Concept of a Life-Cycle Management with Tamper Resistant Distributed Cyber-Physical Systems

Andreas Puesche[1], David Bothe[2], Sabine Sachweh[1], Norbert Pohlmann[2]
[1] Institute for the Digital Transformation of Application and Living Domains
University of Applied Sciences and Arts Dortmund
{andreas.puesche, sabine.sachweh}@fh-dortmund.de
[2] Institute for Internet Security
University of Applied Sciences Gelsenkirchen
{bothe, pohlmann}@internet-sicherheit.de

In this paper we will provide an overall concept of a life-cycle management targeting software components in the field of cyber-physical systems (CPS) from their Internet of Things (IoT) components to the back-end infrastructure. It combines the software development with the operational life-cycle and depicts a high level view on secured update mechanisms with hardware supported security features. We will describe a secure way of commissioning CPS devices within a critical infrastructure, up to decommissioning these components, respecting given privacy laws to prevent unwanted data recovery before destruction.

1 Introduction

Cyber-physical systems (CPS) are part of our infrastructure [1] and are commonly incorporated in smart energy grids, water management, pipelining, industrial manufacturing and medical systems. CPS are a combination of physical, mechanical and computational components that represent a real world entity, e.g. heating pumps or wind turbines [2]. This is accomplished by incorporating sensors and actuators to process domain specific data. CPS that communicate over untrusted networks such as the Internet contain Internet of Things (IoT) components for communication [3]. Managing and monitoring

CPS in a large distributed system is a mandatory task to guarantee certain levels of service quality and availability. Unfortunately, each IoT component of a CPS is prone to attacks if not properly protected. Weak communication channels and hardware tampering can disrupt a CPS in its functionality and even fall victim to false data injection (FDI) which can lead to information hiding in actuator states [4]. Before CPS were openly communicating over the internet, they were commonly bound to local networks, which was regarded as properly safe, due to access restrictions and highly specialized hardware. Sophisticated attacks like the Stuxnet Worm [5] could still target highly specialized systems in critical high-security infrastructures. With smart building automization becoming more popular in private and business environments, CPS are playing an even more important role in every part of modern life. As CPS reflect physical entities, any digital threats that may arise during operation can have a definite impact on the real world [6]. To distribute such systems, it is important to gain an overview on all aspects of their life-cycle from development to manufactoring and finally releasing it into production. Besides security and safety aspects of CPS [7], data protection laws are coupled with the requirement list for CPS, as they become part of households and business facilities alike. In this paper we will provide a concept for securely commissioning CPS into a large distributed infrastructure including considerations of data protection aspects while using hardware based security to ensure data integrity.

2 Brief Introduction of Security Features

Security features are the key to dependable distributed systems. To allow a further understanding of our concept, this section briefly covers some incorporated features for device hardening, communication security and access management.

2.1 Trusted Platform Module

This concept concerns hardware security features. Each CPS device is assumed to have a built-in hardware security chip. Even though *Hardware Security Modules (HSM)* are also suited, a much cheaper solution is available, the *Trusted Platform Module (TPM)*. TPMs are passive security chips that are specified by the *Trusted Computing Group* [8]. The functionalities range from

key generation to securing the whole boot process of a platform and are often embedded in a cryptographic context. Simply put, a TPM can be used to implement cryptographic measures to protect a system or specific components against digital attacks. A TPM is also capable of generating cryptographic keys as well as storing them in a secure location. The TPM encapsules a tiny cryptographic co-processor for encryption and decryption and provides a *True Random Number Generator*.

2.2 Transport Layer Security

The Transport Layer Security (TLS) protocol, accrued from Secure Socket Layer (SSL), is a secure way to communicate over untrusted networks. With the release of TLS1.3 [9], the list of supported symmetric algorithms has been cleared off legacy algorithms. The protocol enforces each peer of the communication channel to use a handshake procedure to negotiate a shared secret. After a successful handshake, the peers use this secret for traffic protection. The incorporation of secure and up to date end-to-end encryption protocols on every internet connection a CPS establishes, ensures a fluid and safe workflow of the CPS's data transmission. In our concept, this is the preferred way to process updates in the maintenance phase.

2.3 OAuth2.0

OAuth2.0 is an authorization framework [10]. It enables third-party applications to obtain access to a service and its resources by identifying access rights. *OAuth2.0* defines the following roles: *resource owner*, *resource server*, *client* and *authorization server*.

A *resource owner* can grant access to a protected resource. Assuming a CPS is rolled out to a home, the device acts as the *resource owner* which grants access to its generated data that is stored in a backend cloud system, the *resource server*.

A *resource server* hosts the protected resources. It can accept and respond to requests targeting the resources using access tokens. The application sending the request to the server takes the role of a *client*. After successfully authenticating the *resource owner*, the *authorization server* issues the access token to the *client*.

3 The Life-Cycle

The life-cycle of the CPS consists of seven phases as depicted in Figure 1. The subsequent sections will briefly describe all of the life-cycle phases of a CPS but concentrates on the most critical ones; *Registration, Activation, Production* and *Decommission.*

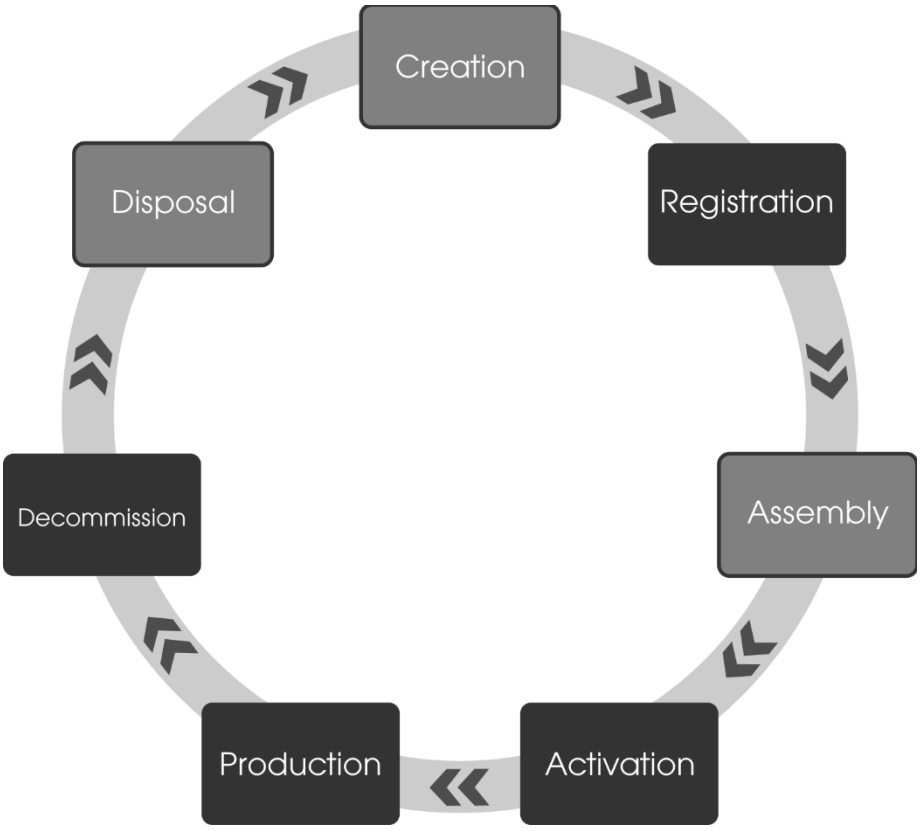

Figure 1: Device Life-Cycle.

Steps of the critical phases described in the following subsections are referenced directly in Figure 2, denoted by (n). All connections a device establishes over the Internet are assumed to incorporate at least TLS1.2. It is recommended to use TLS1.3 instead because it provides a more secure handshake negotiation between peers.

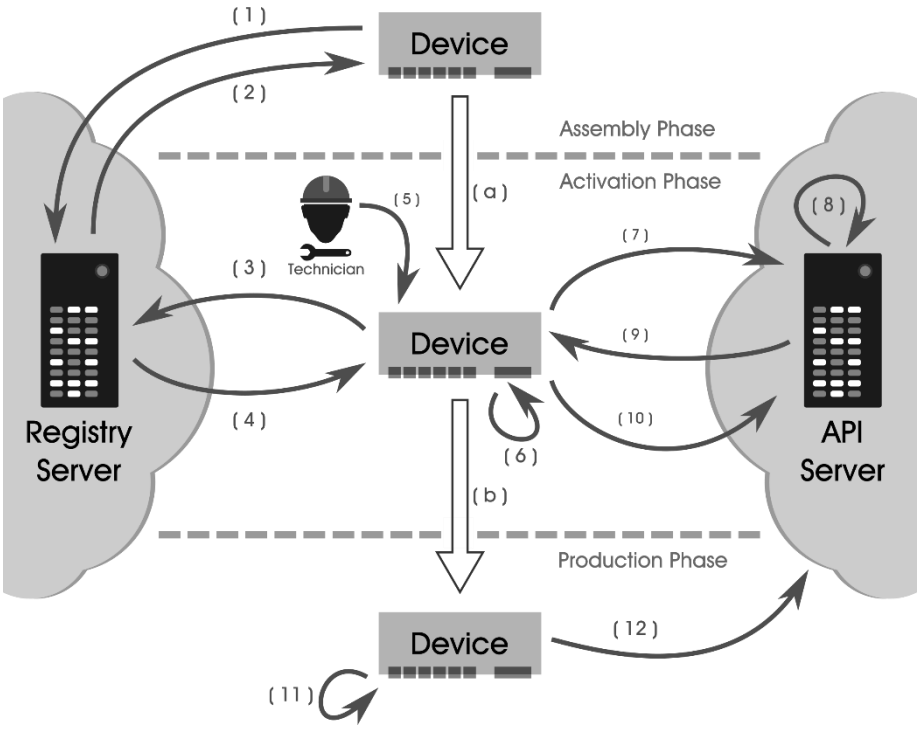

Figure 2: Setup-Cycle Overview.

3.1 Creation Phase

In order to ensure the flawless commissioning of endpoints, we assume that the hardware devices are rolled out by external manufacturers complying with certain policies and are fabrically sealed, according to FIPS-140 [11]. These devices will be flashed with a preconfigured firmware image of the desired system, containing all necessary software components concerning proper operation in terms of the devices workflow specification. The system image is assumed to be clean and devoid of unwanted, or unknown software and configurations. During this procedure, a unique identifier will be generated for each individual device which acts as its serial number and is stored in a secure location. For this purpose, a key will be generated by the TPM's *key generator* that is used to encrypt the serial number which is stored on the device. Since every key follows a strict hierarchy, the *Storage Root Key* from the TPM is needed to encrypt the file.

3.2 Registration Phase

Before entering *assembly phase*, the device needs to be manually registered with its serial number in the backend infrastructure by an authority (1). During registration, a unique one-time token alongside a new certificate as well as a key pair is generated based on a special commissioning public key infrastructure (PKI). The certificate represents the device's digital identity. Token T_0, a certificate and key pair $[k_{s,sec}, k_{s,pub}]$ needs to be trustfully installed on the device as a set of $[T_0, [k_{s,sec}, k_{s,pub}]]$ (2) for use in the *activation phase*. Server-side whitelisting of all commissioned endpoints takes advantage of the devices being vendor provided hardware. Customers are not able to obtain these endpoints elsewhere and no unregistered device will be in circulation. The server-side cloud system knows in advance which devices are commissioned and will enter the *activation phase* in the future.

3.3 Assembly Phase

Before the device can be activated, it needs to enter the *assembly phase*. In this phase, an appropriate technician will install the hardware device (a) at its service location and will connect all external components and their power supply. The technician will then establish an internet connection or other communication channels based on the device's workflow specification. The device has left the vendor at this stage thus losing control of the device's environment. To prevent possible attacks in this phase, the technician is equipped with countermeasures that are handled in the subsequent *activation phase*.

3.4 Activation Phase

The most critical phase in regard to security concerns is the *activation phase*. As the device is placed and assembled within a potentially insecure environment, it becomes mandatory to provide a specialized protocol to enter

the activation sequence. Upon first time system startup after the *assembly phase*, the one-time token T_0 and key pair $[k_{s,sec}, k_{s,pub}]$ generated in the *registration phase*, are used to authenticate the device during its first connection to the cloud system (3). A new setup token T_s will then be provided (4) by the cloud system while invalidating the previous token and key pair.

T_s will grant access to the API during *activation phase*. In addition, the technician provides his own authentication to the device in step (5) by $[T_s, S(T_s, k_{t,sec})]$ with $k_{t,sec}$ being the technician's private key, S creating the signature. It is recommended to utilize the capabilities of smartcards or smart-tokens to accomplish the technician authentication. This separated authentication sequence ensures that the backend system recognizes the activation of a distinct device by a particular technician. The device will then create a key pair $[k_{d,sec}, k_{d,pub}]$ and a certificate request CSR_{dev} (6) to be sent to the API server, together with the technician's authentication (7) by $[CSR_{dev}, T_s, S(T_s, k_{t,sec})]$. On success, the server will sign the certificate request with the PKI secret key $k_{p,sec}$ in step (8) using $S(CSR_{dev}, k_{p,sec}) \rightarrow CRT_{dev}$. The device will be provided with a new and final client certificate CRT_{dev} including a new bearer token T_{dev} (9) used for Open Authentication to gain API access.

As the communication channel is established and secured, the *Trusted Platform Module* (TPM) will be initialized following the *Trusted Computing Group* specification. This is accompanied by the initialization of all locally required cryptographic components, the private certificate key and the bearer token. All components are secured by the TPM through binding them to a TPMs storage key sk_{sec} using encryption E with $E(sk_{sec}, [T_{dev}, k_{d,sec}])$. The backend system is provided with all necessary device information which is required to enter the *production phase*. The TPM is capable of various secure mechanisms to accommodate private secrets, executing remote attestation, integrity measurements and more. Those aspects of the TPM are not covered in this document.

An essential part of the *activation phase* is the implementation of a recovery mechanism. Sequences as described above are prone to errors and may be interrupted or might fail in general. To face these issues, the activation sequence is implemented as a transaction. This will introduce rollback mechanisms which will reset the device to its initial state if the process becomes corrupted or fails. After successfully completing these steps, the device needs to be restarted and subsequently performs its first login under

runtime conditions. If the system successfully restarts, the installation process has been completed. The remaining setup token and registration certificate are deleted and invalidated on the device. In addition, they are revoked in the backend system. This action will also expire the technician's access to the device.

In short, the remaining generated components will consist of the setup token and certificate. The one-time token is invalidated as soon as the first connection to the cloud is established. The activation process then needs to be restarted and each step except the first-time connection needs to be repeated. If the first communication can be established successfully (10), the *activation phase* is completed, and the device turns over into *production phase* (b).

3.5 Production Phase

When entering the *production phase*, the device has been successfully installed and has been restarted (11). It establishes a secure connection with the backend cloud system and performs its first login under runtime conditions (12) using OAuth2.0.

The device then operates on behalf of its workflow specification. The *production phase* is completely independent from the device's behavior and domain logic considering cloud communication. However, continuous monitoring of the device state assures a smooth operation. The device will send regular reports containing the hard- and software configuration of the device to the cloud system. This can be accomplished by integrating a system daemon that will regularly perform integrity checks. As a passive chip, the TPM offers a set of possibilities to attestate system state and provide key management. On runtime, there is no possibility to verify the correctness of the boot procedure of the devices operating system.

It is assumed that the boot process was previously covered in the *activation phase* by initializing the TPM. During operation, the TPM can still be used to manage cryptographic keys and to attestate the integrity of specific system parts. In this regard, the TPM is utilized to verify the state of the client software that handles the domain logic of the CPS. To achieve this, the

checksum of each critical binary file from that client is computed. This checksum is then stored in one the registers of the TPM by extending it with each successive checksum. This extension of the TPMs registers must follow a specific policy. If the client was tampered with during operation, the checksum of the client's components will change. Any attempt to unseal the TPMs registers with the previous client state by the same policy will fail and the client is classified as manipulated.

This information is included inside the reports to the cloud system along other metrics about the CPS client. In return, the cloud system can decide how to handle this tampered device up to blocking communication between the CPS device and the cloud. The regular checks must not interfere with the client's domain logic.

When it comes to maintainability, updates are a critical part in distributed CPS environments and their IoT components. Not only arises the need to continuously advance development on the backend cloud system, each endpoint in this distributed system needs to be always up to date. In general, when CPS devices enter their destined operational state in their lifecycle, they operate without interference from real human users. IoT devices and CPS that are bundled with a specific service and are installed in the maintenance area of a building often need regular checkups from technicians. To cover both cases, the update mechanisms need to work autonomous on a regular basis. When updating binaries on the CPS, all binaries monitored for integrity will change the outcome of the integrity check. All updates must also be adopted to the TPMs values.

Beside a continuous health monitoring of the device, it is essential to provide regular and on demand software updates. This should be handled by a special service keeping contact to the update server and listening for update notifications. On receiving such a notification, an integrity aware update procedure needs to be initiated. Keeping a functional version of the software as a backup image for rollback in a failure case is mandatory during this transaction.

Secure Boot is a security mechanism that can be implemented with a TPM. As it shuts down the boot process if a minimal aberration occurs, it is not recommended for devices that need to be always available [12]. Instead, *Measured Boot* is provided by the system, which just records the boot process.

This is accompanied by continuous integrity checks during runtime. The integrity check will also allow the backend infrastructure to perform *Remote Attestation* at any time. This way, the cloud can verify the trustworthiness of the device before starting a regular communication. If the device is recognized as manipulated, the cloud can react to that by its own means, e.g. quarantined data collection, blocking the communication, issuing a technician to replace the device or commanding the device to operate in predefined error modes.

3.6 Decommission Phase

Similar to the *activation phase*, the deactivation protocol works mostly in reverse order. We assume that the hard- and software is still operational, otherwise, this phase is skipped, and the device must enter the next phase to be trustily disposed. An operator must flag a certain device to enter the deactivation procedure in the backend system. A particular technician is then assigned to shut down the specific device. Arriving at the operation environment of the device, the technician gains physical access to that device and is able to log into the system with a user certificate analogue to the *activation phase*. The technician then initializes the decommissioning.

As there will be a certain time difference between decommissioning and disposal, we consider this a critical phase. The device may still contain user and operator specific data and therefore falls under *General Data Protection Regulation (GDPR)*. As a consequence thereof, the following step will deactivate the device in the backend system by invalidating the tokens, unregistering the devices serial number and revoking certificates. Clearing the TPM will reset the chip, which will dispose all keys and further cryptographic data stored or protected by it. To ensure no data will be left on the device, the non-volatile memory of the system will be overwritten, using the device driver and device specific functions. Finally, the RAM should be cleaned by overwriting all remaining memory (technically excluding the currently processed memory). After completing these steps, the device has to be switched off. At the end of this phase the technician can dismount the device.

3.7 Disposal Phase

At this step, the only remaining component is the hardware itself, which has become useless in terms of the infrastructure. The hardware can now be recycled and the life-cycle of the components continuous with a new device.

4 Conclusion

As cyber-physical systems have a direct impact on the real world and play a huge part in modern life, distributing CPS requires a secure, safe and well-planned commissioning protocol. With the introduction of this life-cycle, we propose a concept to deal with advanced security mechanisms by introducing *Trusted Platform Modules* to utilize hardware security.

5 References

[1] S. Karnouskos, "Cyber-Physical Systems in the SmartGrid," 2011 9th IEEE International Conference on Industrial Informatics, Caparica, Lisbon, 2011, pp. 20-23.

[2] R. Rajkumar, I. Lee, L. Sha and J. Stankovic, "Cyber-physical systems: The next computing revolution," Design Automation Conference, Anaheim, CA, 2010, pp. 731-736.

[3] I. Stojmenovic, “Machine-to-Machine Communications With In-Network Data Aggregation, Processing, and Actuation for Large-Scale Cyber-Physical Systems,“ in IEEE Internet of Things Journal, vol. 1, no. 2, pp. 122-128, April 2014.

[4] S. Wendzel, W. Mazurczyk and G. Haas, “Dont You Touch My Nuts: Information Hiding in Cyber Physical Systems, “ 2017 IEEE Security and Privacy Workshops (SPW), San Jose, CA, 2017, pp. 29-34.

[5] S. Karnouskos: Stuxnet Worm Impact on Industrial Cyber-Physical System Security. In: 37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), 7-10 Nov 2011, Melbourne, Australia.

[6] E. K. Wang, Y. Ye, X. Xu, S. M. Yiu, L. C. K. Hui and K. P. Chow, "Security Issues and Challenges for Cyber Physical System," 2010

IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, Hangzhou, 2010, pp. 733-738.

[7] Sabaliauskaite G., Mathur A.P. (2015) Aligning Cyber-Physical System Safety and Security. In: Cardin MA., Krob D., Lui P., Tan Y., Wood K. (eds) Complex Systems Design & Management Asia. Springer, Cham

[8] Trusted Computing Group, “TPM 2.0 Library Specification, “ 2016, [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-libraryspecification/ [Accessed: Aug. 01, 2018]

[9] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3“, RFC 8446, 2018, [Online]. Available: https://tools.ietf.org/html/rfc8446 [Accessed: Sep. 12.2018]

[10] D. Hardt, “The OAuth 2.0 Authorization Framework“, RFC 6749, 2012, [Online]. Available: https://tools.ietf.org/html/rfc6749 [Accessed: Sep. 23, 2018]

[11] NIST, “FIPS General Information“, 2010 [Online]. Available: https://www.nist.gov/itl/fips-general-information [Accessed: Sep. 24, 2018]

[12] A. Hoeller and R. Toegl, “Trusted Platform Modules in Cyber-Physical Systems: On the Interference Between Security and Dependability, “ 2018 IEEE European Symposium on Security