CYBERSECURITY · DEVOPS · COMPLIANCE

SECURELY SIGNED

Why Code Signing Is Now Tied to Hardware – and How It Can Still Scale

Felix Heitländer
Senior Customer Support Engineer at SEH Computertechnik GmbH

Certificates stored on USB tokens help ensure that software has not been tampered with.

The use of so-called dongles is secure, cost-effective, and can even be implemented on virtual machines in the cloud.

Software has long since become more than just a product. It controls machines, processes sensitive data, automates business operations, and can ultimately determine whether a company remains operational. As a result, the damage caused by tampered or compromised software can be significant. The key question is therefore: Where does a piece of software come from—and can its integrity be trusted?

The answer is code signing. Digital signatures make it possible to verify both the origin and the integrity of software. Modern operating systems and corporate security policies consistently rely on these mechanisms. Unsigned software? It is either blocked outright or, at the very least, treated with considerable suspicion. Code signing has therefore become a fundamental requirement for professional software deployment.

This development extends far beyond traditional desktop applications. Firmware, scripts, container images, and automatically distributed software updates are all subject to strict signature verification. The more deeply software is integrated into critical infrastructure and industrial processes, the greater the need to ensure its integrity. As a result, code signing has become the essential link between software development, operations, and compliance.

From Security Feature to Mandatory Practice

Modern operating systems block unsigned applications by default or display prominent security warnings. In many enterprise environments, running unsigned software is prohibited altogether.

The reason is straightforward: without code signing, attackers could distribute tampered applications that appear indistinguishable from legitimate software. Digital signatures establish a reliable technical foundation of trust, regardless of whether software is installed locally, downloaded from the internet, or distributed automatically.

As a result, code signing is no longer a niche concern for developers—it has become a core component of every organization's cybersecurity strategy.

Stricter Requirements for Private Keys

As the importance of code signing has grown, so too have the security requirements for the certificates used in the process. Particular attention is now paid to the protection of private keys. If these keys fall into the wrong hands, even malicious software can be signed to appear legitimate.

Regulatory requirements introduced in 2023 therefore mandate that private keys may no longer be stored unprotected on file systems. Instead, they must be kept in dedicated secure hardware that meets defined security standards, such as Common Criteria EAL 2+ or FIPS 140-2. This ensures that private keys cannot be copied, exported, or misused without detection.

In practice, USB-based security tokens have become the standard solution. From a cryptographic perspective, they are highly effective—but operationally they introduce new challenges.

For organizations, these regulatory changes represent a significant shift. Responsibilities, access controls, and technical safeguards must now be clearly documented and implemented in a traceable manner. As a result, code signing has become not only a technical security measure but also an integral part of governance and compliance.

When Hardware Meets Modern Development Models

Modern software development is highly automated. Build processes run in virtual machines, CI/CD pipelines sign software automatically, development environments are provisioned on demand, and cloud infrastructures scale computing resources dynamically.

A physical USB dongle does not naturally fit into this model. It is tied to a specific location, offers limited parallel access, and cannot simply be attached to a cloud instance. This creates a conflict between stringent security requirements and the realities of modern software development.

Without the right infrastructure, organizations encounter familiar problems: build processes must pause while waiting for access to a token, additional hardware tokens are purchased to reduce bottlenecks, or security controls are "temporarily" bypassed—a temporary solution that often becomes permanent. Ultimately, either security or efficiency suffers.

This trade-off becomes particularly apparent in dynamic environments with multiple parallel build jobs, where any resource tied to a single physical device quickly becomes a bottleneck.

The Core Challenge: Lack of Decoupling

The central issue is not code signing itself, but the tight coupling between the security token and a physical machine. As long as a certificate is available only on the computer to which the USB token is physically connected, virtualization remains an exception rather than the norm.

In practice, this results in greater administrative overhead, the need for additional hardware tokens, limited scalability, and workarounds that gradually undermine established security practices.

The logical solution is to decouple access from physical location: keep the certificate securely protected in hardware while making access to it flexible.

USB Over the Network: A Proven Concept Reimagined

One practical approach is to host USB security tokens centrally and make them available over the network. Specialized software emulates a locally connected USB controller on the client system while the physical token remains securely installed elsewhere.

To the signing server, the token appears to be connected locally. In reality, USB traffic is securely transmitted and controlled over the network. This enables virtual machines, containers, and cloud instances to use hardware-protected certificates without requiring direct physical access to the USB device.

Example:
A software vendor operates its CI/CD pipeline in a cloud environment with automatically scaling build agents. Rather than assigning each build agent its own hardware token, a single centrally managed token is hosted in the company's data center. Build instances connect to the token over the network when needed, sign the software artifact, and immediately release the token for the next job. The result is a fully automated signing process while private keys remain protected by dedicated hardware.

Automation as the Key to Success

The real value comes from automating access to the security token. Command-line interfaces allow a token to be reserved, used, and released programmatically, making code signing an integral part of the build pipeline.

A single hardware token can therefore be shared sequentially among many systems, remaining allocated only for the duration of the signing operation. This significantly reduces both hardware requirements and associated costs.

Automation also improves process reliability. Repeatable workflows reduce the risk of configuration errors while creating a clear separation between development environments and security-critical cryptographic keys.

Example:
An industrial manufacturer develops firmware for control systems across several locations. Previously, each development team maintained its own USB token, creating administrative overhead and making access difficult to document consistently. By replacing individual devices with a centrally managed pool of hardware tokens, signing operations can now be coordinated across all locations and fully audited. Compliance audits become considerably simpler—and IT administration does as well.

Cloud Deployment Without Compromising Security

This model can also be implemented in cloud environments, provided the underlying network architecture is designed appropriately. USB traffic is transmitted over dedicated TCP/IP connections protected by VPNs, firewall policies, or network segmentation.

Additional safeguards, including encrypted TLS communication and password-based access controls, ensure that certificates remain protected even when development workloads run outside the organization's own infrastructure.

This approach is particularly well suited to hybrid environments, where signing operations can be centralized while computing resources remain distributed and scalable.

Organizational Benefits

Fewer certificates mean less administrative overhead. Centrally managed hardware tokens are easier to monitor, access can be comprehensively logged, and security processes can be standardized.

For larger development organizations, this creates transparency where ad hoc workarounds and inconsistent practices previously prevailed.

Security and Efficiency Are Not Mutually Exclusive

Hardware-protected code-signing certificates have become an essential security requirement. At the same time, modern software development depends on flexibility, automation, and scalability. Network-based access to USB security tokens bridges these two requirements.

Organizations benefit from reliable build pipelines, lower infrastructure costs, and full compliance with regulatory requirements—without sacrificing the agility of modern development workflows.

Looking Ahead

The importance of code signing will continue to grow. Software supply chains are becoming more complex, development cycles are accelerating, and regulatory requirements are becoming increasingly stringent. At the same time, organizations face growing pressure to automate every stage of the software delivery process.

Without the right infrastructure, code signing risks becoming a structural bottleneck. Limited scalability and rigid hardware dependencies are fundamentally at odds with modern DevOps practices.

Transparency and auditability are becoming equally important. Organizations must be able to demonstrate who signed which code, when it was signed, and under what conditions.

As a result, code signing is evolving from a standalone security tool into a core infrastructure service. Access to hardware-protected certificates is increasingly being treated as a controlled, scalable service that integrates seamlessly into modern development environments.

Ultimately, the goal is an environment in which code signing becomes almost invisible—not because it has become less important, but because it operates reliably and securely in the background.

References

- **DigiCert. (2024). Code signing changes in 2023: Private key storage requirements on FIPS 140-2 Level 2 / Common Criteria EAL4+ hardware.**
- **DigiCert. (2025). New private key storage requirement for Code Signing certificates.**
- **CA/Browser Forum. (2025). Code Signing Baseline Requirements (CSBR): Private key protection and hardware requirements.**
- **Entrust. (2022). CA/Browser Forum updates: Requirements for code signing certificate private keys.**

SEH

Made in Germany

vmware PARTNER TECHNOLOGY ALLIANCE

citrix Ready™

SEH Computertechnik GmbH Suedring 11 33647 Bielefeld Phone: +49 (0)521 94 22 6-0 Fax: +49 (0)521 94 22 6-99 Internet: www.seh-technology.com Email: info@seh.de

SEH

Stand 07/2026