Multi-leveled and ISA/IEC 62443-aware Certificate Transparency to Protect the PKI Service Supply Chain of Operational Technology

Adrian Reuter
adrian.reuter@aisec.fraunhofer.de
Technical University of Munich
Fraunhofer AISEC
Garching bei München, Germany

Michael P. Heinl
michael.heinl@hm.edu
Technical University of Munich
Munich University of Applied Sciences HM
Munich, Germany

Maximilian Pursche
maximilian.pursche@tum.de
Technical University of Munich
Fraunhofer AISEC
Garching bei München, Germany

Abstract

To address the expanding attack surface caused by increasing digitization and interconnection, operators of Industrial Automation and Control Systems (IACS) adopt security measures already established in information technology, such as Public Key Infrastructure (PKI), to Operational Technology (OT). However, operating a PKI proves to be challenging in complex and heterogeneous IACS landscapes. Hence, operators might rely on external PKI service providers, resulting in new trust dependencies and a loss of direct control over critical security components. In the WebPKI, Certificate Transparency (CT) is leveraged to monitor the certificate issuance of publicly trusted certificate authorities. Since CT's original WebPKI-centric design and trust assumptions do not align with the isolated and constrained nature of IACS environments, we investigate the adaptation of CT to a private IACS-specific PKI infrastructure operated by a service provider. We propose amendments to CT processes and roles, an IACS operator-controlled CT infrastructure, and a layered approach to align with ISA/IEC 62443. Despite the lack of CT support by crypto libraries intended for OT devices, we demonstrate the feasibility of our approach by a proof-of-concept implementation.

Keywords: Industrial Security, ISA/IEC 62443, IACS, OT, PKI-as-a-Service, Service Supply Chain Security

1 Introduction

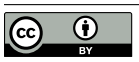
Over the last decade, manufacturing industry and utility infrastructure got more and more digitized. The convergence of production and business processes as well as the increased interconnection between machines mark the most prominent characteristics of the fourth industrial revolution. This has given rise to a novel form of value creation, encompassing made-to-order production, precise demand-based control, and enhanced visibility into the manufacturing process. However, this surge in connectivity has also exposed Operational Technology (OT) infrastructure to novel security risks that were previously unconsidered when most of the infrastructure was isolated. Common Information Technology (IT) threats now have the potential to directly impact OT environments, with substantial real-world consequences, including manufacturing delays, environmental damage, and danger to human safety.

In the light of these risks, the importance of cybersecurity concepts for OT has increased and prompted the International Society of Automation (ISA) and the International Electrotechnical Commission (IEC) to jointly develop a series of standards intended to provide guidance regarding the security of Industrial Automation and Control Systems (IACS). The ISA/IEC 62443 series of standards [14], which introduced the term IACS, was coined by the process industry. Since 2021, however, ISA/IEC 62443 is recognized as a horizontal standard [17] and strives to address diverse OT domains, such as aviation, marine, energy, and rail. We use IACS as an exemplary OT domain, but the presented concepts are applicable to other OT domains as well.

ISA/IEC 62443 introduces core concepts such as adversary-model driven Security Levels (SLs) ranging from 0 (≙ no threat anticipated) to 4 (≙ nation-state adversary) which can be mapped to zones, i.e., divisions of the overall system with similar functionality and security demands. To achieve a specific SL, certain Component Requirements (CRs) as well as System Requirement (SR) have to be fulfilled. These CRs and SRs encompass a wide range of IT security mechanisms and their adaptation to IACS environments, e.g., authentication via certificates, enabled by a Public Key Infrastructure (PKI).

1.1 Problem Statement

While PKIs facilitate new security functionalities in varying use cases, their setup and secure operation is complex and costly. This complexity has prompted IACS manufacturers, integrators, and operators to commission service providers with operating their PKI infrastructure. However, outsourcing the PKI and particularly the root Certificate Authority (CA), externalizes the trust leveraged by all certificate-based



This work is licensed under a Creative Commons 4.0 International License.

security services to a third party. Self-evidently, the trustworthiness of these service providers has to be carefully scrutinized. But even if they are considered trustworthy, there is still a risk that they are compromised by advanced adversaries, potentially even able to hide their traces by tampering with regular logs recording certificate issuance.

To establish an additional layer of defense and to equip manufacturers, integrators, and operators with the ability to actively control these scenarios instead of just passively trusting PKI service providers, we propose to adopt and adapt a security mechanism well established in the WebPKI, namely Certificate Transparency (CT), to serve OT-focused PKI use cases.

1.2 Certificate Transparency

In the WebPKI, i.e., the entirety of root certificates contained in the trust stores of the major web browsers and corresponding subordinate certificates, a multitude of different CAs are accorded the same level of trust. This results in relying parties being unable to distinguish between more and less trustworthy CA operators [9]. The issue became prominent in 2011 when DigiNotar [31], a commercial Transport Layer Security (TLS) certificate provider, was compromised by state-sponsored adversaries that used the CA to issue illegitimate but valid certificates for well-known websites such as Google and Skype. To address this security concern, CT was introduced in 2013, with the aim of enhancing transparency and trust in the certificate issuance process. It encourages CAs to log all of their issued certificates into publicly available logs, enabling comprehensive monitoring and auditing by all interested stakeholders, e.g., domain owners, to detect misissuance or indications of CA compromise. Moreover, CT can also reveal CAs that do not adhere to commonly accepted requirements. Following a 2017 security incident in which Symantec was found to have issued valid certificates for prominent websites without explicit consent, Google Chrome began to enforce CT in 2018 [23] by requiring certificates to contain a Signed Certificate Timestamp (SCT), i.e., a CT Log's cryptographic promise to record the corresponding certificate. Meanwhile, CT has been widely adopted and has become a mandatory feature of many browsers.

However, despite the substantial adoption rate in the WebPKI, CT has not found its way into the industrial PKI sector. One important factor might be that CT is conceptualized and standardized particularly for the WebPKI.

1.3 Contributions

In this paper, we examine the challenges of leveraging CT for IACS environments and present a practical solution to overcome them. In particular, our contributions are:

- We develop an IACS-specific, yet widely RFC-conform CT architecture based on adapted trust assumptions and newly defined CT roles.
- We evaluate CT's suitability for potential integration into the ISA/IEC 62443 series of standards.
- We assess CT support by open-source software components targeting IACS.
- We describe a proof-of-concept implementation, mainly based on open-source components.

1.4 Research Questions

Our contributions are guided by the following Research Questions (RQs):

RQ1 How does the trust model for private, IACS-specific CT deployments look like, especially in comparison to the trust model of CT for the public WebPKI?

RQ2 How must CT be adapted conceptually to fit the trust model of IACS while respecting their specific constraints?

RQ3 Can CT comply with and integrate into ISA/IEC 62443?

RQ4 How can CT's key components be practically integrated into an IACS using open-source OT software?

2 Related Work

To the best of our knowledge, there is no previous research on adopting CT for industrial security. Since its first standardization in RFC 6962 [21] in 2013, recent research on CT focused on feature expansions and protocol improvements. Ryan [28] proposed to include certificate revocation in CT and built an encrypted mail system on top of it. Chuat et al. [3] designed a gossip protocol that should prevent logs from offering split views to clients. Amann et al. [1] conducted a scan of the internet focusing on the adoption of CT shortly before Google announced to enforce CT [7] within their Chrome browser. They found a considerable adoption rate, especially among the most popular internet websites. A year later, Gasser et al. [6] examined the CT logs for hit list generation and the security properties of deployed certificates, finding a significant number of certificates that did not adhere to the CA/Browser Forum's baseline requirements.

An unsolved problem that continues to attract research is the disclosure of subdomains by certificates in publicly accessible CT logs, which enables malicious actors to identify targets. Scheitle et al. [29] were the first to deploy a honeypot with tailored certificates uploaded to CT in order to observe bot nets using CT as a reconnaissance method. Pletinckx et al. [24] and Kondracki et al. [19] expanded on that research by classifying the bot nets using CT as well as extracting more security-relevant data from the certificates, such as life cycle information. Roberts and Levin [27] even found subdomain names within certificates to contain sensitive data such as employee names, email addresses, and unreleased product names. Multiple research groups addressed this issue and sought to find solutions. Eskandarian et al. [4], for instance, proposed a method of redacting the relevant information from the certificates while still providing a useful SCT via

cryptographic commitments that can be verified only if the client knows the correct subdomain. Building on this idea, Sun et al. [30] proposed a semi-transparent mechanism that enables domain owners to monitor the CT logs for their domains but makes the subdomain information opaque to snoopers.

Eskandarian et al. [4], Kwon et al. [20], and Kales et al. [18] investigated possible solutions to the CT clients' problem of leaking information about a desired connection when querying for an inclusion proof. Meiklejohn et al. [22] provided a comprehensive overview of these and other proposed solutions to enhance CT. Huang et al. [12] and Wan et al. [32] addressed resource-constrained devices by examining the usage of CT for the lightweight Elliptic Curve Qu-Vanstone (ECQV) certificates. Due to the changed certificate structure, ECQV certificates are, however, not compatible with the X.509 standard and thus also not compatible with the current CT standardization. Heinl and Embacher [8] developed a Binary Transparency (BT) mechanism specifically targeted for OT software on top of the Sigstore ecosystem and proposed the potential integration into ISA/IEC 62443. Although they also leverage CT in the context of OT, their usage of CT is of rather auxiliary nature in the form of logging short-lived certificates used to compute the software signatures which are then stored in the actual BT log.

3 Threat Model, Assumptions, and Constraints

The threat model that the solution developed in this paper shall defend against comprises the following threats:

- T1 compromised and
- T2 untrustworthy CAs operated by a third-party PKI service provider.

In both scenarios, adversaries can illegitimately exploit their privileges to issue rogue certificates. These rogue certificates, in turn, can be used for a variety of attacks including but not limited to spoofing device identities, conducting man-in-the-middle attacks, and signing manipulated firmware. We address these threats by leveraging CT to provide transparency and non-repudiation with the goal of detecting the issuance of such rogue certificates. Our approach builds on the assumptions that

- A1 attackers cannot break state-of-the-art cryptography;
- A2 CT components, especially the CT Log, are controlled by the IACS operator and therefore trustworthy;
- A3 CT components, especially the CT Log, are located in a different security zone than the CA and therefore not considered compromised even if the CA is compromised.
- A4 the CT log is not public in order to
  - a. enable the usage of private PKIs since public CT logs are only meant to log certificates from CAs which are transitively signed by public root CAs [21] and
  - b. fulfill Constraint C1.

Moreover, a solution leveraging CT for OT environments needs to consider common OT constraints, including

- C1 digital certificates can be leveraged for a variety of different use cases (beyond CT's original use case, namely TLS certificates), and thus can potentially contain
  - a. sensitive information regarding customers, serial numbers, new products, internal network architecture, or even personnel.
  - b. non-standard certificate fields that may introduce difficulties, e.g., during parsing.
- C2 systems with high availability and functional reliability requirements;
- C3 heterogeneous types of devices, ranging from high-performance servers to resource-constrained, low-performance micro controllers; and
- C4 legacy or proprietary devices that do not support CT and are not readily extendable.

Therefore, a modular and leveled approach is required to adapt to heterogeneous device platforms.

4 Adapting the CT Architecture to IACS

In this section, we explore the central research question RQ2 of this paper: the adaptation of the CT infrastructure taking into account the threat model and IACS-specific constraints discussed in the previous section. We examine the different stakeholders, the trust relations between components, and the CT processes required to effectively establish transparency and non-repudiation.

As visualized in Figure 1, we generalize an IACS model as well as the relationship between the PKI service provider and the IACS operator. The IACS operator and the PKI service provider both take part in the PKI enrollment. The service provider operates the CA and most of the issuance and revocation processes. On the side of the IACS operator, the certificates issued to OT devices are often managed by manufacturer-provided engineering platforms such as TIA Portal[1] or Codesys[2]. We generalize these platforms by introducing the new role of the *OT PKI Manager* as part of our IACS-specific CT architecture. As use cases for certificates within OT can vary significantly, this paper will not delve into the details of protocols or mechanisms such as secure boot on OT devices. The objective is rather to identify a suitable integration of CT into the general relationship between PKI service provider and IACS, thereby ensuring the optimal effectiveness of CT across diverse use cases and

[1] https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html
[2] https://www.codesys.com/

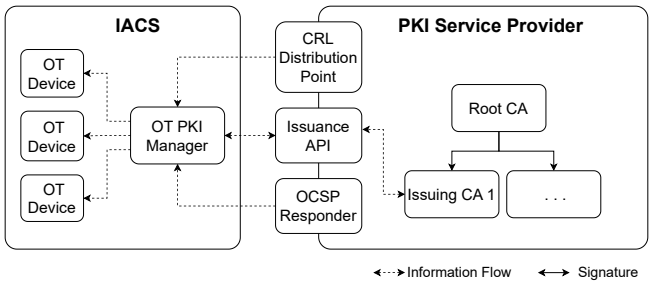


Figure 1. Generalized IACS PKI relationship with exemplary components.

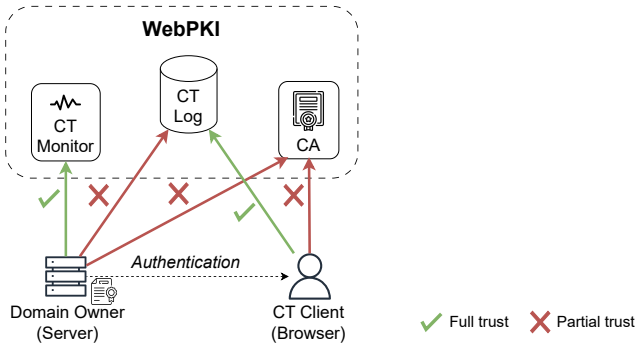


Figure 2. Trust relations in the WebPKI-CT ecosystem.

identifying any security implications that may arise from this integration.

4.1 Analysis of Stakeholder Trust Relations

In the WebPKI, CT typically involves five key stakeholders with distinct trust relationships. Due to the public nature of the WebPKI and the various attack vectors present in TLS authentication via certificates, CT is designed to mitigate multiple of the risks posed by malicious actors. The main stakeholders in a standard TLS server authentication scenario utilizing CT are: 1. CA, 2. CT Log, 3. CT Monitor, 4. domain owner, and 5. CT Client.

Figure 2 illustrates the trust relationships between the domain owner, CT Client, CA, and CT infrastructure. The domain owner does not inherently trust CAs, including the CA they contract to issue their own certificates. Historically, Google developed CT because, as a domain owner, they wanted protection against fraudulent CAs issuing unauthorized certificates. To achieve this, certificate issuance must be recorded in a CT Log, which internally uses a Merkle Tree data structure to cryptographically concatenate the log entries. However, the domain owner also does not fully trust the CT Log without additional validation. A CT Log could conspire with a CA by signing an SCT without incorporating the certificate into its Merkle Tree. Thus, a valid SCT alone does not guarantee honest CA behavior. While this attack may seem impractical, it is worth noting that many log operators are also operating large CAs at the same time. Even if they do not collude with other companies, it could be possible for a malicious or compromised CA to generate an SCT with their own CT infrastructure. To prevent such attacks, the domain owner must periodically verify the consistency of the CT Log. Due to the large size of WebPKI CT logs and the cryptographic operations required for consistency proof verification, domain owners typically delegate this task to a dedicated CT Monitor.

CT Monitors not only verify log consistency, but also notify domain owners when certificates matching their domain appear in the log. These monitors are usually commercial service providers contracted by domain owners and operate independently from CA or CT providers. This forms the first and initial positive trust relation for the domain owner. Since the CT Monitor checks the CT Log for honest behavior, which in turn makes the issuance of the CA transparent, the domain owner gains trust in the whole ecosystem by detecting misbehaving actors. Additionally, a CT Log attempting to conduct a split-view attack against a CT Monitor is impractical, since the monitor serves multiple domain owners, making it difficult for the log to predict which CT Monitor a domain owner relies on.

For the CT Client, the trust relations are more straightforward. During the TLS handshake, the client receives the server certificate and validates its chain up to the root CA. If the root certificate is found in the client's trust store, the CT Client proceeds with the verification of the SCT. The SCT is obtained either from the X.509 or TLS extension and verified using the CT Log's public key that the client obtained from a trusted source. In the WebPKI, this source is typically a browser vendor that maintains a list of trusted CT logs and serves it via an Application Programming Interface (API) endpoint. An SCT will be valid if the corresponding CT Log is marked as usable on the CT Log list and the SCT's signature validation succeeds. Consequently, CT Clients implicitly trust all CT logs approved by their maintainer. RFC 6962 [21] suggests that CT clients should audit the SCT by requesting an inclusion proof from the CT Log to guarantee certificate inclusion in the Merkle Tree. However, direct inclusion queries from the CT client to the CT Log would disclose the client's browsing history. For this reason, all web browsers to this date only use some variant of opt-in SCT auditing [22]. Only if the SCT verification succeeds, the CT client fully trusts the CA and can authenticate the TLS server.

Considering Assumption A4 and Constraint C1, there is a need for a private CT Log. Due to the fact that the purpose of the CT Log is to control a potentially compromised or untrustworthy third-party PKI service provider, it is required that the CT Log is under the sole control of the IACS operator. Otherwise, the same threat model would apply to the CT Log itself, only shifting the problem of trust from

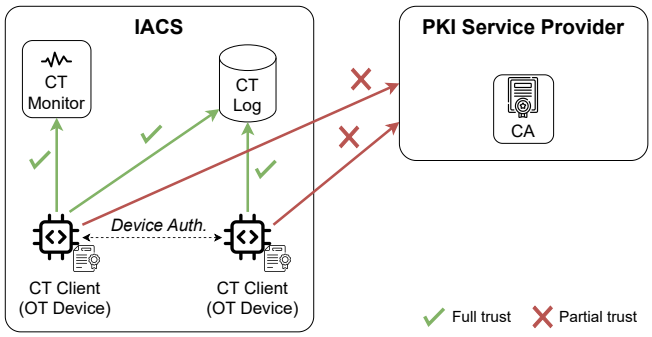


Figure 3. Trust relations in the IACS-CT ecosystem.

the PKI service provider to the CT Log operator. The same applies to the CT Monitor. By hosting the CT infrastructure components within their own domain, the IACS operator effectively replaces trust by control. As a consequence, trust relationships are significantly simpler because most CT infrastructure components are hosted and operated by the IACS operator.

Figure 3 visualizes the adapted CT trust relations within the IACS environment. Regarding the stakeholders, the IACS operator plays the role of the domain owner while also hosting the CT Log and a CT Monitor. The OT devices within the IACS take the role of CT Clients. The primary goal of deploying CT in the IACS ecosystem is to monitor the PKI service provider, ensuring that all involved parties do not fully trust the CA without additional verification.

The key difference compared to the WebPKI is the reduced complexity of trust relationships. There is only a single CT Log that needs to be trusted and one CT Monitor scanning newly logged certificates. Since the CT Log is operated by the IACS operator themselves, inclusion proofs are not necessary in order to trust the issued SCTs. The CT Client can assume that the CT Log operates honestly and includes all certificates in its Merkle Tree. However, cryptographic verification of SCTs remains essential to prevent CAs from forging signatures that resemble valid SCTs without actually logging the certificates. The centralized control of the CT infrastructure also reduces the responsibilities of the CT Monitor. Its primary function in the IACS environment is to detect unexpected certificate issuance by scanning newly logged certificates. Since the IACS operator also serves as the CT Log operator, consistency proofs are unnecessary to establish trust in the CT Log's integrity.

4.2 Adaptation of CT Components and Processes

Based on the analyzed trust relations, we present the adapted responsibilities and operational aspects of CT components within IACS as well as their mapping to the IACS environment.

4.2.1 CT Log. At the core of the CT infrastructure is the CT Log, supported by its underlying database. Its tasks and features in the IACS environment remain largely unchanged compared to the WebPKI. The CT Log's standardized API endpoints [21] must be available both externally to the PKI service provider and internally within the IACS environment, which increases the log's attack surface. As depicted in Figure 4, the endpoints allow the CA to upload certificates and enable the CT Monitor to retrieve logged certificates. The CT Log's API can be either directly exposed to the CA or proxied through another server. Without the CT Log, the CA of the service provider cannot issue certificates with valid SCTs, so prolonged downtime should be avoided. However, the uptime constraints are less critical in comparison to the WebPKI, since there are fewer certificates logged and proofs queried.

While consistency proofs are not required to establish trust in the CT Log within the IACS (cf. Section 4.1), they remain essential for operational monitoring in order to verify that no errors are present and that the Merkle Tree correctly spans all logged certificates. If the CT Log fails to incorporate submitted certificates correctly within the Maximum Merge Delay (MMD), it fails to fulfill its role within the CT ecosystem.

The CT Log's key pair is responsible for signing SCTs and serves as the second trust anchor, besides the certificate trust store, that CT Clients base their authentication processes on. The CT Log should only accept certificates from a minimal set of root CAs, i.e., only the root CAs of the service provider, to reduce the attack surface for unauthorized certificate issuance. If feasible, the service provider should also limit the number of CAs issuing certificates for the IACS to keep the system manageable and to reduce complexity. Similar to the loss of key material for a root CA, losing control of the CT Log's key pair would invalidate all SCTs signed with it. Changes to the CT Log's configuration and key material should therefore only be performed by authorized personnel following the four-eyes principle and adhering to established root CA security guidelines, such as those defined by the CA/Browser Forum, ideally adapted for the IACS environment [10].

4.2.2 The PKI Service Provider. For the PKI service provider, interacting with the private IACS CT Log is identical to interacting with any other CT Log in the WebPKI. The same API endpoints are exposed to the CA, and the responses follow standardized formats. Similar to the WebPKI, the CA first issues and submits pre-certificates before signing the final certificate that includes the returned SCT as an X.509 extension. While CT policies in the WebPKI require multiple SCTs from different CT Log operators to prevent collusion between the CA and a single CT Log, the IACS use case offers more flexibility in SCT requirements. In many IACS scenarios, a single SCT will be sufficient due to the internal

operation of the CT infrastructure and the limited number of participating entities. However, due to the diversity and segmentation commonly found in IACS environments, there may be deployments that operate multiple CT logs and require high security certificates to have multiple SCTs for higher assurance. In the context of this paper, the CT integration is scoped to require only a single SCT per certificate, reflecting a minimal viable deployment.

4.2.3 CT Monitor. The CT Monitor is the key component that detects unauthorized certificate issuance by the PKI service provider. In IACS environments, the CT Monitor has reduced responsibilities, as it does not need to verify CT Log consistency, but only scans for newly logged certificates. Thanks to the fact that the IACS operator manages the CT Log, the risk of a split-view attack in which a log colludes with the CA to deceive a CT Client is eliminated.

Since the CA of the PKI service provider can obtain a valid SCT for any certificate it submits to the log, the potential threat of unauthorized issuance of certificates that were not requested by the IACS operator must be mitigated. In an IACS environment, the number and type of certificates are highly predictable, since most of them are long-term certificates issued for pre-defined use cases. This predictability allows the CT Monitor to use an allowlist approach, matching all expected certificates. To do so, the CT Monitor continuously scans the CT Log for newly logged certificates, checking for unexpected attributes. These attributes may include any number of X.509 fields or extensions that are utilized in the IACS environment, e.g., serial numbers in the Common Name (CN) [13] or IP addresses in the Subject Alternative Name (SAN). Monitoring the key usage extensions and certificate validity periods can provide additional insight into the PKI ecosystem and detect unauthorized certificates.

While allowlist-based monitoring effectively detects misissued certificates with unexpected attributes, it does not detect misissued certificates that have the same attributes as already existing ones. It is much harder for the CT Monitor to detect unauthorized duplicate certificates, since there are legitimate reasons for duplicate CNs or SANs to exist, mainly because the certificate expired and was renewed. Additionally, IACS components may have multiple certificates for different applications. For example, a device may have one certificate for secure machine-to-machine communication and another one for an administrative web frontend. Distinguishing these benign duplicates from unauthorized ones is difficult and can only be partially automated. The CT Monitor may perform some automated checks, such as verifying whether duplicate certificates are valid simultaneously, but manual supervision will still be required.

To react to unexpected attributes, the CT Monitor must be interfaced to an alerting system that enables swift reactions and requires human acknowledgment. Any certificate that appears in the CT Log, but is not on the allowlist, is an indicator of CA misbehavior, since the PKI service provider should only issue certificates requested by the IACS operator. Therefore, the maintenance of such an allowlist must be tightly integrated into the certificate issuance process.

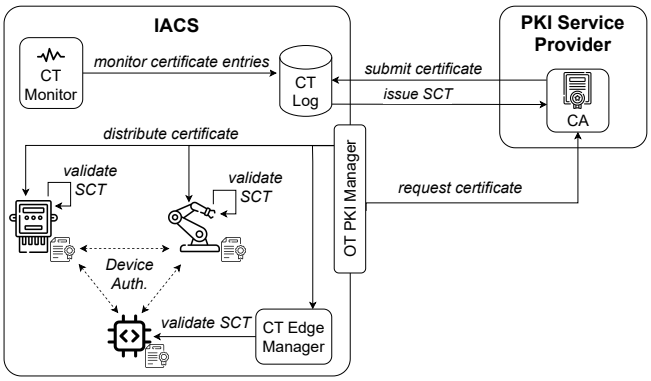


Figure 4. Overview of the CT ecosystem mapped to IACS

4.2.4 CT Client. Handling the CT Client in operation is more complex and depends on the device capabilities and the specific certificate authentication use case. However, all CT Clients share a fundamental requirement: the ability to validate an SCT using the CT Log's signature. Therefore, the CT Log's public key must be distributed to all CT Clients in advance. Because the public key serves as the trust anchor for the CT Client, they must be managed applying the same security measures as a PKI trust store update. If an attacker manipulates the CT Log's public key, they could generate fraudulent SCTs, nullifying all security benefits of CT.

In general, all CT Clients should validate the SCT signature during certificate verification. In human-to-machine and machine-to-machine communication, a missing or invalid SCT must result in certificate rejection. For other use cases, such as trusted execution environments, secure boot, or code signing, a single validation during the initial installation may suffice, provided that the certificate remains unchanged. OT devices that cannot generate their own key material or cannot obtain certificates via provisioning protocols like Enrollment over Secure Transport (EST), Simple Certificate Enrolment Protocol (SCEP), or Automatic Certificate Management Environment (ACME) are usually bootstrapped through the external OT PKI Manager.

4.3 New CT Components and Processes

In addition to adapting existing CT components to IACS environments, we propose new conceptual roles for CT that particularly address the constraints of OT environments and overcome the shortcomings of traditional CT deployments.

4.3.1 CT Edge Manager. According to an empirical evaluation of CT support among common OT software that we conducted and present in Section 6, it is likely that CT is currently not widely supported in OT environments. In cases

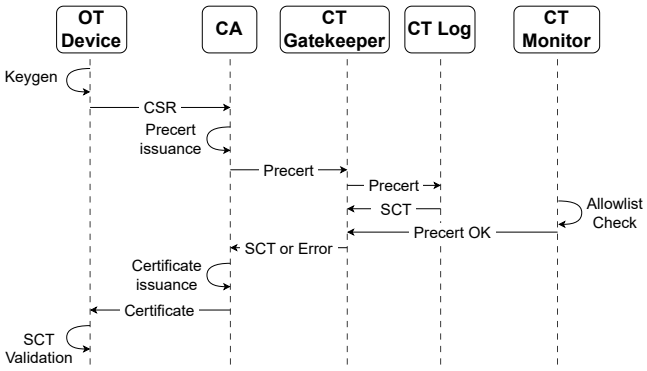


Figure 5. Step-wise illustration of a certificate issuance process within the IACS CT ecosystem.

where OT devices cannot validate SCTs on their own, e.g., due to proprietary or legacy software (Constraint C4), a supplementary CT Edge Manager can therefore do so on their behalf before installing certificates onto these devices.

Such an externalization of SCT validation can be especially useful for certificates used in self-contained use cases with little to no certificate fluctuation, such as certificates used in the secure boot process or signed code validation. However, dependent devices will lack the capability to verify SCTs of fluctuating certificates that are dynamically presented to them, such as in TLS communication or maintenance authentication scenarios. Externalizing the SCT validation removes the CT Client role from the OT device, weakening the security benefits that CT provides. Many use cases, such as secure communication and operator access control, could no longer be effectively protected by CT in such a scenario. Figure 4 visually combines the CT components with the task they are assigned to.

4.3.2 CT Gatekeeper. To turn the sole detectability of unauthorized certificates into the prevention of such, we propose the new role of the CT Gatekeeper. Acting as a proxy between the CT Log and the CA, it forwards messages in both directions. Waiting for the result of the allowlist check by the CT Monitor, the CT Gatekeeper can prevent the issuance of unauthorized certificates by blocking the SCT from reaching the CA unless the CT Monitor approves it. This approach ensures that the CT Log records pre-certificates as usual, providing accountability of the CA. However, since an issued SCT for an unauthorized pre-certificate never reaches the CA, the final certificate cannot be issued. On the one hand, this improves the overall security of the PKI ecosystem by utilizing CT as a solution not only for detection but also for prevention of fraudulent CA behavior. On the other hand, this approach also makes the CT Monitor a critical dependency for the entire certificate issuance process.

4.3.3 The Certificate Issuance Process in IACS. The sequence of the certificate issuance process is presented in Figure 5. The process begins with the generation of key material, which may occur either on the end device or through the OT PKI Manager. Many microcontrollers and OT devices used in industrial automation, such as PLCs, are equipped with libraries that can handle certificates, but require an engineering platform for bootstrapping key material. In the next step, the key material is used to generate a Certificate Signing Request (CSR) [2], binding the key to identification data. The CSR is then sent to the CA of the PKI service provider for signing. At this point at the latest, the CT Monitor's allowlist must be updated out-of-band to include the requested certificate. The CA first signs a pre-certificate containing the CSR information and submits it to the CT Log along with the complete certificate chain. Upon receipt, the CT Log validates the chain, signs the To-Be-Signed (TBS)-certificate, and returns the SCT to the CA. At this stage, the CT Monitor inspects the newly logged certificate, comparing its attributes against the allowlist and triggering alerts if unexpected values are detected.

Regardless of whether a CT Gatekeeper is used, once the CA receives the SCT, it issues the final certificate and delivers it to the requester. Depending on the CT Client's capabilities, either the end entity or the CT Edge Manager should validate the SCT before using the certificate. This ensures that the certificate was indeed included in the CT Log and can be considered legitimate.

5 ISA/IEC 62443 Integration

This section addresses RQ3 by evaluating existing requirements of ISA/IEC 62443 relevant to CT integration. Subsequently, we propose a set of layered levels for CT deployment, which relates SLs to the CT ecosystem defined in Section 4, as a basis for new ISA/IEC 62443 requirements.

5.1 ISA/IEC 62443 Requirement Analysis

CT is only applicable to an IACS if a PKI is deployed for certificate-based authentication. PKI deployment itself is governed by a set of ISA/IEC 62443's SRs [15] and CRs [16]. These have been examined in detail by Heinl et al. [10] in the context of ISA/IEC 62443-compliant PKI implementations. Several SRs and CRs are indirectly relevant for CT integration and should be considered before introducing CT-specific requirements. The primary PKI-related requirements are SR 1.8 and CR 1.8 *public key infrastructure certificates* in conjunction with SR 1.9 and CR 1.9 *strength of public key authentication*. These requirements are applied to the PKI infrastructure and to the IACS components relying on it.

SR 1.8 specifies that any PKI employed within the IACS must conform to best practices and be governed by a well-defined certificate policy. Complementary, CR 1.8 mandates that components using certificate-based authentication integrate into the IACS PKI according to standardized practices, such as those outlined in RFC 3647 [5]. These requirements

are general and primarily serve to introduce PKI as a security measure, with relevant security levels starting at SL 2. For CT integration, this implies that CT infrastructure deployment must occur at higher SLs. This is consistent with the threat model discussed in Section 3, which assumes an advanced attacker capable of obtaining valid certificates from the PKI service provider.

SR 1.9 and CR 1.9 define technical requirements for certificate-based authentication. These include certificate signature validation, verification of the certificate authority path to the root, revocation status checking, private key protection, and proper identity mapping. Additionally, they mandate the use of secure channels for the distribution of authentication material and minimization of the trust anchor set. Both requirements include a Requirement Enhancement (RE) that introduces the use of hardware-based security mechanisms to protect private keys. Achieving SL 3 and above necessitates implementing such technologies such as Trusted Platform Modules. CT adds further steps to the certificate validation process. Specifically, presented SCTs must be successfully validated before a certificate can be considered trustworthy. This involves reconstructing the TBS certificate and validating the SCT signature using the CT Log's public key. As it fulfills a similar purpose, the CT Log public key must be treated with the same level of scrutiny as a root CA certificate.

In addition to SRs and CRs, ISA/IEC 62443 defines requirements for specific device categories: Host Device Requirements (HDRs), Embedded Device Requirements (EDRs), and Network Device Requirements (NDRs). From these categories, two sets of requirements are especially relevant for CT integration. HDR 3.12, EDR 3.12, and NDR 3.12 *provisioning product supplier roots of trust* specify that devices must provide a way for the manufacturer to supply a trust anchor. The standard does not prescribe the specific nature of the trust anchor but requires that its integrity, authenticity, and confidentiality be protected starting at SL 2. For CT, this implies that newly manufactured devices with CT Client functionality must be able to be provisioned with a CT Log public key during manufacturing. Proper protection of this trust anchor is essential, especially if device ownership is transferred.

For CT functionality within the IACS, HDR 3.13, EDR 3.13, and NDR 3.13 *provisioning asset owner roots of trust* apply. These requirements mandate that asset owners, in this context typically IACS operators, must be able to provision their own trust anchors. The provisioned trust anchors must be protected to the same standard as those provided by the manufacturer. Provisioning must be possible without reliance on external components located outside the device's security zone. For the discussed CT Client adoption, this highlights the importance of securely provisioning the CT Log public key. Devices incapable of this provisioning requirement cannot take part in the CT ecosystem directly but must rely on an CT Edge Manager that validates the SCT at least once before loading the certificate onto the device.

Several SRs and CRs in ISA/IEC 62443 present inherently difficult challenges for compliant PKI deployment. One of the most notable examples is SR 5.2 *zone boundary protection*, which requires control and monitoring of inter-zone communication. In the case of an incident, the REs *Island Mode* and *Fail Close* mandate for a communication breakdown, posing a challenge for resilient PKI infrastructure [10]. The proposed CT ecosystem is not impacted by SR 5.2, as CT Clients do not need to communicate with other CT components during normal operation, excluding certificate enrollment. During enrollment, the CT Log, the CT Monitor, and - if present - the CT Gatekeeper are required to communicate with each other, which can be compliant when they are situated within the same IACS zone.

5.2 Layered CT Requirements

One of the most important factors for integrating new technology in IACS-specific standardization is flexibility. Due to the wide range of different use cases and device capabilities, a leveled approach has the best prospects of adoption. In this section, we do not directly formulate ISA/IEC 62443 SRs and CRs, but rather present a leveled, flexible approach to deploy CT in accordance with ISA/IEC 62443. However, these different levels might indeed be used as a basis for further standardization activities, e.g., by ISA and IEC's Joint Team (JT) JT-62443-3-1, that continuously analyzes applicability of formerly unregarded technology to IACS use cases. Where appropriate, exemplary SLs are suggested to relate to ISA/IEC 62443 and existing PKI requirements. For instance, since the application of SR 1.8 *public key infrastructure certificates* starts at SL 2, a CT-related SR must not introduce obligations below SL 2. As CT depends on the presence of a PKI, lower SLs are not applicable.

5.2.1 Client-side Validation Levels. Starting with the CT Client, four levels of SCT validation are presented in Table 1 and referred to as Validation Levels (VLs). VL0 is not explicitly mentioned, since it involves no SCT auditing, effectively disabling CT Client functionality. In terms of ISA/IEC 62443, this maps to SL0, where the security measure and the corresponding requirement are not applied. VL1 requires that the SCT associated with a certificate be validated at least once during deployment. This validation is delegated to the CT Edge Manager. VL2 requires that the formerly described validation is performed by the device itself. While VL1 and VL2 do not ensure continuous CT protection, they may suffice for limited, self-contained use cases such as secure boot or validation of signed code. Especially VL1 may represent a feasible intermediate solution for resource-constrained or otherwise limited devices. VL3 involves SCT validation as part of every certificate validation. This enables full CT Client functionality in the IACS and supports all

authentication scenarios, including secure channel authentication and operator login. VL4 introduces complete SCT auditing per RFC 6962 [21], including validation of inclusion proofs. As discussed previously, inclusion proofs are actually not required in private CT deployments due to our trust model which assumes a trustworthy private CT log under our sole control. However, inclusion proofs can further strengthen ISA/IEC 62443's defense-in-depth paradigm by providing an additional mechanism to detect potentially compromised CT logs.

5.2.2 Infrastructure Integration Levels. The CT infrastructure, particularly the CT Monitor, offers more integration flexibility. Our proposed levels are shown in Table 2 and are referred to as CT Infrastructure Integration Levels (ILs). These ILs are of cumulative nature, i.e., fulfilling a specific IL implies fulfilling all lower ILs as well. At IL1, the CT Monitor is not in operation. While this severely limits the protection a CT ecosystem can provide, it may be acceptable if the only desired objective is accountability of the PKI service provider, which can be achieved with a standalone CT Log. This corresponds to SL2 in terms of ISA/IEC 62443 because PKI authentication is required from SL2 onwards. At IL2, the CT Monitor scans the CT Log for unauthorized or suspicious certificates. The choice of monitored attributes is left to the operator and will vary between IACS deployments. In either case, a robust alerting system must be in place and actively reacted to. IL3 leverages the CT Gatekeeper to integrate the CT Monitor into the certificate issuance process as described in Section 4. This allows operators to withhold SCTs, preventing unauthorized certificate issuance by the CA. In terms of CT Monitor operation, this provides the highest level of security for the IACS. IL4 includes the full CT Monitor capabilities detailed in RFC 6962 [21], including consistency proof queries to the CT Log. Consistency proofs are not necessary for private CT logs corresponding to our trust model and Assumption A2, since here the CT Log is inherently trusted. However, just like inclusion proofs, consistency proofs can further strengthen ISA/IEC 62443's defense-in-depth paradigm by providing an additional mechanism to detect potentially compromised CT logs on the IACS operator side.

6 CT Support of Existing Software

To address RQ4, we first evaluate the current state of CT support of existing open-source software suitable for use in IACS environments.

6.1 CT Log

The primary open-source software suite for CT is the *certificate-transparency-go*[3] repository, successor of the former official reference implementation. It includes a fully functional CT Log implementation and tools to interact with the log's endpoints, supporting operations such as certificate submission, log entry retrieval, and consistency proof validation. The CT Log is built on top of the Trillian project[4], which implements Merkle Tree logic and handles database interaction for the transparency log.

6.2 CT Monitors

The number of open-source CT Monitors is limited. The *certificate-transparency-go* suite implements the necessary endpoints for a CT Monitor to scan a CT Log, but lacks logic for active monitoring or alerting. Probably the most distinguished open-source CT Monitor is the SSLMate Cert Spotter[5] utility. It allows the monitoring of CT logs for certificates with specified DNS names in the SAN field and supports alerting via email or custom scripts. However, with this feature set, Cert Spotter does not fully meet the IACS CT Monitor needs. Within the IACS domain, certificates that bind to IP addresses in the SAN are common as well as the device serial number being located in the CN field [13], which Cert Spotter does not support. Moreover, an allowlist-based monitoring strategy would need to apply inverse logic, identifying all certificates that lack the allowed attributes rather than detecting specific attributes.

6.3 PKI Suites

Popular open-source PKI suites such as OpenSSL CA, OpenXPKI, DogTag CA, and Step-CA do not natively support interactions with a CT Log to retrieve SCTs. While these tools permit issuing certificates with custom X.509 extensions, they do not explicitly support issuing pre-certificates or populating the SCT list extension. Interestingly, OpenSSL can parse SCTs and includes SCT validation routines in its C library[6]. The general lack of CT support likely stems from the exclusion of private PKIs from the WebPKI CT ecosystem, as only certificates issued by trusted root CAs are accepted by public CT logs. However, most open-source PKI software is primarily designed for private PKI deployments, where CT has not yet seen integration.

6.4 CT Client Support

While SCT validation is provided by the underlying libraries of the popular web browsers and the *certificate-transparency-go* suite, TLS libraries targeting OT devices and IACS applications generally lack CT support. OpenSSL implements SCT handling in a general fashion that could be used for OT use cases. However, due to its size and reliance on other standard libraries, its deployment is unsuitable for computationally constrained devices. Dedicated TLS and certificate handling libraries for OT, such as WolfSSL, MbedTLS, and

[3] https://github.com/google/certificate-transparency-go

[4] https://github.com/google/trillian

[5] https://github.com/SSLMate/certspotter

[6] https://docs.openssl.org/3.5/man3/SCT_validate/

Table 1. CT Client SCT Validation Levels

VL	Moment of SCT Validation	By	Use Case	SL
1	On Certificate Deployment	CT Edge Manager	Self-Contained (e.g., Secure Boot, Code Signing)	3
2	On Certificate Deployment	On Device	Self-Contained (e.g., Secure Boot, Code Signing)	3
3	On Certificate Usage	On Device	Communication Authentication (e.g., TLS)	4
4	Additional Inclusion Proof	Any of Both	Detect Compromise of CT Log	-

Table 2. CT Infrastructure Integration Levels (Cumulative)

IL	CT Infrastructure Integration	By	Protection Provided	SL
1	Standalone CT Log w/o Monitor	CT Log	Accountability only	2
2	Scans for Unauthorized Certificates	CT Monitor	Detection of Unauthorized Certificates	3
3	Controls Certificate Issuance	CT Gatekeeper	Prevention of Unauthorized Certificates	4
4	Certificate Scan + Consistency Proofs	CT Monitor	Detect Compromise of CT Log	-

CycloneSSL do not support SCT validation and thus no CT Client operations.

7 Proof-of-Concept Implementation

We present a proof-of-concept implementation for mutual TLS authentication using certificates with SCT extensions to assess whether CT infrastructure can be deployed within an IACS environment. Figure 6 illustrates the components of and their relations within our proof-of-concept implementation. Our setup consists of a PKI for the simulated service provider CA, a private CT Log with an associated database, a CT Monitor, and two CT Clients that perform TLS communication using SCT-backed certificates.

The CT Log is implemented using the Trillian log server, which adheres to RFC 6962 [21] and consists of three sub components fulfilling different roles in the CT Log. The Trillian log server implements a transparent append-only log using Merkle Trees and can operate with various SQL databases as its storage backend. We use the recommended[7] MariaDB.

The CT Frontend (CTFE) interfaces with the Log Server on one side and with CT ecosystem users via the standardized CT Log API on the other. The CTFE setup requires several configurations, including the generation of the CT Log key pair used to sign the Signed Tree Heads (STHs) and SCTs. For our proof of concept, an elliptic curve key pair on the NIST P-256 curve is generated. Moreover, a trust store of accepted root certificates is defined, including a self-signed root CA with 4096-bit RSA keys.

The PKI is built with the OpenSSL CA tool. The previously generated root CA certificate is used to sign an intermediate issuing CA certificate of also 4096-bit RSA key length. As previously stated, the OpenSSL CA tool does not support direct interaction with a CT Log to submit a certificate and obtain an SCT. However, it does support inserting custom X.509 extensions when issuing certificates. For this reason, we create two certificate profiles in the configuration file, one for adding the pre-certificate poison extension and another for including the SCT list extension. Provided a CSR, OpenSSL CA first issues the desired end-entity certificate with the poison extension marked as critical. This pre-certificate must then be submitted to the CT Log via its API, and the response must be parsed and re-encoded for inclusion in the SCT. The pre-certificate is bundled with its certificate chain and submitted to the CT Log. The CT Log responds with a JavaScript Object Notation (JSON) message containing the Base64-encoded Log ID, timestamp, and TLS-encoded signature over the TBS certificate. All X.509 extensions are standardized to be ASN.1 encoded [2], however, the CT RFC [21] defines TLS encoding for the SCT list extension, which results in a double encoding of the SCT data before its inclusion in the certificate itself.

We implement this procedure as a script that parses the response, formats the values into the correct encoding, and writes the result into the certificate profile. The OpenSSL CA tool can now issue a final end-entity certificate from the same CSR with the generated SCT list extension. A remaining challenge is OpenSSL CA's safeguard against issuing the same certificate multiple times. Because the pre-certificate and final certificate must have identical TBS data, the validity period and serial number must match. However, reusing a serial number violates the requirement for serial number uniqueness. OpenSSL CA enforces this by tracking issued serials and blocking duplicates. To address this, we implemented a tweak to OpenSSL CA such that only the final certificate is tracked persistently in its data base and the pre-certificate is discarded.

To demonstrate a suitable monitoring solution, we develop a minimal CT Monitor in Python. Its primary task is to poll the CT Log at regular intervals and compare new certificate entries against a pre-defined list of certificate attributes.

[7] https://github.com/google/certificate-transparency-go/blob/master/trillian/docs/ManualDeployment.md

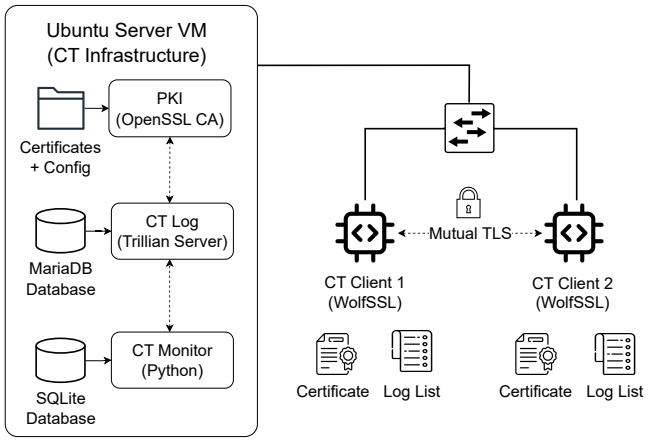


Figure 6. Proof-of-Concept Setup

Finally, we implement the CT Client proof of concept. We compile WolfSSL[8] version 5.7.6 from source and install it on two client devices previously provisioned with individual end-entity certificates. WolfSSL does not natively support SCT validation or auditing, but allows custom callbacks during the TLS handshake and certificate validation. The goal of the proof of concept is demonstrate a TLS handshake example which includes the required SCT verification routine. In this setup, the key callback is triggered during the TLS handshake after receiving the peer's certificate.

By default, the callback is only invoked if the certificate validation fails. However, compiling the library with the *WOLFSSL_ALWAYS_VERIFY_CB* flag ensures the callback is executed during every TLS handshake when a certificate is validated. Instead of re-implementing the parsing logic in the WolfSSL library, we utilize the *sctcheck* utility from the *certificate-transparency-go* repository. It validates SCT list extensions against a set of known logs provided by a custom log list. The clients are bootstrapped with a custom log list that includes only the previously deployed CT Log. If the SCT validation succeeds, the TLS handshake proceeds normally. If it fails, the certificate validation is canceled.

8 Discussion

In this section, we conclude the paper by discussing our work and mapping it to the initially defined RQs, threats, assumptions, and constraints. Eventually, we provide a scientific outlook on possible future work.

8.1 RQ1: Trust Model

To emphasize the practical relevance of our work we first want to address two critical questions. The first is: *From the perspective of an IACS operator, why not employ an internal CA instead of contracting a PKI service provider in combination with an internal CT Log?* In contrast to complex PKIs processes including deployment, renewal, and revocation, initially setting up and operating a private CT Log is rather simple. Even the lowest integration level IL1, which requires nothing more than passively logging certificate issuance, provides substantial benefits to the IACS operator by establishing the defense-in-depth paradigm and thereby ensuring accountability and evidence, e.g., to investigate in case of an incident or support claims for compensation, not least due to Assumption A3. This brings us to the second question to be addressed: *Why not using public CAs and public CT Logs?* The benefit of a public CA is that certificates are publicly trusted. However, they can only be publicly trusted if they are logged in public CT Logs which contradicts Constraint C1. Therefore, Assumption A4 mandates the usage of a private CT Log. This implies shifting from a public, distributed, multi-party trust model to a private, internalized, IACS operator-controlled CT infrastructure, which allows to focus solely on CA behavior and therefore our threat model consisting of a compromised (Threat T1) or untrustworthy third-party CA (Threat T2).

8.2 RQ2: CT Adaptation

In addition to architecture and configuration changes to existing roles, including the CT Monitor and the CT Client, we introduced new roles to realize IACS-related improvements: 1. the CT Gatekeeper to allow IACS operators to actively control the issuance process of third party-operated CAs and 2. the CT Edge Manager to enable devices without CT to benefit from CT's security benefits nevertheless. Based on the mentioned IACS-specific trust model elaborated in Section 4.1 and different from the mechanisms CT provides to the WebPKI, our proposed solution is even capable of preventing the illegitimate issuance of rogue certificates by leveraging the CT Gatekeeper. In light of the potential physical damage a rogue certificate could cause to an IACS, this capability constitutes a significant OT security extension. Another important advantage of our solution is its flexibility which allows leveraging the benefits of CT by a wide variety of devices. In general, the overhead introduced by SCT verification on end devices is manageable. No new cryptographic primitives are required and only a single additional digital signature is validated. However, devices performing time-critical tasks (Constraint C2), low-performance devices (Constraint C3), or proprietary devices (Constraint C4) may even delegate SCT validation to the CT Edge Manager and still be partially protected by CT.

8.3 RQ3: ISA/IEC 62443 Integration

In addition, we examined the ISA/IEC 62443 series of standards to evaluate CT's compatibility with corresponding requirements on PKI employment and proposed a leveled approach for CT adoption, distinguishing between efforts on CT infrastructure (ILs) and client side (VLs). This approach can help to gradually introduce CT in line with

[8] https://github.com/wolfSSL/wolfssl

Table 3. Certificate size comparison in bytes with and without embedded SCT list extension

File Format	Excluding SCT	Including SCT	Difference	% Increase
PEM	1505	1696	191	12.69%
DER	1070	1212	142	13.27%

ISA/IEC 62443's tiered security model and therefore increase IACS security, while accommodating device and environment capabilities through leveled adoption of VLs and ILs. At the same time, this leveled approach aligns well with ISA/IEC 62443's risk-based approach to efficiently allocate resources only to the implementation of controls where they are really needed according to the corresponding threat and adversary model.

8.4 RQ4: Practical Open-Source Integration

The CT Log component implemented with Trillian fully supports the IACS use case. Its flexibility and scalability, combined with RFC 6962 [21] compliance, makes it a viable candidate for long-term deployment. The CT Monitor presents more of a challenge. While basic functionality can be achieved using open-source software originally developed for the WebPKI, meaningful deployment in the IACS context requires integration with the certificate issuance process. Monitoring logic tailored to IACS environments, such as allowlist validation of IP addresses or device serial numbers, demands custom implementation. Our proof-of-concept implementation demonstrates that such customization is possible, but it does not constitute a production-ready product. To provide a higher level of protection, the CT Monitor must be directly involved in the issuance process and interact with the CT Gatekeeper as described in Section 4.3. This would enable the CT Monitor to prevent the issuance of unauthorized certificates, increasing overall security. The interactions between the CT Monitor and CT Gatekeeper are elaborated in Section 4.3.2. How to manage an allowlist without rejecting legitimate authorized certificates remains a challenge to be explored in future work.

CT Client support in OT libraries is currently the weakest point. Unlike TLS implementations for web browsers, OT-specific libraries such as WolfSSL, MbedTLS, or CycloneSSL lack native support for SCT validation. Although our proof of concept demonstrates how custom SCT parsing can be added manually, this approach is not sustainable for product-grade IACS deployments. Realistically, CT Client support needs to be implemented by providers of TLS libraries to guarantee long-term stability through regular function testing and audits. Adding CT Client support to their base crypto libraries would facilitate CT adoption across multiple applications and devices. Expecting IACS operators to develop this functionality independently is unrealistic and implies an increased risk to result in incorrect handling of cryptographic functions. Certificate size is another important consideration, particularly for embedded OT devices with limited memory. Table 3 details the size increase of the certificates in our proof-of-concept implementation with and without the SCT list extension. It shows that the size increase is around 142 bytes in DER encoded certificates, which is likely manageable even for constrained devices. While lacking native CT support, open-source PKI software presents fewer issues concerning the IACS use case. Since the CT Log interface in private deployments is identical to that of public CT logs, service providers with existing WebPKI CT integration can reuse tools with minimal changes. Most PKI providers operate public CAs and therefore must already have tools available for managing pre-certificates and embedding SCTs.

We empirically analyzed available open-source solutions, revealing limited support for IACS use cases. We therefore developed custom tooling for CT Monitor and CT Client to illustrate key functions to highlight current software gaps and the need for CT support in OT-specific libraries such as embedded TLS stacks, secure bootloaders, and crypto toolkits. Although our study showed viability, the current implementations are not ready for production use.

8.5 Future Work

Future work includes the integration of the CT Monitor into the certificate issuance process, transforming the CT Monitor from a passive observer to an active security control. Another currently unimplemented but relevant concept is SCT edge validation to support clients which lack the necessary resources to validate SCTs themselves. The ideal point for integrating this feature into the issuance or onboarding process remains an open research question. Moreover, CT standardization currently lacks a defined secure method to distribute CT Log public keys among CT Clients. In IACS, protocols such as Bootstrapping Remote Secure Key Infrastructure (BRSKI) [11, 25], which enable secure and scalable PKI deployment without human interaction, could potentially be adapted for this task. To be prepared to withstand adversaries with access to post-quantum computers, IACS-specific use of Post-Quantum Cryptography (PQC) [26] should be applied to CT.

9 Conclusion

To regain control over outsourced PKI infrastructure, this paper examines the application of CT in IACS. For this, CT is conceptually adapted to fit the differing trust model of IACS, including changes to architecture and configuration of existing roles and the definition of new roles. Unlike CT for

the WebPKI, our proposed solution can even prevent the illegitimate issuance of rogue certificates. We further examine CT's suitability for integration into the ISA/IEC 62443 series of standards and present a leveled approach for gradual adoption of CT in line with ISA's tiered security model. Our proof of concept demonstrates that implementation using open-source software for embedded devices is technically feasible without violation of OT constraints such as CPU power, memory, or connectivity.

References

[1] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. 2017. Mission accomplished? HTTPS security after diginotar. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 325–340. doi:10.1145/3131365.3131401

[2] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. doi:10.17487/RFC5280

[3] Laurent Chuat, Pawel Szalachowski, Adrian Perrig, Ben Laurie, and Eran Messeri. 2015. Efficient gossip protocols for verifying the consistency of Certificate logs. In *2015 IEEE Conference on Communications and Network Security (CNS)*. 415–423. doi:10.1109/CNS.2015.7346853

[4] Saba Eskandarian, Eran Messeri, Joseph Bonneau, and Dan Boneh. 2017. Certificate Transparency with Privacy. *Proc. Priv. Enhancing Technol.* 2017, 4 (2017), 329–344. doi:10.1515/POPETS-2017-0052

[5] Dr. Warwick S. Ford, Dr. Santosh Chokhani, Stephen S. Wu, Randy V. Sabett, and Charles (Chas) R. Merrill. 2003. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 3647. doi:10.17487/RFC3647

[6] Oliver Gasser, Benjamin Hof, Max Helm, Maciej Korczynski, Ralph Holz, and Georg Carle. 2018. In Log We Trust: Revealing Poor Security Practices with Certificate Transparency Logs and Internet Measurements. In *Passive and Active Measurement Conference 2018*.

[7] Google Chrome. 2018. *Certificate Transparency*. https://chromium.googlesource.com/chromium/src/+/master/net/docs/certificate-transparency.md

[8] Michael P. Heinl and Victor Embacher. 2024. BT2X: Multi-Leveled Binary Transparency to Protect the Software Supply Chain of Operational Technology. In *Proceedings of the Sixth Workshop on CPS-IoT Security and Privacy* (Salt Lake City, UT, USA) *(CPSIoTSec'24)*. Association for Computing Machinery, New York, NY, USA, 41–54. doi:10.1145/3690134.3694816

[9] Michael P. Heinl, Alexander Giehl, Norbert Wiedermann, Sven Plaga, and Frank Kargl. 2019. MERCAT: A Metric for the Evaluation and Reconsideration of Certificate Authority Trustworthiness. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop* (London, United Kingdom) *(CCSW'19)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3338466.3358917

[10] Michael P. Heinl, Maximilian Pursche, Nikolai Puch, Sebastian N. Peters, and Alexander Giehl. 2023. From Standard to Practice: Towards ISA/IEC 62443-Conform Public Key Infrastructures. In *Computer Safety, Reliability, and Security*, Jérémie Guiochet, Stefano Tonetta, and Friedemann Bitsch (Eds.). Springer Nature Switzerland, Cham, 196–210.

[11] Michael P. Heinl, Adrian Reuter, Sebastian N. Peters, and Markus Bever. 2025. Leveraging BRSKI to Protect the Hardware Supply Chain of Operational Technology: Opportunities and Challenges. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing* (Catania International Airport, Catania, Italy) *(SAC '25)*. Association for Computing Machinery, New York, NY, USA, 245–254. doi:10.1145/3672608.3707707

[12] Wanling Huang, Jingqiang Lin, Qiongxiao Wang, Yajun Teng, Huiqing Wan, and Wei Wang. 2021. Certificate Transparency for ECQV Implicit Certificates. In *ICC 2021 - IEEE International Conference on Communications*. 1–6. doi:10.1109/ICC42927.2021.9500380

[13] IEEE. 2018. IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity. *IEEE Std 802.1AR-2018 (Revision of IEEE Std 802.1AR-2009)* (2018), 1–73. doi:10.1109/IEEESTD.2018.8423794

[14] International Electrotechnical Commission. 2009. *Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models*. International Standard IEC 62443-1-1:2009. International Electrotechnical Commission, Geneva, CH. https://webstore.iec.ch/preview/info_iects62443-1-1%7Bed1.0%7Den.pdf

[15] International Electrotechnical Commission. 2013. *Industrial communication networks – Network and system security – Part 3-3: System security requirements and security levels*. International Standard IEC 62443-3-3:2013. International Electrotechnical Commission, Geneva, CH. https://webstore.iec.ch/preview/info_iec62443-3-3%7Bed1.0%7Db.pdf

[16] International Electrotechnical Commission. 2019. *Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components*. International Standard IEC 62443-4-2:2019. International Electrotechnical Commission, Geneva, CH. https://webstore.iec.ch/preview/info_iec62443-4-2%7Bed1.0%7Db.pdf

[17] International Electrotechnical Commission. 2021. *The International Electrotechnical Commission Designates ISA/IEC 62443 as a Horizontal Standard*. https://www.isa.org/news-press-releases/2021/november/the-international-electrotechnical-commission-desi

[18] Daniel Kales, Olamide Omolola, and Sebastian Ramacher. 2019. Revisiting User Privacy for Certificate Transparency. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. 432–447. doi:10.1109/EuroSP.2019.00039

[19] Brian Kondracki, Johnny So, and Nick Nikiforakis. 2022. Uninvited Guests: Analyzing the Identity and Behavior of Certificate Transparency Bots. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 53–70. https://www.usenix.org/conference/usenixsecurity22/presentation/kondracki

[20] Hyunsoo Kwon, Sangtae Lee, Minjae Kim, Changhee Hahn, and Junbeom Hur. 2023. Certificate Transparency With Enhanced Privacy. *IEEE Transactions on Dependable and Secure Computing* 20, 05 (Sept. 2023), 3860–3872. doi:10.1109/TDSC.2022.3214235

[21] Ben Laurie, Adam Langley, and Emilia Kasper. 2013. Certificate Transparency. RFC 6962. doi:10.17487/RFC6962

[22] Sarah Meiklejohn, Joe DeBlasio, Devon O'Brien, Chris Thompson, Kevin Yeo, and Emily Stark. 2022. SoK: SCT Auditing in Certificate Transparency. *Proc. Priv. Enhancing Technol.* 2022, 3 (2022), 336–353. doi:10.56553/POPETS-2022-0075

[23] Devon O'Brien, Ryan Sleevi, and Andrew Whalley. 2017. *Chrome's Plan to Distrust Symantec Certificates*. https://security.googleblog.com/2017/09/chromes-plan-to-distrust-symantec.html

[24] Stijn Pletinckx, Thanh-Dat Nguyen, Tobias Fiebig, Christopher Kruegel, and Giovanni Vigna. 2023. Certifiably Vulnerable: Using Certificate Transparency Logs for Target Reconnaissance. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. 817–831. doi:10.1109/EuroSP57164.2023.00053

[25] Max Pritikin, Michael Richardson, Toerless Eckert, Michael H. Behringer, and Kent Watsen. 2021. Bootstrapping Remote Secure Key Infrastructure (BRSKI). RFC 8995. doi:10.17487/RFC8995

[26] Maximilian Pursche, Nikolai Puch, Sebastian N. Peters, and Michael P. Heinl. 2024. SoK: The Engineer's Guide to Post-Quantum Cryptography for Embedded Devices. Cryptology ePrint Archive, Paper

2024/1345. https://eprint.iacr.org/2024/1345

[27] Richard Roberts and Dave Levin. 2019. When Certificate Transparency Is Too Transparent: Analyzing Information Leakage in HTTPS Domain Names. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society* (London, United Kingdom) *(WPES'19)*. Association for Computing Machinery, New York, NY, USA, 87–92. doi:10.1145/3338498.3358655

[28] Mark D. Ryan. 2013. Enhanced certificate transparency and end-to-end encrypted mail. Cryptology ePrint Archive, Paper 2013/595. https://eprint.iacr.org/2013/595

[29] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. 2018. The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18)*. Association for Computing Machinery, New York, NY, USA, 343–349. doi:10.1145/3278532.3278562

[30] Aozhuo Sun, Bingyu Li, Qiongxiao Wang, Huiqing Wan, Jingqiang Lin, and Wei Wang. 2023. Semi-CT: Certificates Transparent to Identity Owners but Opaque to Snoopers. In *2023 IEEE Symposium on Computers and Communications (ISCC)*. 1207–1213. doi:10.1109/ISCC58397.2023.10217862

[31] The Dutch Safety Board. 2012. *THE DIGINOTAR INCIDENT: Why digital safety fails to attract enough attention from public administrators.* https://www.onderzoeksraad.nl/wp-content/uploads/2023/11/rapport_diginotar_en_summary.pdf

[32] Huiqing Wan, Qiongxiao Wang, Yajun Teng, Cunqing Ma, Jingqiang Lin, and Mingyu Wang. 2023. ImCT: A Feasible Scheme for Deploying Implicit Certificates with Certificate Transparency in IoT. In *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. 1–10. doi:10.1109/ICCCN58024.2023.10230166