Methods

Sören Finster*, Abdallah Dawoud, Florian Kohnhäuser and Abdulkadir Karaagac

Secure bootstrapping for next-gen industrial automation systems

Sicheres Bootstrapping für die nächste Generation industrieller Automatisierungssysteme

https://doi.org/10.1515/auto-2023-0074
Received April 28, 2023; accepted July 25, 2023

Abstract: The digitalization of industry and the convergence of IT and OT bring about the next generation of industrial automation systems which are expected to work with an orchestration of physical and virtualized components using a single converged network. The increase of complexity in such systems must be managed by an increase in automation for orchestration and management. However, bootstrapping such a complex system from out-of-the-box components is still a manual and error-prone process. We present a bootstrapping concept that brings up a system from out-of-the-box components to an operational solution with physical and virtualized components. The concept combines incremental network discovery with secure incremental bootstrapping of discovered physical components. The gained trust in the physical components of the network is then used to translate this trust into virtualized components. By attesting the trustworthiness of hosting infrastructure, the concept allows for virtualized components to be securely assigned a cryptographically secure identity that can be used in further application onboarding. Such securely bootstrapped systems are then capable to deliver the required adaptable, modular, and secure automation solutions of the future.

Keywords: secure bootstrapping; orchestration; industrial automation systems; secure identities; network discovery; virtualization

Zusammenfassung: Die Digitalisierung der Industrie und die Konvergenz von IT und OT bringen die nächste Generation industrieller Automatisierungssysteme auf den Weg. Diese Systeme arbeiten mit orchestrierten physischen und virtualisierten Komponenten über ein einziges konvergiertes Netzwerk. Die zunehmende Komplexität solcher Systeme muss durch eine verstärkte Automatisierung für Orchestrierung und Management bewältigt werden. Die initiale Einrichtung eines solch komplexen Systems ist jedoch immer noch ein manueller und fehleranfälliger Prozess. Wir stellen ein Bootstrapping-Konzept vor, das automatisiert aus Standardkomponenten ein betriebsbereites komplexes System mit physischen und virtualisierten Komponenten sicher einrichtet. Das Konzept kombiniert eine inkrementelle Netzwerkerkennung mit einem sicheren inkrementellen Bootstrapping der gefundenen physischen Komponenten. Das erreichte Vertrauen in die physischen Komponenten wird dann in virtualisierte Komponenten übertragen. Durch Attestierung der Hosting-Infrastruktur ermöglicht das Konzept die Zuweisung kryptografisch sicherer Identitäten an virtualisierte Komponenten, die dann das reguläre Onboarding durchlaufen können. Ein auf diese Weise sicher eingerichtetes System ist dann in der Lage, anpassungsfähige, modulare und sichere Automatisierungslösungen der Zukunft bereit zu stellen.

Schlagwörter: Secure Bootstrapping; Orchestrierung; Industrielle Automatisierung; Sichere Identitäten; Network Discovery; Virtualisierung

***Corresponding author: Sören Finster**, ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Deutschland,
E-mail: soeren.finster@de.abb.com. https://orcid.org/0000-0002-4611-7146
Abdallah Dawoud, **Florian Kohnhäuser and Abdulkadir Karaagac**, ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Deutschland, E-mail: abdallah.dawoud@de.abb.com (A. Dawoud), florian.kohnhaeuser@de.abb.com (F. Kohnhäuser), abdulkadir.karaagac@de.abb.com (A. Karaagac). https://orcid.org/0000-0002-0084-2246 (F. Kohnhäuser). https://orcid.org/0000-0003-4924-640X (A. Karaagac)

1 Introduction

The digitalization of industry and the convergence of IT and OT bring about the next generation of industrial automation systems. They will empower industries to tackle the demanding and dynamic landscape of future applications in industries. To tackle future demands for flexibility, next generation industrial automation systems are envisioned as independent software modules with defined

communication interfaces [1, 2]. They will deliver adaptable, modular, and secure automation solutions using standard-based, flexible, software-based orchestration of physical and virtualized components.

With Security by Design and Zero Trust Architectures as guidelines, delivering such a complex system of physical and virtualized components is challenging, labor-intensive, and error-prone. It is crucial to address the complexity of modern systems not with outdated and inefficient manual procedures but with modern, automated, and secure functionality. The increasing complexity of systems and networks should be matched by increasing automation in orchestration and management to prevent an increase in costly human effort and equally costly human errors.

A crucial part of next generation industrial automation system will be the converged network, that establishes secure, reliable and functional connectivity between all physical and virtualized components. Most modern network components in such a converged network can no longer be treated like cables that are bought, physically installed, and forgotten. Switches, firewalls, and routers are increasingly complex systems and play a crucial part not only in function and performance of a system but also in its security properties.

Fitting to their increase in functionality and role, network components themselves are getting more complex in configuration and administration. Many modern network components take an effort in configuration and administration comparable to server systems. Most of this increase in complexity is to be managed through intelligent network orchestration. However, adding out-of-the-box components to network orchestration is still a manual and error-prone process with many possibilities for human errors resulting in insecure configurations.

We present an automated, secure bootstrapping concept that is capable of securely bringing up a next generation industrial automation system from out-of-the-box components to an operational, orchestrated solution with both, physical and virtualized components, securely and automatically bootstrapped.

The remainder of this article is structured as follows. In Section 2 we discuss the pressing need for automatic network and software bootstrapping triggered by next generation industrial systems. Then, we detail our solution for securely bootstrapping networks in Section 3 which forms a foundation for secure bootstrapping of virtualized software, discussed in Section 4. Both Sections 3 and 4 first motivate the addressed problem and then describe the technical solution. We conclude in Section 5.

2 Next-generation industrial automation systems

With the digitalization of the industry and the convergence of IT and OT, industrial automation systems are going through a transformation that is shaping the control system of tomorrow and already empowering industries to respond to the increasingly demanding and dynamic landscape of future industries [1]. In parallel with this transformation, a change in automation systems is also being motivated by an abundance of user-driven initiatives toward openness, security and interoperability.

For instance, the Open Process Automation Forum (OPAF) is a 110-member consortium of end-user companies and automation providers, which is working to define a standard-based, open, secure and interoperable architecture. In line with the OPAF's vision for open control systems with independent software modules and defined communication interfaces, future process automation systems are becoming virtual and modular as containerized functional entities. These entities will be flexibly deployable, dynamically available and independent of system hardware. Connectivity will be provided with industry-standard secure protocols and interoperable technologies. To this end, the resulting converged and secure system and network architecture for next generation industrial automation systems is illustrated in Figure 1.

Delivering such adaptable, reliable, integrated, modular, scalable, and secure automation solutions requires flexible, simple, and automated system and network operations and processes that can replace the outdated and inefficient manual procedures while hiding the increasing diversity and complexity of systems and networks. Automation and orchestration are key technologies from IT to tackle these challenges. However, there are differences between IT and industrial applications that still provide for challenges. One example is the initial setup of each industrial application from out-of-the-box components on site, that has no equivalent in orchestrated IT.

For this purpose, there is a need for a software-based bootstrapping mechanism and workflow for systems and their networks, which will allow for interoperable and efficient future Distributed Control Systems. That solution needs to be able to create the foundation for secure industrial systems that meet latest security requirements defined by industrial standards in order to properly deal with the evolving threat landscape. While doing that, it needs to utilize latest security technologies (e.g., advanced container isolation, zero-trust security) and apply operations and orchestration flows which are secure by design.

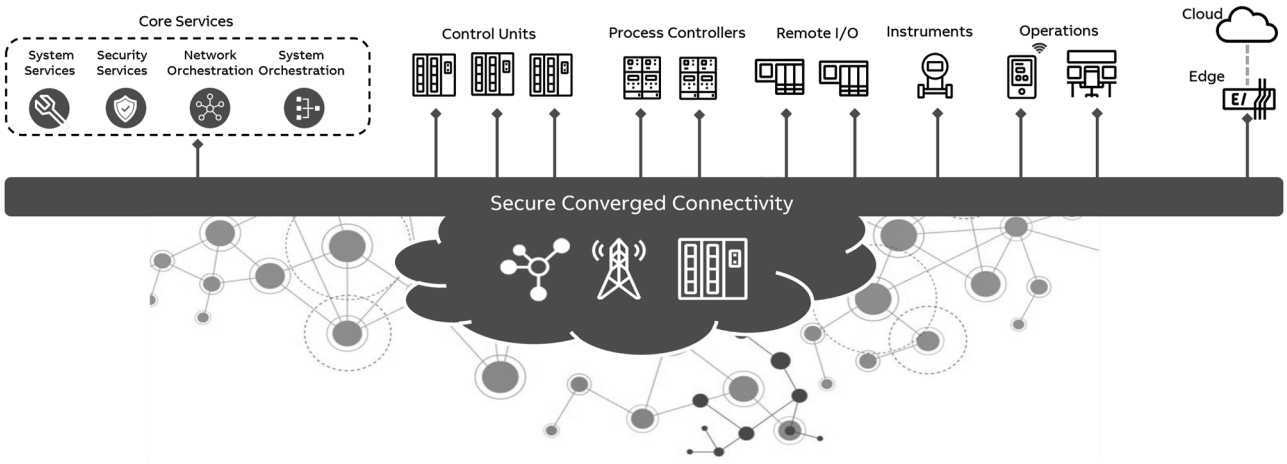


Figure 1: Unified, modular, flexible and secure system and network architecture for next generation industrial automation systems.

3 Secure network bootstrapping

Network Bootstrapping is a process that involves discovering, authenticating, and commissioning network devices to fulfill specific network and security requirements. For instance, starting from an unknown network topology of network devices running in an out-of-the-box factory default configuration, often called *commissioning mode*, network bootstrapping reconstructs the topology, verifies the authenticity of each device, manages authorization material, configures network requirements such as TSN and VLAN, and transitions devices into operational mode. As the number of connected network devices grows, the complexity of network bootstrapping increases. Especially with a mix of network devices from different vendors, of different age, and of different capabilities, this initial configuration of the network is a very complex and error-prone task. This complexity manifests in security, compatibility, and scalability challenges, hindering naive network bootstrapping techniques.

3.1 Problem description

As network devices become increasingly intricate, and with the requirement to accommodate legacy devices in real-world industrial networks, network bootstrapping has become an arduous task, particularly when executed manually due to the lack of automated approaches. Manual processes often leave behind residual and sensitive material, such as user accounts, credentials, certificates, and temporary configurations [3]. Some security risks stemming from poor management of sensitive material linger indefinitely (e.g., user accounts and insecure configurations), while others arise at a later and undocumented point in time (e.g., manually managed certificates). Regular maintenance of critical systems might include guidelines to mitigate some of those risks (e.g., checking for expired certificates [4]) but these processes only work for actually known and used configurations.

Due to its laborious nature, engineers are tempted to cut corners and omit security bootstrapping if functionality can be reached otherwise. For example, suppose a managed switch is installed but needs only to perform the functionality of an unmanaged switch. The managed switch will provide this functionality often out-of-the-box in its factory default settings. In that case, engineers might be tempted to leave the switch in the default configuration. However, the default configuration leaves the device as a permanent security risk in the network, open for an attacker to claim, configure, and exploit certain vulnerabilities that would not have been possible otherwise (e.g., local privilege escalation), maintain persistence in the network, and allow for lateral movement. All while staying undetected since the switch itself still provides all functional duties. Manual network bootstrapping additionally struggles in meeting reproducibility and extensibility requirements as knowledge of the network often remains undocumented and gets lost as the engineers who bootstrapped it move on.

Moreover, although network bootstrapping was once considered a one-time effort, the increasing complexity and convergence of IT/OT networks necessitate its recurrence. In particular, even minimal structural and configurational changes in a network require overall evaluation to avoid

conflicts and minimize attack surface through proper security configurations. Such continuous evaluation, however, poses practical challenges, especially when done manually.

Automated network bootstrapping, as an alternative to manual approaches, streamlines some of the manual processes but brings different challenges, including the need to ensure the authenticity of the commissioned device and the automatic discovery of neighbor devices, e.g., connected to Ethernet ports. As devices vary in complexity due to vendor fragmentation, some challenges are more relevant than others, necessitating a backward-compatible approach. For instance, when the authenticity of a device cannot be verified, manual authentication is inevitable but the commissioner should be sufficiently supported.

3.2 Solution

We propose a largely automated, secure network bootstrapping technique tailored to modern devices but also compatible with a wide range of legacy devices. The proposed solution has minimal requirements on network devices. It can be used by average network engineers in a fully or semi-automated fashion, depending on the level of complexity of the target devices. Besides reducing tedious manual effort, the proposed technique elevates network security by adhering to the zero-trust security principle as a defining design decision. The zero-trust principle places the security of network devices, and transitively the whole network, at higher standards, by limiting capabilities of local user accounts on network devices, ensuring strong authentication and secret management mechanisms, enforcing fine-grained access control, and establishing the authenticity of network devices participating in a network [5]. The proposed solution bridges a gap in industrial automation and paves the way for more rigorous secure network bootstrapping.

3.2.1 Components overview

As shown in Figure 2, the proposed solution, referred to as the *Network Orchestrator* hereafter, consists of four components, the Network Controller, Security Controller, Vault, and Web User Interface (Web UI). The Network Orchestrator interfaces with the users, who wish to bootstrap the network, and network devices, which are limited to switches in our prototypical design. We first introduce these components and then describe their interactions to securely bootstrap networks.

The *Network Controller* manages the overall bootstrapping process. It offers the necessary APIs to start, stop, and visualize the bootstrapping process through a Web UI (see Figure 2). It additionally accepts the network configurations from the users and deploys them on the target

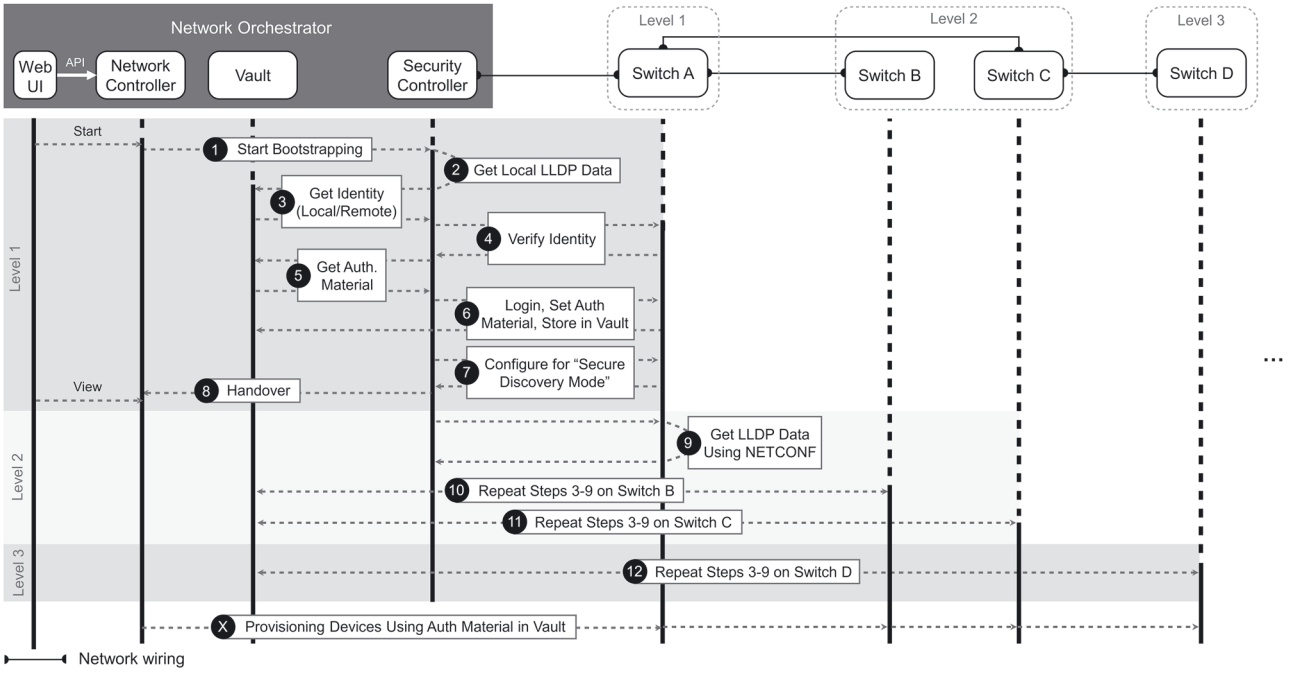


Figure 2: Execution flow for bootstrapping a network consisting of four network devices. The figure shows the network orchestrator that composes four basic components necessary for bootstrapping the network. It also shows the steps necessary to discover, authenticate, and commission a single network device (i.e., steps 3–9 and then step X). The discovery process follows the depth-first traversal algorithm (i.e., discovery of switches B, C, and D, in order).

devices as they get authenticated and securely configured during the process. The Network Controller also implements access control to segregate roles, e.g., auditors only have view access rights while network commissioners have edit rights. It also persists users' actions and requests to ensure accountability.

The *Web UI* gives users oversight and control over the bootstrapping process, leveraging the APIs exposed from the Network Controller. While bootstrapping modern network devices is fully automated, bootstrapping legacy devices requires manual intervention, in which users are supported with information-rich UIs that facilitate taking decisions and corresponding actions. In a corporate setup, the Web UI is the sole access point to the network bootstrapping process. It must be operated by securely authenticated users and should be protected via Single Sign On (SSO). This aligns with zero-trust architectures by using a single, strong source of user identity and avoiding user authentication on the level of network devices.

The *Vault* is a repository for sensitive material, including authorization and authentication material necessary to manage the devices in the network. It enforces second-layer access control on who can access what from the Vault, further elevating the security of the overall solution. It is a form of privileged access management (PAM) on the level of machine-to-machine authentication. When a user initiates operations on network devices through the Web UI, the network controller executes them using authorization and authentication material from the Vault.

The *Security Controller* has the responsibility of incrementally discovering the network devices, authenticating them, and handing them over to the Network Controller for further commissioning. Along the way, the Security Controller ensures that devices' default authorization tokens and potentially insecure default configurations are replaced with stronger ones. Authorization and authentication material is securely persisted in the Vault. It is worth noting that discovering network devices entails several security operations and, therefore, falls under the responsibilities of the Security Controller.

3.2.2 Bootstrapping process overview

The network bootstrapping process begins when the Network Controller receives a "start" request from the user via the corresponding exposed API triggered from the Web UI (see Figure 2). This request will be passed on to the Security Controller (step ❶), which has to be connected to an arbitrary device of the network for the bootstrapping process to work. The Security Controller will then query its local Link Layer Discovery Protocol [6] (LLDP) data (step ❷). The LLDP data describes directly connected devices, i.e., Switch A in this example, conveying its IP address, the address of a management endpoint, MAC address, chassis number, model, and manufacturer details. Using this information, the security controller configures its network interface accordingly to establish a connection to the management endpoint (e.g., by configuring an IP address and subnet).

At this point, the Security Controller attempts to authenticate switch A. Several options for authenticating the switch are possible, for example, using transport layer security certificates issued by a manufacturer CA or a public CA. In our prototypical implementation it queries a pre-stored manufacturer-issued SSH server public key from the Vault that corresponds with switch A (step ❸). Such key material could be pre-stored using trust from the Web PKI or directly derived from communication with the vendor. If no such authentication key exists, the Security Controller resorts to other techniques to validate the authenticity of the target device, which we describe in Section 3.2.3.

After validating the authenticity of switch A (step ❹), the Security Controller proceeds by accessing the management interface with default credentials or credentials pre-stored in the vault. It checks if the switch is in its default settings, i.e. commissioning mode, resetting default authorization material of the target device, and storing the new material in the Vault (steps ❺ and ❻). The Security Controller is then also responsible to change the network configuration to a configuration that avoids conflicts and ensures a disturbance-free progress of the bootstrapping process. For example, the Security Controller configures switch A with a separate VLAN per port, isolating the devices connected to Switch A from each other. The devices connected to Switch A remain accessible from the Security Controller since the Security Controller configures its own connecting port of Switch A to carry all frames VLAN tagged (i.e., a so-called trunk port). This not only prevents problems due to conflicting default configuration, but also provides a barrier for threat actors to access other devices. This leaves switch A with a defined, secure configuration and sets up connected devices for the further progress of secure discovery (step ❼). At this point, the Security Controller announces to the Network Controller that Switch A is trusted and ready to be commissioned (step ❽), so that the Network Controller can update the Web UI. This marks a checkpoint indicating that network devices of Level 1 have been fully discovered. However, the Network Controller waits for further orchestration until all devices are discovered and set up with secure configurations.

To discover further devices directly connected to Switch A, the Security Controller queries the LLDP data from Switch A using NETCONF [7] over SSH transport and authenticates and authorizes itself using material from the Vault. The LLDP data yield information about Switches B and C (step ❾). Following a depth-first traversal approach, the bootstrapping process now considers Switch B for authentication and basic configurations. The Security Controller sets up an interface with the corresponding VLAN configuration for switch B and then repeats steps ❸ to ❾.

Since Switch B is not connected to any further undiscovered devices, its LLDP data will not yield further information about unknown devices. Discovery will stop for this path. The exact same process then repeats for Switch C, marking a checkpoint for Level 2 devices, and then for Switch D, marking a checkpoint for Level 3 and a conclusion of the whole bootstrapping process. With all devices authenticated and corresponding authorization material persisted in the Vault, the Security Controller signals to the Network Controller, that all necessary information to orchestrate the network to the user's requirements is ready (step ❿).

3.2.3 Device authentication

The goal of this step is to make sure we are talking to the intended network device. This not only prevents misconfiguration due to errors, it also prevents malicious actors from infiltrating the network. To authenticate a device, the Network Orchestrator supports three mechanisms, depending on whether the device offers a verifiable secure identity (e.g., a DevID as defined in IEEE 802.1AR [8]) or not:

1. Automatic authentication does not require any manual effort but requires certain functionality from devices and organizational measures. A device needs to provide a secure identity and relevant information for checking the secure identity (i.e., public keys, certificates) existing in the Vault.
2. Semi-automatic authentication is when the device offers a secure identity, but the identity is not available in the Vault. Depending on configuration, manual intervention might be needed to confirm the identity and store it in the Vault. Alternatively, device authentication could be configured to automatically trust secure identities from certain vendors. Then, the device could be authenticated like in the automatic authentication step. However, information about this should be still conveyed in the Web UI.
3. Manual authentication is when the device does not offer a secure identity, a limitation common in legacy devices. In such a case, plausibility checking offers another automated verification layer, including checks on MAC address if it comes from the intended vendor and chassis ids and serial number matching on the device. A visualization of the device showing where it is located in the network is also shown. Such meta information about the device facilitates decision making and provide a more secure alternative than pure-manual authentication. Finally, the Network Orchestrator can only provide clues to the user if a device is authentic. The decision itself has to be taken by the user.

After a device is authenticated, its configuration is checked if it still in factory default. Only then, the device is ready and trusted for operation. If there are unexpected changes to the configuration, it is not considered secure. Depending on the device in question, a reset to factory default can be issued and device authentication restarted. However, this is only secure if the device in question supports a complete reset to factory defaults and measures to check for success (e.g., secure boot, remote attestation). Otherwise, human intervention is necessary.

3.2.4 Device authorization

Network devices are shipped with default credentials and users. Intuitively, keeping devices in such a state is insecure. In our design, we also consider non-default credentials that are not stored in the Vault to be a threat. We argue that such credentials, despite possibly being strong, can be easily leaked, as opposed to credentials stored in the Vault. To mitigate this situation, the Network Controller leverages default credentials to login into the devices for the first time. It then resets the login credentials to stronger ones and stores them in the Vault. It additionally removes unused users and invalidates existing sessions to preclude attackers who maintain active access. This effectively limits access to the devices to only the Network Controller and Security Controller, which are strongly protected behind an auditable access control policy, shielded behind an SSO scheme in our design.

3.2.5 Device commissioning

After all devices are discovered and authenticated, the Network Orchestrator configures the network according to user-defined requirements for networking and security configurations, such as establishing VLANs and setting up TSN. Configuring network devices, thanks to our proposed solution, is a centrally-managed process, as opposed to prevalent distributed and device-by-device approach. As a last step, the Network Orchestrator brings up all devices into

Table 1: Average execution time for core network bootstrapping operations in seconds. All operations use an RSA key of 2048 bits. The size of retrieved LLDP data is 2.49 KB.

Operation	Execution time (s)	Code
Identity verification	0.091	V
Administrative access	0.206	A
LLDP data retrieval	0.743	R

operational mode, restricting access and further modifications to the network to strongly authorized entities while adhering to a configurable access control policy.

3.2.6 Scalability considerations

Evaluating the execution runtime of automated versus manual network bootstrapping is unreasonable for the huge disparity in favor of the automated approach. However, it is beneficial to evaluate the absolute execution runtime of automated network bootstrapping to understand its scalability prospective. To that end, we present a hypothetical example of a network consisting of X switches. Leveraging the measured execution times of basic bootstrapping operations, we argue for worst-case execution runtime assuming no parallelization of execution available.

Setup: The network device used in the measurements runs a Dual-core ARM Cortex-A9 CPU with 1 GB RAM and offers an automatically verifiable identity. The average execution time over 100 runs of the core operations are summarized in Table 1. Notice that the execution runtime of all operations vary depending on the used cryptographic algorithm, the switch's processing power, and networking overhead. Similarly, retrieving the LLDP data depends on the size of the payload influenced by the number of directly connected switches.

In a worst case scenario, with X switches that are serially connected forming an unbalanced tree of depth X, the overall runtime execution, excluding time necessary for configuring the device as it is dependent on the device and type of configuration, is approximately

$$(A + V + R) * X \quad \text{(seconds)}.$$

Where A, V, and R are explained in Table 1 and X is the number of switches in the target network.

In this worst case scenario, the execution runtime is linear to the number of network devices participating in the network. However, a setup of serially connected switches is highly unlikely. In practice, the branching factor of switches is considered to be much higher. This enables opportunities for significant speedup by bootstrapping all network devices discovered in one step in parallel. Then, execution runtime becomes logarithmic to the number of devices following the branching factor of the network topology. Both best-case and worst-case execution runtimes are very affordable, compared to the manual execution runtime, rendering our proposed solution highly scalable.

3.3 Limitations

Some legacy network devices offer human-only interfaces to access and configure them (e.g., web configuration). Especially modern web user interfaces are usually not machine friendly since they are expected to be accessed in the browser and implement functionality client-side. This poses a limitation as our solution works best with machine friendly interfaces like SSH shells or NETCONF. Notice that workarounds exist to mitigate this issue, e.g., via machine controlled, instrumented browsers, which could improve the range of devices fit for the proposed solution. Another limitation is related to the discovery protocols. Our solution relies on network devices to support LLDP for discovery. Discovery of network devices without LLDP support is out of the scope of our solution but still feasible in an active discovery effort (e.g., using network scanning tools like nmap [9]).

4 Bootstrapping virtualized components

In next-gen industrial automation systems, components are often no longer physical devices that serve one particular purpose (see Section 2). Instead, components are being virtualized, which provides several benefits related to availability, flexibility, and scalability. However, securely bootstrapping virtualized components poses several challenges, since traditional solutions for physical components no longer apply. In this section, we first introduce the problem and then describe our solution for a secure bootstrapping of virtualized components.

4.1 Problem description

Secure bootstrapping solutions typically rely on manufacturers pre-installing cryptographic keys and digital certificates in components, which give the component a unique bootstrapping identity. During bootstrapping, automation

engineers and tools then verify these manufacturer-assigned identities to ensure the authenticity of new components before integrating them into the network. To ensure security, the necessary cryptographic material is typically stored on devices in hardware-protected memory.

In modern industrial systems, components are increasingly implemented as virtual software, so-called containerized applications. Existing bootstrapping techniques fail to address the security needs of such containerized applications and virtual devices, since pre-installed device identities cannot be used to authenticate containerized applications. The reason for this is that containers may run on various host devices and may even be shifted from one host to another host during operation. An alternative approach would be pre-installing cryptographic identities in the containerized applications themselves. However, this approach provides insufficient security, as an adversary with access to a container can copy it, obtain multiple components with the same bootstrapping identity, and thus impersonate the original component. This highlights the need for a solution to securely bootstrap software components, including containerized applications, in industrial systems.

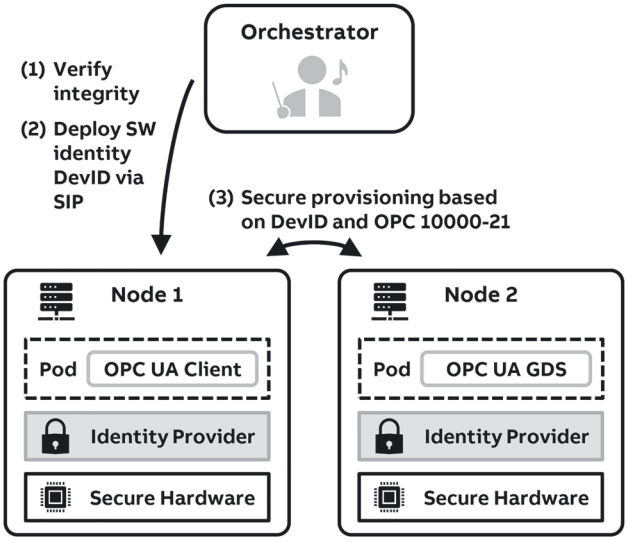


Figure 3: Illustration of the proposed secure onboarding process in case of OPC UA.

4.2 Solution

While existing solutions rely on pre-installed bootstrapping credentials [10], we enable the Orchestrator to assign bootstrapping identities immediately at deployment. The Orchestrator is a common paradigm in modern software architectures, which deploys and manages software containers running on a distributed cluster of host devices. Our solution is the first that combines secure bootstrapping identities with remote attestation and a secure hardware key storage for containers, to enable the secure bootstrapping of containerized applications by Orchestrators.

In detail, we propose to leverage secure hardware (e.g., a TPM) that is built into devices hosting containerized applications. Based on the secure hardware, a remote attestation procedure is implemented between Orchestrator and host devices. During attestation, the Orchestrator verifies that containerized applications run on legitimate devices in an uncompromised software environment. This way, our solution ensures that the Orchestrator assigns secure bootstrapping identities only to containerized applications running in trustworthy environments, i.e., environments that are not controlled by an adversary. Note that containerized applications themselves need not be verified [11, 12], as they are securely deployed by the Orchestrator and thus are trustworthy. In addition, host devices use the same technique to verify the integrity and authenticity of the Orchestrator.

To manage bootstrapping identities, we propose that host devices implement a so-called Software Identity Provider (SIP), which provides two novel features. First, it offers an abstraction layer to the underlying secure hardware, which ensures that each containerized application can only access its own bootstrapping key but not the keys of other applications. Second, it allows credentials to be securely moved from one host device to another, which is required when the Orchestrator shifts containerized applications in the cluster.

The secure bootstrapping of containerized applications is a 4-step process, illustrated in Figure 3. In the first step, the Orchestrator verifies the integrity of the host server and its software (see Section 4.2.1). The second step focuses on managing the identity of the volatile containerized applications (see Section 4.2.2). To enable secure communications between applications, the third step implants a trusted certificate in all containers (see Section 4.2.3). The final step describes the necessary measures for securely migrating applications across the host while maintaining functionality (see Section 4.2.4).

4.2.1 Integrity verification

Before installing containerized applications on a specific host device, the Orchestrator uses a technique called remote attestation to ensure the integrity of host devices and their installed software. Remote attestation is a security solution that allows a computer, the verifier, to check the software

integrity of another computer, the prover. During attestation, the verifier determines whether the software running on the prover is healthy or compromised, which enables the verifier to remotely establish trust in the prover. Attestation is typically implemented leveraging secure hardware, such as a TPM, ARM TrustZone, or Intel SGX. In our solution, the Orchestrator acts as a verifier and ensures that the host has not been compromised by an adversary, as it is being checked that the host only runs legitimate, unmodified, and up-to-date software. In addition, the Orchestrator can verify the legitimacy of the host device, e.g., whether it has been installed by a system integrator or is maintained by a cloud provider, if the host device has been certified by them. Only if the authenticity and integrity of the host device and its execution environment has been verified, the Orchestrator deploys the containerized application and provides it with a bootstrapping identity, as described in the next step. Additionally, host devices can use the same technique to ensure the integrity and authenticity of the Orchestrator. This prevents an attacker from impersonating an Orchestrator and misusing host devices, albeit this would be detected by the actual Orchestrator.

4.2.2 Bootstrapping identity assignment

To manage bootstrapping identities, host devices implement a so-called Software Identity Provider (SIP). The SIP requests and receives bootstrapping credentials from the Orchestrator and assigns them to a specific containerized application. Bootstrapping identities typically consist of a unique key and an associated digital certificate, e.g., DevIDs as proposed in IEEE 802.1AR [8]. The digital certificate is signed by a Certificate Authority (CA) that the Orchestrator manages. To protect the private key associated with the bootstrapping identity from unauthorized access by other containerized applications, the SIP commands the generation of a new key in hardware-protected storage (e.g., in a TPM). Thus, each containerized application holds its own key that is stored hardware-protected and managed by the SIP. The SIP only allows the designated application to use its own key. This prevents other applications, which may be compromised by an adversary after deployment, from accessing the keys of the containerized application. Figure 4 illustrates the bootstrapping identity assignment.

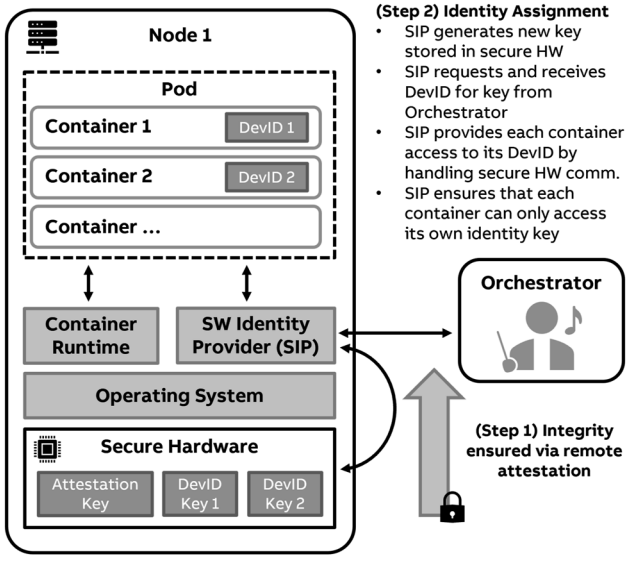


Figure 4: Illustration of bootstrapping identity assignment in details (step 2).

4.2.3 Protocol-specific secure bootstrapping

The provided secure bootstrapping identity enables the containerized application to authenticate itself towards other applications and devices in the network. This is used by existing solutions, such as FDO [13], BRSKI [14], OPC 10000-21 [15] or SZTP [16], to securely bootstrap the containerized application. For this step, the Orchestrator's CA certificate, from which the bootstrapping identities have been derived, constitutes a common root of trust in the network. Thus, it may be needed to import the Orchestrator's CA certificate as a trusted root certificate in devices and applications that are not managed by the Orchestrator. Alternatively, the Orchestrator may implement an intermediate CA whose certificate is signed by a well-known root CA that other devices or applications already consider as trustworthy. During bootstrapping, the containerized application is typically provided with protocol- and environment-specific credentials and settings, which enables the containerized application to operate as expected after bootstrapping [13–15].

4.2.4 Bootstrapping identity migration

Note that this step is only required when moving containers between hosts. After deployment, i.e., during operation, the Orchestrator may decide to shift a containerized application from one host device to another, e.g., due to shortage of resources on the original host device. As containerized applications should preserve their identities, issued bootstrapping credentials need to be migrated from the original host to the new host.

Some secure hardware, such as a TPM, allow the migration of keys. However, hardware key storage migration mechanisms are often slow and inconvenient to use, e.g.,

involving manual effort. For this reason, the SIP implements a migration protocol that is executed between the Orchestrator, the SIP of the original host, and the SIP of the new host. During migration, keys and certificates are deleted on the old host, and new keys and certificates are generated on the new host. The new certificates provide the same identity as the original certificates and are signed by the Orchestrator. For instance, in case of DevID certificates that are used for the secure bootstrapping of OPC UA, the new certificates make use the same ApplicationURI as the original certificates. Afterwards, a protocol-specific solution performs the actual bootstrapping with the migrated containerized application, as described in the previous step. This way, the migrated application gets new credentials, but still preserves its original identity that has been assigned by the Orchestrator during deployment.

5 Conclusions

In this article, we propose an orchestration system that will allow secure, automated bootstrapping of next generation industrial automation systems. By filling the automation gap between out-of-the-box components and orchestrated components with an automated and secure solution, we remove most of the tedious, manual effort and we remove immense room for human error with dire consequences in the security properties of these systems.

Our orchestration system combines incremental network discovery with secure incremental bootstrapping of discovered components. Using modern self-identifying and self-authenticating components, this allows for a completely automated, secure bootstrapping process with trust in each incremental step of network discovery. The resulting secure, orchestrated network then forms a trustworthy foundation for bootstrapping of virtualized components.

For securely operating virtualized components, our orchestration system provides them with a unique, secure identity during the bootstrapping process. The trust in these secure identities is derived from trust in all underlying components during bootstrapping. After secure network bootstrapping, our orchestration system uses remote attestation techniques to ensure, that the hosting systems for virtualized components themselves are only running the intended software. After establishing trust into these host systems, our orchestration system can deploy trusted cryptographically secured identities for virtualized components that benefit from the trust the orchestration system provides throughout the network.

With the physical network automatically, securely, bootstrapped and virtualized components automatically equipped with secure, trusted identities, a next generation industrial automation system is ready to be orchestrated to the users requirements and capable to deliver the required adaptable, modular, and secure automation solutions of the future.

Author contributions: All authors have accepted responsibility for the entire content of this submitted manuscript and approved submission.
Competing interests: The authors declare no conflicts of interest regarding this article.
Research funding: None declared.

References

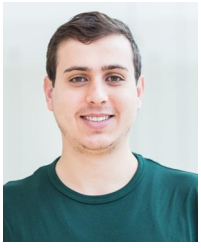
[1] ABB Process Automation, *White Paper: The DCS of Tomorrow*, 2022.
[2] The Open Group Open Process Automation Forum Business Working Group, *The Open Process Automation Business Guide*, 2021.
[3] S. Samtani, S. Yu, H. Zhu, M. Patton, and H. Chen, "Identifying scada vulnerabilities using passive and active vulnerability assessment techniques," in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, 2016, pp. 25—30.
[4] J. Schneider, S. Obermeier, and R. Schlegel, "Cyber security maintenance for scada systems," in *3rd International Symposium for ICS and SCADA Cyber Security Research 2015 (ICS-CSR)*, 2015, pp. 89—94.
[5] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," in *Tech. Rep.*, National Institute of Standards and Technology, 2020.
[6] "IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery," in *IEEE Std 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009)*, 2016, pp. 1—146.
[7] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, *Network Configuration Protocol (netconf)*, RFC 6241, RFC Editor, 2011. Available at: http://www.rfc-editor.org/rfc/rfc6241.txt.
[8] "IEEE standard for local and metropolitan area networks — secure device identity," in *IEEE Std 802.1AR-2018 (Revision of IEEE Std 802.1AR-2009)*, 2018, pp. 1—73.
[9] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*, Sunnyvale, CA, USA, Insecure, 2009.
[10] F. Kohnhäuser, D. Meier, F. Patzer, and S. Finster, "On the security of IIoT deployments: an investigation of secure provisioning solutions for OPC UA," *IEEE Access*, vol. 9, pp. 99299—99311, 2021.
[11] W. Luo, Q. Shen, Y. Xia, and Z. Wu, "Container-IMA: a privacy-preserving integrity measurement architecture for containers," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, 2019, pp. 487—500.
[12] M. De Benedictis and A. Lioy, "Integrity verification of docker containers for a lightweight cloud environment," *Future Generat. Comput. Syst.*, vol. 97, pp. 236—246, 2019.
[13] G. Cooper, B. Behm, A. Chakraborty, et al., *FIDO Device Onboard Specification 1.1*, 2021.

[14] M. Pritikin, M. Richardson, T. Eckert, M. Behringer, and K. Watsen, *Bootstrapping Remote Secure Key Infrastructures (BRSKI)*, Internet-Draft, 2020.
[15] OPC Foundation, *OPC Unified Architecture Specification Part 21: Device Onboarding*, 2022.
[16] K. Watsen, I. Farrer, and M. Abrahamsson, *Secure Zero Touch Provisioning (SZTP)*, RFC 8572, 2019.

Bionotes

Sören Finster
ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Deutschland
soeren.finster@de.abb.com
https://orcid.org/0000-0002-4611-7146

Sören Finster received his diploma degree in computer science from University of Karlsruhe (now Karlsruhe Institute of Technology, KIT), Karlsruhe, Germany, in 2008. In 2014, he received his Ph.D. degree in computer science from KIT. From 2015 to 2018 he worked as security architect and lead of the embedded security development team at Wibu-Systems AG, Karlsruhe, Germany. Since 2018, has been with ABB corporate research in Ladenburg, Germany. His research interests include security for resource constrained devices, privacy-aware protocols, and security for industrial automation systems.

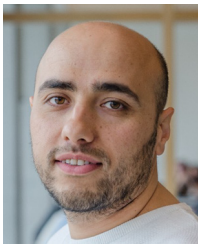


Abdallah Dawoud
ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Deutschland
abdallah.dawoud@de.abb.com

Abdallah Dawoud is a scientist with ABB Corporate Research, working on secure connected systems for industrial automation. He received his BSc in Computer Engineering from IUG, Palestine and his MSc in Computer Science from Saarland University, Germany. He also pursues a Ph.D. degree in Trusted Systems Group at CISPA Helmholtz Center for Information Security, Saarland University, Germany. Since 2022, has been with ABB corporate research in Ladenburg, Germany.

Florian Kohnhäuser
ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Deutschland
florian.kohnhaeuser@de.abb.com
https://orcid.org/0000-0002-0084-2246

Florian Kohnhäuser received his B.Sc. degree in computer science from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2012, his M.Sc. degree in IT security from the Technical University of Darmstadt (TU Darmstadt), Darmstadt, Germany, in 2014, and his Ph.D. degree in computer science also from the TU Darmstadt in 2019. Since 2019, has been with corporate research in Ladenburg, Germany. His research focuses on secure protocols, systems, and applications for industrial automation systems.



Abdulkadir Karaagac
ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Deutschland
abdulkadir.karaagac@de.abb.com
https://orcid.org/0000-0003-4924-640X

Abdulkadir Karaagac is a senior scientist with ABB Corporate Research, working on communication and interoperation solutions for industrial automation systems. Karaagac holds a Ph.D. in computer science from Ghent University in Belgium, and he has been with ABB since 2020.