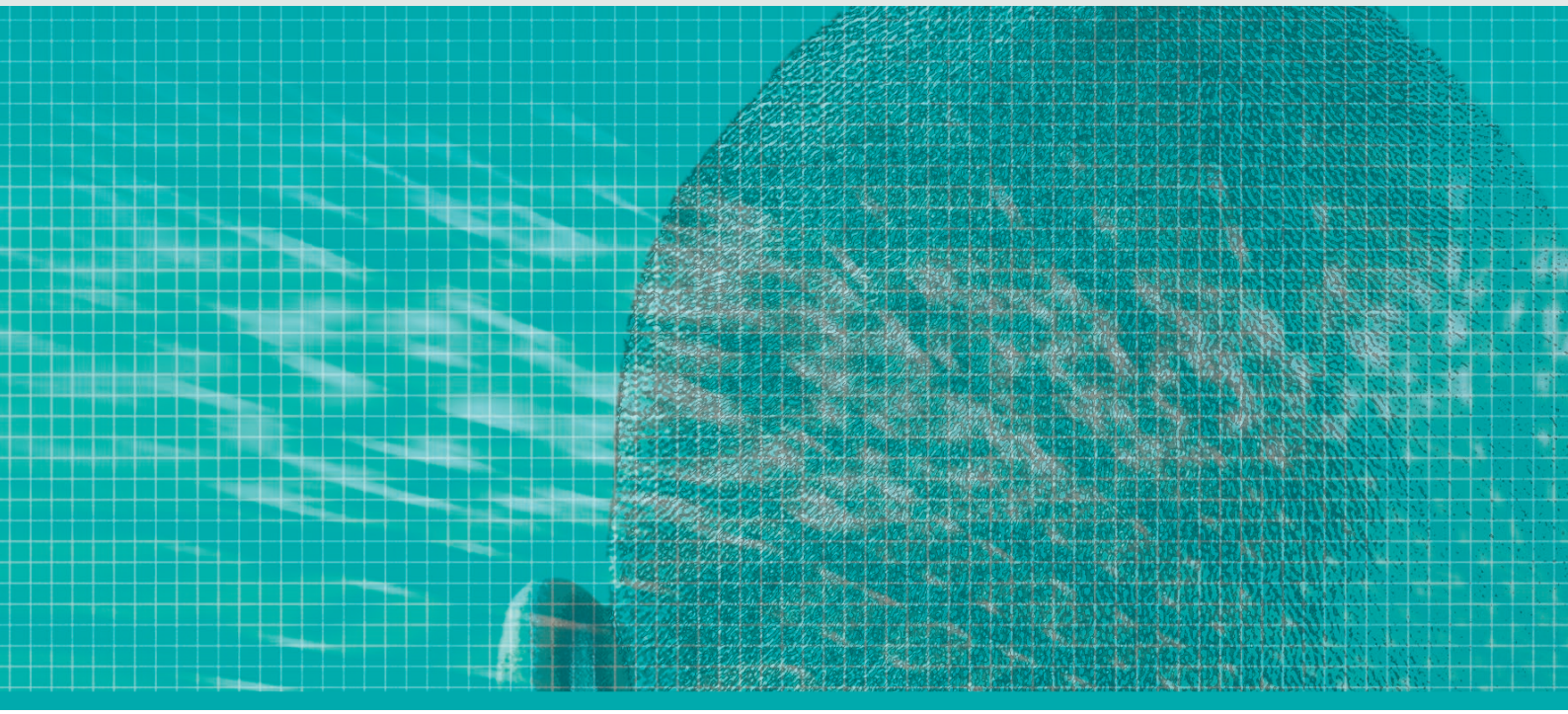
White Paper



Software Integrity Protection

Wibu-Systems mission statement for the Cyber Alliance

Oliver Winzenried, CEO WIBU-SYSTEMS AG
www.wibu.com

WIBU SYSTEMS

Content

Author:
Oliver Winzenried is a security enthusiast with a vocation to expand universal knowledge and apply innovative technologies to protect the intellectual property and business revenues of ISVs. With a degree in Electrical Engineering achieved at the University in Karlsruhe, he has immediately started an entrepreneurial career in electronic and ASIC design, hardware, microcontroller and embedded application development for consumer electronics, automotive and industrial engineering. Together with Marcellus Buchheit, in 1989 he then founded WIBU-SYSTEMS AG, whom he's still the CEO of. His passion for software integrity has resulted in numerous patent awards that span across secure license management and dongle feature innovations. He's also a prolific author, greatly contributing to editorials and books on the one hand, as well as addressing large audiences at trade shows, conferences, industry associations and technology centers like the Fraunhofer Institute. His personal involvement in international R&D projects and organizations for standardization, such as the SD Card Association, tops his profile off. Oliver Winzenried is also serving as chairman in the Product Protection and Know-how Protection "Protect-Ing" committee of VDMA, in the board of directors of BITKOM as well as in the managing board of FZI at KIT.

The Alliance for Cyber Security

WIBU-SYSTEMS is actively involved with the Alliance for Cyber-Security, an initiative of the German Federal Office for Information Technology Security (BSI), which was founded in cooperation with BITKOM, the voice of the Information Technology, telecommunications and new media industry in Germany.

The mission of the Alliance is to provide current, valid and complete information. With this goal in mind, it is collecting a great store of knowledge and expertise from the members which will be shared in the form of a comprehensive knowledge base supporting the exchange of information and experience.

The following document contributes to the understanding of the main foundations underlying the use of integrity protection for electronic computers in a fully dedicated technical environment, the embedded systems.

It is crucial that the know-how delivered with these devices be protected against reverse-engineering; more importantly the ever-growing threat of economic crime and cyber-wars is a problem that should also be tackled.

Integrity Protection for Embedded Systems

Control systems are increasingly interconnected and communicate via public networks. Cyber-Physical systems merge the analog world with the physical world, and the digital world with virtual reality.

This is also referred to as "Industry 4.0", a trend which offers not only comfort and added value, but also opens the systems to the outside, thus intensifying the risk of attacks from outer sources.

This added risk is real and gives rise to a need for additional counter measures; not only to prevent the loss of intellectual property but to prevent the introduction of malware through malicious code tampering. By using the right integrity protection measures security layers can be added to accomplish these new protection tasks.

What is Integrity Protection?

The term "Integrity Protection" encompasses security measures, namely protection of system resources, programs and data against unauthorized manipulation, or at least identification and display of such modifications.

The challenge consists in guaranteeing data integrity, and, if not possible, bringing the system to a safe mode and stopping the execution of any function.

The best integrity protection solutions are based on cryptography and associated security mechanisms, such as digital signatures and message authentication.

Where are the Touch Points for Copy and Know-How Protection?

Wich copy protection we intend to prevent the counterfeiting of complete machines and equipment. IP protection means the protection of the proprietary algorithms and methods from reverse-engineering.

To effectively withstand the increase in counterfeiting of complete systems, the protection of software and data of these systems gains critical importance.

Currently software represents a significant investment; in addition the software innovations in machinery and plant engineering are on the rise.

The automotive sector alone is responsible for 90% of all electronic- and software- driven innovations; it is estimated that the software-based added value will increase to 40% over the next few years.

The results of a survey lead by VDMA in 2012 show a sales shortfall of 7.9 billion euro in 2012. 9 out of 10 companies are affected by product piracy and 48% of the manufacturers are suffering from reproduction of complete machines.

28% of the respondents said that they are ready to implement technical protection solutions. 37,000 jobs could be created in the German machinery and plant engineering industry, if only product piracy was reduced. The following chart shows the rising trend in recent years.

An effective software protection solution is therefore a fundamental requirement for the protection of product innovations.

As the digitalization of production systems increases, the relevance of digital data protection in production and, more generally, of networked data protection and systems integrity will play a much greater role.

New solutions based on strong encryption and embedding secure hardware elements (like dedicated ASICs and smart card chips) can be used for software protection, IP protection as well as integrity protection.
As the digitalization of production systems increase, the relevance of digital data protection in production and, more generally, of networked data protection and systems integrity will play a much greater role.

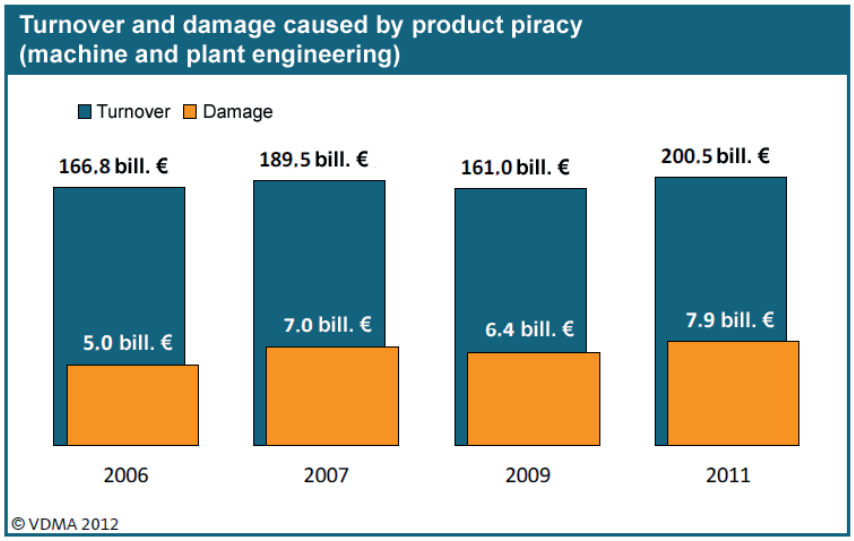


Figure 1: Revenue and damage caused by product piracy (source: VDMA)

New solutions based on strong encryption and embedding secure hardware elements (like dedicated ASICs and smart card chips) can be used for software protection, IP protection as well as integrity protection.

Basic Glossary of Cryptography:

To better understand the processes described below, here is a brief explanation of some terms:

Symmetric Encryption: With symmetric cryptosystems both parties use the same key or a simply derived key. Examples of a symmetric cryptosystem are FEAL (Fast Encryption Algorithm), IDEA (International Date Encryption Algorithm), DES (Data Encryption Standard) or AES (Advanced Encryption Standard). The advantage of a symmetric cryptosystem is a high encryption rate, while the disadvantage or the challenge is the secure key exchange.

Asymmetric encryption: With asymmetric cryptosystems, also referred to as PKI (Public Key Infrastructure), the parties don't have to share a common secret key in order to communicate. Unlike symmetric cryptosystems, a user creates a key pair consisting of a secret part, the private key, and a nonsecret part, the public key. The public key allows anyone to encrypt data for the holder of the private key to verify his digital signature or for authentication. The private key enables its holder to decrypt data which has been encrypted with the public key, to create digital signatures and to authenticate. Examples of asymmetric algorithms are RSA (Rivest, Shamir and Adleman) or ECC (Elliptic Curve Cryptography).

Hash functions: A hash function is a function that maps a string of arbitrary length to a character string of fixed length. A cryptographic hash function is a special form of the hash function, which is additionally collision-resistant or a one-way function (or both). Hash functions are used to verify the integrity of files or messages; to obfuscate password files; as a basis for digital signatures; as pseudo-random number generators and for the construction of block ciphers. Examples are MD5 as well as SHA-1 and SHA-256.

Digital Certificates: A digital certificate is a digital record that guarantees the identity for certain characteristics of people or objects, and whose authenticity and integrity can be verified through cryptographic methods. The digital certificate contains in particular the data necessary for its verification. Widespread are the public key certificates compliant with the X.509 standard, which confirm the identity of the holder, and other properties of a public cryptographic key.

Attacks to Cyber-Physical Systems

In order to develop effective methods aimed at preventing attacks, the threat scenario should be known. Some of the possible attacks to embedded systems are listed here below:

1) Attackers develop a **"fake device"**, a device that looks just like the original, but whose functions have been altered for nefarious purposes, that could be installed for example as a replacement part during equipment service.
2) Attackers develop their **own software** and run it by replacing the memory card in the embedded system.
3) Attackers extract the memory card out of the embedded system, **manipulate** the software and plug the card back into the system.
4) Attackers modify the software on the embedded system by controlling the **communication interfaces** from the outside.
5) Attackers monitor an **embedded system**, while in use by the application, in order to analyze it and to develop avenues of attack.

Development of an Embedded System and its Challenges

In principle, a typical embedded system is designed in different shells. It has to be kept in mind that an outer shell can access the memory of an inner shell, whereas the opposite is not possible in most cases. The outer shell (hardware / boot loader) is the initial shell of the overall process.
The sequence below outlines the actions needed to implement an integrity protection system. The step-by-step guidelines show how the integrity can be guaranteed level by level. The procedure is described in greater detail in the following sections.

1) The **boot loader** verifies the integrity of the operating system and loads it after validation.
2) The operating system only starts once the boot loader has been validated to be trust-worthy through a reverse check.

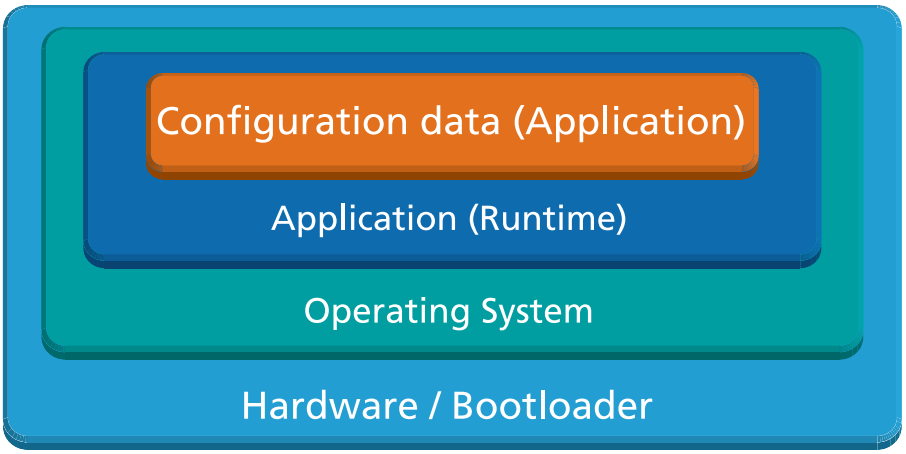


Figure 2: Basic structure of an embedded system

3) The **operating system** verifies the integrity of the application and loads it only if it has been validated.
4) The **application** only starts if the operating system has been validated to be trust-worthy through a reverse check. The application verifies the integrity of the configuration data and only uses them if they are validated.
5) Should the configuration data also contain executable code, a verification of the application level of trust is also possible.

Implementation of the Integrity Check

First, the unprotected original software has to be signed and encrypted in accordance with the following procedure:

The AxProtector, a commercial tool for software protection, will be used for the following steps:

1) Calculation of the **hash values** of the original software
2) **Signature** of the hash value with the private key of the vendor.
3) **Encryption** of the Originals Software using a key that is generated from a seed value within the original software, a secret key of the vendor and some other parameters that the publisher selects.
4) Attachment of the public portion of the **signature certificate** to the encrypted software.

The first part of the integrity check (forward check), i.e. the verification of the software to be loaded or the corresponding data, is carried out as follows:

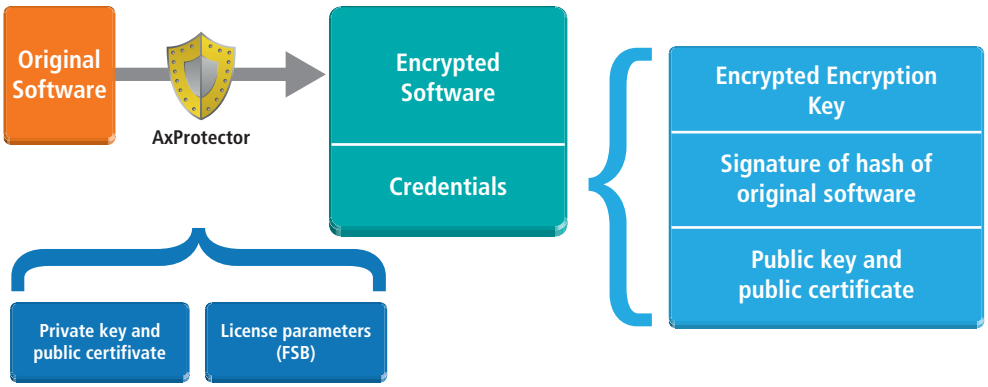


Figure 3: Encryption and signature of an application

The verification consists of the following steps, which are executed while the application is being loaded. This is performed with the aid of a system integrated tool, the AxEngine from WIBU-SYSTEMS.

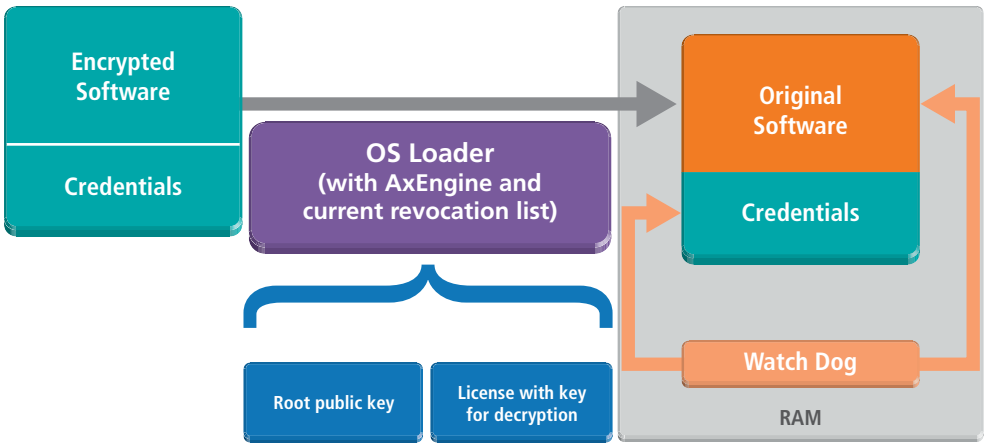


Figure 4: Integrity verification (Forward-Check)

1) If a valid license is present, the encrypted software is **decrypted**.
2) The certificate attached to the credentials or the certificate chain is **verified** against the public root key.
3) The hash value of the decrypted original software is **calculated**.
4) **The signature of the hash is verified using the public key.**

In addition to these necessary steps, further measures can be implemented increasing the security to a higher level, such as sophisticated handling of certificates for the authorized use of a specific device.
It is also possible to implement checks against a preset expiration date of the certificate, or the existence of a certificate revocation list - CRL. Such verifications can also be executed periodically at runtime in the system memory (watchdog).

The solution based on the CodeMeter technology covers the following steps, which can also prove especially effective in automated build processes:

- **Encryption** of the program code, to prevent static code analysis and reverse engineering
- **Signature** of the program code, of both the application and operating system image.
- Storage of the shared **secrets** for decryption
- Storage of the private signature key on the vendor's site.
- Signature and hash **verification** during loading and runtime operations.

Backward Check

This is the check that verifies whether the boot process was carried out correctly by the operating system. Unless this step is present, the integrity check of the operating system by the application is difficult to carry out because the subsequent step has only limited access to the previous step.

To compensate for the missed access, a state machine in a trustworthy hardware is needed. Such configuration is found in the Trusted Computing Group, TCG.

By using the so-called Trusted Platform Modules (TPM), it is possible to save correct states in registries. These registries include, for example, measurements of the boot loader, which will be considered later by the operating system to verify the integrity of the previous step.

The following chart shows the process of a backward check using a „Trusted Device", which manages the current status:

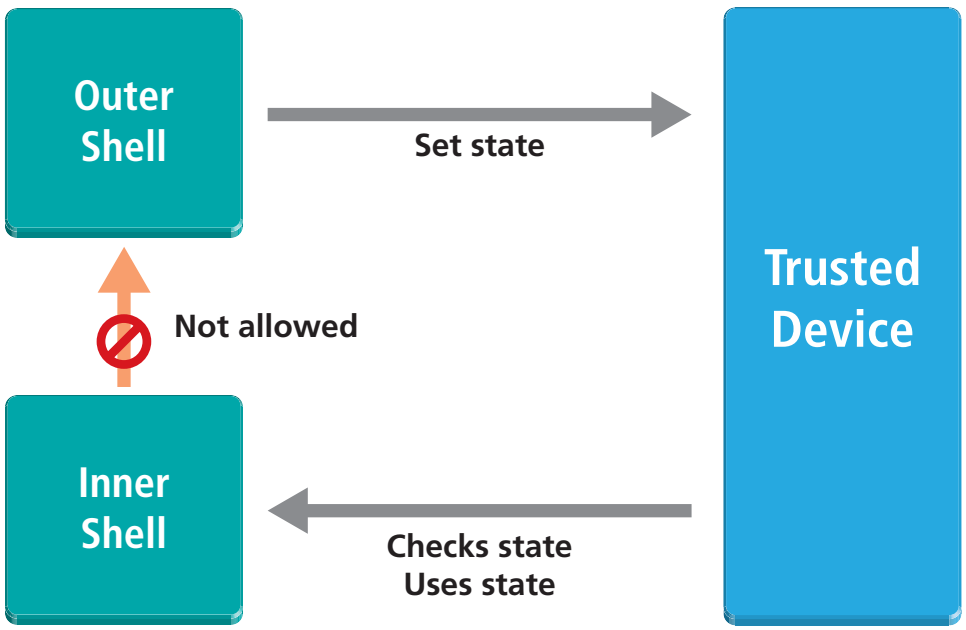


Figure 5: Backward check with a Trusted Device

The „inner shell" is, for example, the operating system, the „outer shell" of the boot loader. The trusted device, e.g. a TPM chip or CodeMeter dongle, stores the status of the boot loader. Only when this has been executed correctly, can the operating system start. This also applies to the subsequent stages. CodeMeter offers therefore a secure status machine. This feature is called "Enabling".

The decryption of the operating system, for example, will not be released until the integrity of the boot process has been validated; moreover the shared secrets are stored and not released until the previous step has been successfully executed for the next step to occur.

Pre-Boot loader – The First Step

A safe first step is essential, since all subsequent checks depend upon its accuracy. Attackers may in no case be able to decipher the code, or to extract secret keys. One solution is called "System on Chip" (SOC), in which these codes and keys are stored permanently on the chip, safe from any reading or manipulation attempted from the outside.

This compact pre-boot loader simply ensures the integrity of the actual boot loader. To prevent attacks from the outside, it is developed only once and cannot be updated on the system.

Chain of Certificates

Private keys are used for signing the program codes and parameters. These are saved in a secure hardware device, i.e. a CodeMeter dongle. The certificates are saved as files. They contain:

- The **Public Key.**
- Validity **restrictions**, e.g. expiration date or binding to a specific device.
- **The Purpose**, for example, to sign the boot loader, the application, the configuration files, or to create other certificates.
- **Certificate** of the key, which was used to create this certificate chain.

It seems complicated at first. But makes sense in actual practice. The so-called root certificate, which is at the top of the chain, is only used to create the certificates that will be utilized later in the process.

The root certificate is stored securely and only used when new certificates must be created. The root certificate must not be compromised under any circumstances whatsoever.

The chart below shows how the certificates for signing various program code and configuration files are derived from the root certificate and how the so-called "Certificate Revocation Lists", CRLs are created. This means that, if need be, through a simple update of this list, the certificates of all devices in the field are revoked or invalidated.

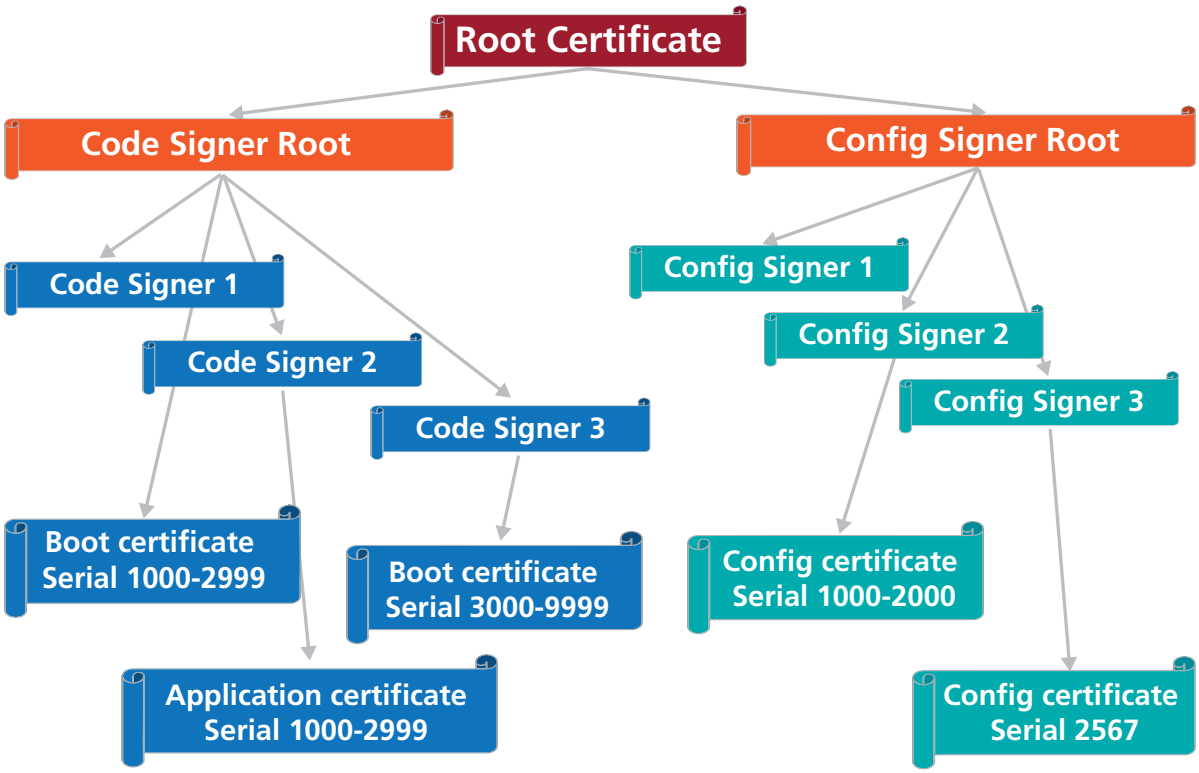


Figure 6: Graphic representation of a certificate chain

- **Root certificates** are valid indefinitely. They need to be locked in a safe place to prevent their loss and should only be used to create new Code Signing Root or Config Signing Root certificates. When the root certificate has been compromised, the devices must be physically replaced.
- **Code Signing Root Certificates** are limited in time. They are used to create the actual Boot Signing and Code Signing certificates. In the case of loss, new certificates can be created with the aid of the root certificate. Should the certificate be compromised, it can be added to a Certificate Revocation List (CRL) or forced invalid due to the expiration date. A CRL is present for example in the boot loader.
- **Boot Signing Certificates** are used to sign the boot loader.
- **Code Signing x-Certificates** are used to sign certain operating system images (like VxWorks) or applications. These certificates receive additional parameters that are valid on specific systems only. These certificates can be revoked just like the Code Signing Root certificates.
- **Config Signing Root Certificates** are used to create a certificate for configuration data signature. **Config Signing x-Certificates** are used to sign configuration data.
- **CRL** stands for "Certificate Revocation List" and is used to revoke certificates. Such lists are distributed online or through updates.

Summary

The integrity of embedded systems can be ensured through the use of cryptographic methods in a clearly defined process and a secure hardware device for key management and state storage. With CodeMeter, Wibu-Systems offers a smart card-based protection system, which is available for industrial interfaces. CodeMeter supports common operating systems like Windows, Mac OS X, Linux as well as Windows Embedded, Real Time Linux, VxWorks and PLCs like CODESYS and more. It contains a secure implementation of symmetric and asymmetric encryption methods (AES, RSA, ECC) as well as hash functions (SHA-256), functions for signature validation (ECDSA) and a random number generator. CodeMeter includes all the available tools needed to implement all the steps described above for integrity protection, software protection and the prevention of code tampering.

Figure 7: Hardware protection CmDongles available in different form factors

Headquarters

WIBU-SYSTEMS AG
Rueppurrer Str. 52-54,
76137 Karlsruhe, Germany
Telephone: +49 721 93172-0
Fax :+49 721 93172-22
sales@wibu.com | www.wibu.com

WIBU-SYSTEMS Branch Offices

WIBU-SYSTEMS (Shanghai) Co., Ltd.
Shanghai: +86 21 556 617 90
Beijing: +86 10 829 615 60
info@wibu.com.cn

WIBU-SYSTEMS USA, Inc.
USA: +1 800 6 Go Wibu
+1 425 775 6900
sales@wibu.us

WIBU-SYSTEMS BV
Netherlands
+31 74 750 14 95
sales@wibu-systems.nl

WIBU-SYSTEMS NV/SA
Belgium | Luxembourg
+32 3 400 03 14
sales@wibu.be

WIBU-SYSTEMS LTD
United Kingdom | Ireland
+44 20 314 747 27
sales@wibu.co.uk

WIBU-SYSTEMS IBERIA
Spain | Portugal
+ 34 91 414 8768
sales@wibu.es

WIBU-SYSTEMS sarl
France
+33 1 73 03 04 91
sales@wibu.fr

5062-002-02/20130327

WIBU-SYSTEMS AG (WIBU®), a privately held company founded by engineers Oliver Winzenried and Marcellus Buchheit in 1989, is an innovative technology leader in the global software licensing market.

In its mission to deliver unique, most secure and highly flexible technologies to software publishers and industrial manufacturers, Wibu-Systems has developed a comprehensive, award-winning suite of hardware- and software-based solutions incorporating internationally patented processes dedicated to the integrity protection of digital assets and intellectual property. Wibu-Systems' product portfolio addresses a wide variety of application delivery models, including PCs, mobile, embedded automation, cloud computing, SaaS, and virtualized architectures.

Through its motto "Perfection in Protection, Licensing and Security", Wibu-Systems is standing up for ethically produced software and reinforces its dedication to eradicate software counterfeiting, reverse-engineering, code tampering, as well as device and smart factory sabotage, espionage and cyber-attacks.

Headquartered in Karlsruhe, Germany, Wibu-Systems holds subsidiaries in Seattle, USA, as well as in Shanghai and Beijing, China; the company also has sales offices in Belgium, France, the Netherlands, Portugal, Spain, the United Kingdom and a capillary world distribution network.

SECURITY
LICENSING
PERFECTION IN PROTECTION

WIBU SYSTEMS