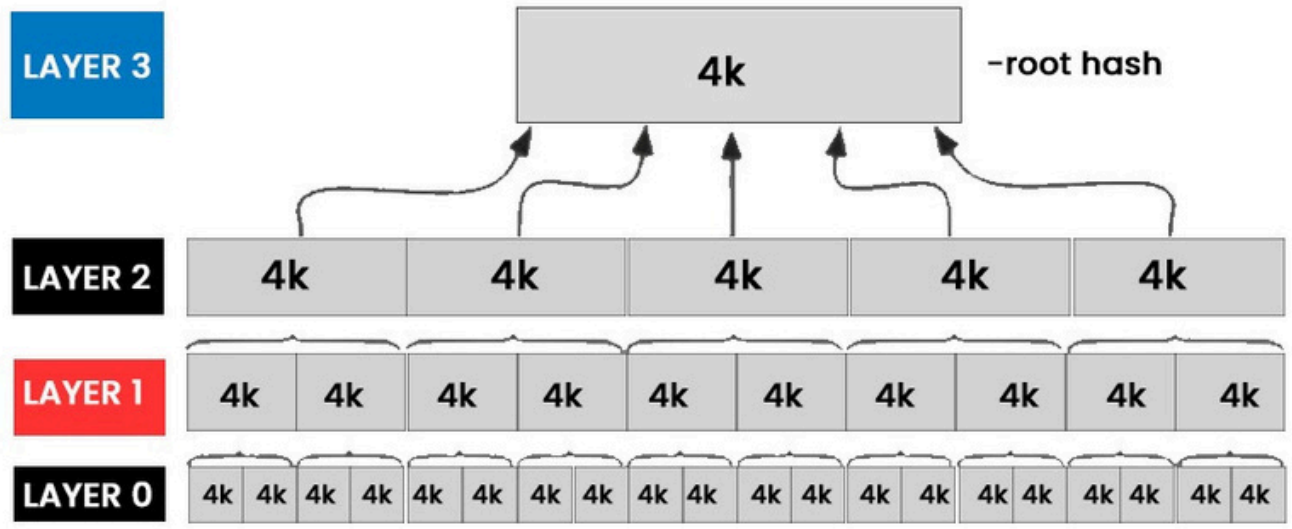


RootFS and Application Integrity

Bootloader verification and kernel image authentication help protect against bootkits, where attackers attempt to alter the boot flow and compromise the entire boot process. To further protect against rootkit attacks and ensure the integrity of the root filesystem (rootfs) and applications, **dm-verity** and **dm-crypt** techniques are used.

In these techniques, the root hash of all applications and files is calculated at build time and stored in the kernel. If an attacker tries to modify files or applications, the **rootfs hash** will change. Upon the next reboot, if the rootfs hash doesn't match the build-time hash stored in the kernel, the system will reject execution, preventing unauthorized changes.

The dm-verity and dm-crypt techniques are highly effective hashing methods. They **calculate hashes recursively** by dividing the root filesystem into multiple blocks, enabling them to detect even the smallest changes, such as a single word modification, in the filesystem. This gives them an advantage over other hashing techniques like md5sum and sha256sum.



Data Encryption and Secure Storage

Another form of cyber threat involves hackers infiltrating IoT systems to steal users' private data without modifying applications or boot components. These threats can be mitigated by implementing secure **encrypted storage** in the hardware to protect critical user data.

As a result, Secure Boot is often complemented by a **Trusted Platform Module (TPM)** or **Trusted Execution Environment (TEE)**, which runs alongside the main operating system. This setup allows safety-critical applications to run in a trusted environment, offering secure encrypted storage and supporting cryptographic operations.

Secure Storage with External Trusted Platform Module (TPM)

A **Trusted Platform Module (TPM)** provides hardware-based encryption and secure storage for sensitive data, ensuring protection against unauthorized access, even if the operating system is compromised.

Secure Storage with OP-TEE

Open-Portable Trusted Execution Environment (OP-TEE) offers software-based encrypted storage within a trusted environment, isolating critical data from the main operating system and safeguarding against data breaches and leaks.

OP-TEE as Secure Storage Provider

Key Management for Secure Boot

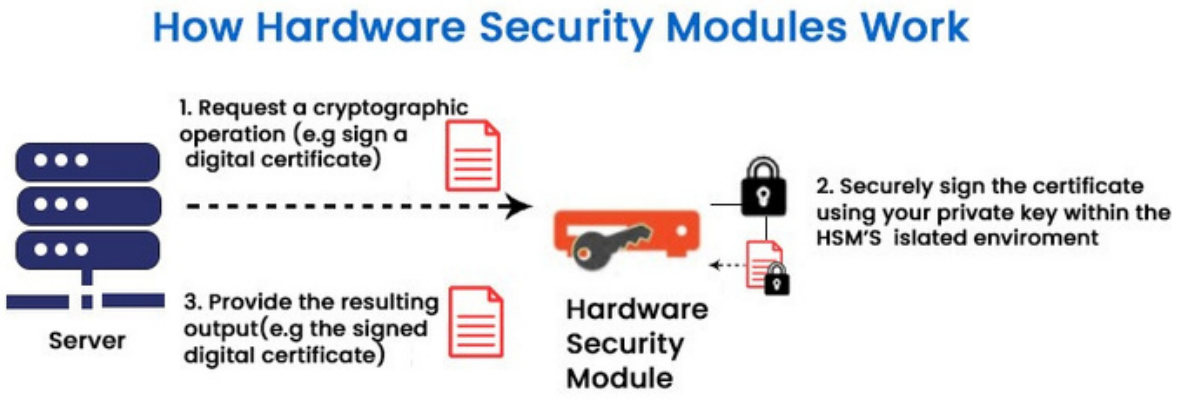
Hackers sometimes target production servers to steal the keys used for signing Secure Boot components. Therefore, it is not advisable to store signing keys openly on production servers. There are two techniques for managing **Secure Boot Keys** in production environments:

Hardware Security Modules (HSMs)

HSMs come in three forms: hardware, software, and cloud-based. Key generation and boot component signing are performed within the HSM, which does not transmit the keys to production servers. Instead, it uses cryptographic processors to sign the required components, ensuring that the keys never leave the HSM.

Vault Secret Managers

Vault Secret Managers are cloud-based encrypted storage solutions designed to secure private keys in encrypted form. In these secret managers, keys are stored securely and are only retrieved by production servers at the time of signing.



Secure Boot for GPSR Compliance

Our Recommendation: Secure Boot is essential for cybersecurity in IoT and embedded systems. The EU General Product Safety Regulation (GPSR) section 2023/988 requires Original Device Manufacturers (ODMs) to implement Secure Boot in their devices **by December 13, 2024**. Non-compliance may result in significant fines and product bans in the European market.

This regulation highlights the importance of Secure Boot in networked IoT and embedded devices. It is vital for ODMs to adopt Secure Boot to maintain their market presence in Europe and build client trust by reducing the attack surface on their devices.

Need a Secure Boot Partner?

If you are seeking a reliable partner to implement Secure Boot in your IoT or embedded devices, we are here to help. We have extensive **expertise in Secure Boot implementations** across various processor architectures and have delivered tailored security solutions to top ODMs throughout Europe.

Book a **Free Consultation Call** today to explore the best Secure Boot implementation solutions for your IoT and embedded devices!

BOOK A FREE CONSULTATION