Journal of Information Security and Applications 59 (2021) 102848

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Verifiable and auditable authorizations for smart industries and industrial Internet-of-Things

Luca Ferretti [a],[*], Francesco Longo [b], Giovanni Merlino [b], Michele Colajanni [c], Antonio Puliafito [b], Nachiket Tapas [b]

[a] *University of Modena and Reggio Emilia, Italy*
[b] *University of Messina, Italy*
[c] *University of Bologna, Italy*

ARTICLE INFO

Keywords:
Industrial Internet of Things
Authorization
Delegation
Access control
Authentication
Transparency

ABSTRACT

Modern industrial systems are enriched by cyber–physical devices and interconnections with business processes that enable flexible production, remote monitoring, control and maintenance. These systems are typically subject to multiple authorities which must cooperate with each other, as in the case of segmented industrial environments and supply chains. In similar contexts, voluntary or unintentional damages may be caused by cyber attacks or by misbehaving authorized parties. We propose an original architecture that regulates accesses to industrial systems' resources through authorization delegation procedures. It guarantees several benefits that include the possibility of auditing authorizations released by delegated third parties, of detecting misconducts and possible attacks, and of assuring attribution of misconducts. The proposed solution is compatible with constraints characterizing industrial environments and with security and performance requirements of industrial architectures. The performance and latencies of the auditing mechanisms are evaluated through a prototype.

1. Introduction

Modern enterprises are deploying a large amount of IoT devices and interconnections among production plants and business infrastructures, building so-called Industrial Internet-of-Things (IIoT) [1]. These components can provide enterprises with advantages in terms of flexibility, productivity, management through remote monitoring and control [2], fast detection and even prediction of component failures [3,4], and continuous data exchange among supply chain partners [1,5]. However, they open new attack surfaces that increase the cyber vulnerability of industrial infrastructures. Trustworthy industrial systems must ensure safety of people and infrastructure security, operation continuity and reliability. Any violation of these requirements must be prevented or at least audited with the twofold goal of identifying the causes and attributing the fault to the entities and operations that have caused the problem [6]. We propose an original solution to the auditing issue in industrial settings that are governed by multiple authorities, where an authority A can allow an authority B to access the resources of A but only after explicit authorization. For example, let us consider an enterprise governing many manufacturing plants (Operational Technologies), the Information Technologies, human operators and external maintainers. The enterprise must grant an authorization to let other enterprises' operators to access an IT or OT resource for maintenance. Several conflicting requirements arise in a similar scenario. The delegating enterprise A would prefer to release a fine-grained and short-lived authorization to the enterprise B's maintainers in order to limit the risks related to external accesses. On the other hand, the delegated enterprise B would like to receive broader and more flexible access privileges so to decide later which maintainer involving in operations and intervention times. Existing solutions based on delegated authorization protocols [7] adopt recursive authorization strategies where each authority can release authorizations based on coarse-grained access control rules. Then, each authorized party has the capability of releasing fine-grained authorizations autonomously with the only constraint of not violating the original delegation. These approaches cannot satisfy the requirements of an industrial setting because, if an authorized authority is compromised, then it can become the vector of attacks against several critical components causing safety and operational risks [8].

[*] Corresponding author.
E-mail addresses: luca.ferretti@unimore.it (L. Ferretti), flongo@unime.it (F. Longo), gmerlino@unime.it (G. Merlino), michele.colajanni@unibo.it (M. Colajanni), apuliafito@unime.it (A. Puliafito), ntapas@unime.it (N. Tapas).

https://doi.org/10.1016/j.jisa.2021.102848

Available online 26 April 2021

We improve the security level of recursive authorization strategies by introducing a mechanism to audit authorizations released by delegated parties. Our proposal enforces security by considering covert security assumptions with public auditability [9,10]. That is, the proposed mechanisms do not prevent misbehaviors of delegated parties, but guarantee accountability of their actions. As a consequence, the proposed audit mechanism has a twofold benefit: it acts as a deterrent for third parties with a reputation because any authority can detect and show evidence of wrong or malicious behaviors of the other party; it can enable early detection of cyber attacks of third parties that are behaving maliciously or that have been violated.

Our proposal is based on a novel authorization protocol that requires delegated authorities to store released authorizations in a centralized authenticated log service. Services and devices that validate access requests are able to reject any request that is associated with authorization information that has not been stored in the log service. All authorities can monitor the behavior of the log service and verify its integrity at any time. The proposed protocol protects also the confidentiality of authorization information that is stored in the log service: the log service cannot read authorization information stored in logs, and each authority is able to access only information associated to its authorizations. The protocol supports devices that are characterized by low storage and computing capabilities, and by limited or absent Internet connections.

The performance of the proposed architecture is evaluated by instantiating it through an authenticated log system that is based on Google Trillian. It is an open source project that is used for Certificate Transparency. This evaluation demonstrates that our proposal can be deployed through existing technologies. Moreover, its performance is acceptable for improving the security of real industrial-based systems.

This paper is organized as following. Section 2 models authorization delegation protocols in the context of IIoT and analyzes security threats. Section 3 describes the details of the proposal including its architecture and protocols. Section 4 presents comparative analyses and the experimental results. Section 5 compares our proposal against related work. Section 6 concludes the paper and presents future work.

2. Reference architecture and authorization protocols

We consider the reference architecture for IIoT as represented in Fig. 1. It is based on use case-driven frameworks identified by the Industrial Internet Consortium (IIC) [1,11]. The architecture includes three types of systems: *operational platform*, *domain platform* and *inter-domain platform*.

The operational platform represents the network of the *operational devices* installed in the same physical environment (as an example, an industry production room or a truck) that are connected within local networks called *proximity networks*. Operational devices include typical industrial control systems, such as *actuators*, *sensors* and *controllers*, and *safety monitors*. They are special devices that guarantee the safety of the operational platform, such as activating ventilation systems and deactivating chain trolleys. The *edge gateway* is the device that implements *routing and protocols transformation* operations. It enables communications among devices connected with different heterogeneous local networks (as an example, different LANs, WLANs and PANs) and with the domain platform. Moreover, the edge gateway typically *enforces authentication and authorization* procedures, deciding which communications are allowed (authentication) depending on the access policies decided by the domain platform (authorization).

The domain platform represents an information technology system for analyzing data collected from all devices, and is connected to many operational platforms through *access networks*. Among the many services and components of a domain platform, we focus on *authorizations management* managing the access policies of the operational platform, and on *monitoring and diagnosis* operations for analyzing information collected from operational devices. We consider that a domain platform is controlled by a single *authority*, that is a subject with his own interests, such as an enterprise.

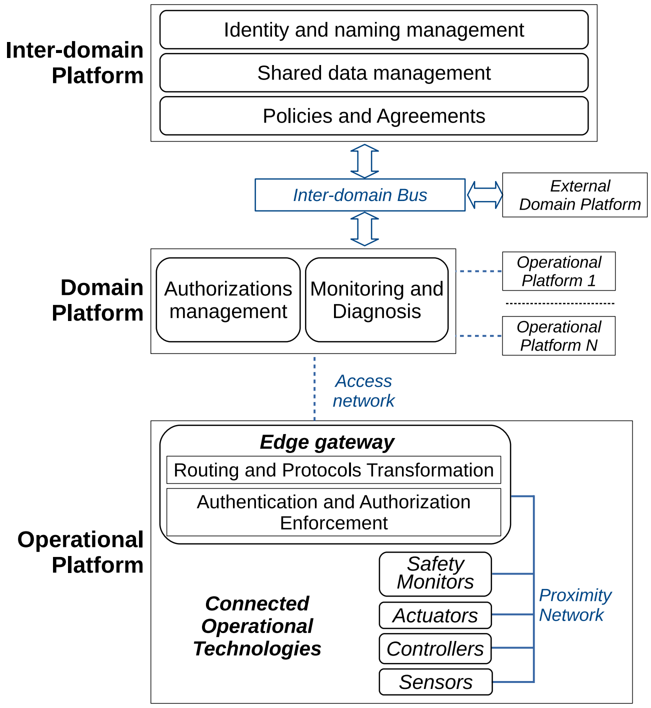

Fig. 1. Reference architecture for Industrial Internet of Things.

The *inter-domain platform* offers a set of services to multiple domain platforms through an *inter-domain bus*. The platform offers *shared data management* functions to allow collaboration among the domains, including sharing logistic, business and operational information. As an example, the inter-domain platform can offer well-known security services for analyzing anomalous network traffic collected from domain platforms and to raise security alerts [12]. In our approach, we focus on *identity and naming management* services that are operated by the *inter-domain platform*, that manage unique addressing and naming of the devices available within the platforms, and on *policies and agreements*, that regulate how the domain platforms can access each other's services.

From a security perspective, we highlight that the inter-domain platform represents a system managed by multiple authorities, and must be designed appropriately with regard to the trust assumptions of the domain platforms that participate in the system. An effective but dangerous approach might assume that the all domain platforms trust each other in operating all authorization procedures correctly. Such an approach might be acceptable in systems with little value at stake, but are not acceptable in industrial environments. In this paper, we design a system that avoids trusted parties. On the one hand, the inter-domain platform stores all the authorization operations of domain platforms and. On the other hand, any domain platform can operate audit operations on the inter-domain platform. Any platform can produce cryptographic proofs that allow to prove misbehaviors of other platforms to other parties publicly. This approach acts as a deterrent against malicious parties, but also incentives all parties to deploy better security measures to avoid being flagged as a flaw in the security of the supply chain.

Before introducing our proposal, we model the authorization framework for the considered IIoT reference architecture. In particular, we are interested in the IT authorization procedures that regulate access to an operational platform. From a cyber-security perspective, to design proper authorization protocols it is important to identify the trust boundaries of the system. We consider a use case system as represented

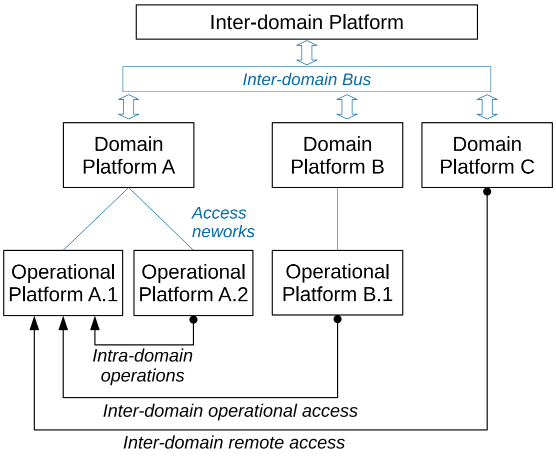

Fig. 2. Operations and authorization procedures within the IIoT architecture.

in Fig. 2, including three domain platforms A, B, C which collaborate by using an inter-domain platform. We identify three types of operations: *intra-domain access*, *inter-domain operational access*, *inter-domain remote access*. *Intra-domain access* represents operations performed within operational platforms of the same domain, such as monitoring and maintenance by local personnel. *Inter-domain operational access* represents operations directly performed between operational platforms of different authorities. Examples of this type of operations are moving operational devices (e.g., robotic machines, transportation systems, external maintenance personnel) or environments populated by devices managed by multiple authorities. *Inter-domain remote access* refer to operations performed across trust boundaries through information technologies, such as remote assistance procedures by external personnel. This type of operation typically distinguishes from inter-domain operational access because it assumes, by the definition of the operation, that the accessed resource is available online and thus less constrained than those placed within isolated operational environments.

Intra-domain access represents all operations within the same trust domain. Due to the adoption of industrial legacy protocols, constrained devices and environments, and strict safety requirements, even enforcing authorizations in intra-domain IIoT environments is a challenging design problem. However, from the point of view of trust management this is not a challenging research issue. We focus on inter-domain operations, that represent a challenging issue in the context of trust management. Both inter-domain operational and remote access must rely on authorization procedures controlled through delegation procedures that involve recursive authorization. Each authority A can delegate coarse-grained authorizations to another authority B. Authority B can then release one or multiple fine-grained authorizations to its operators. In the following Section 3 we propose a novel protocol to improve the security of these delegated authorization procedures in IIoT environments.

3. Auditable and verifiable authorization architecture for IIoT systems

We describe the proposed system for auditable authorizations in IIoT environments in three steps. First, we describe the adopted access control model in Section 3.1. Second, we outline the architecture and operations framework of the system in Section 3.2. Third, we describe the details of the proposed authorization protocol in Section 3.3.

3.1. Access control model and notation

For ease of presentation, we describe our proposal by using a simple discretionary access control model that does not distinguish access privileges (e.g., read, write) and multiple resources available on an operational platform. That is, an authorized client can operate any type of operation on all of the platform resources. Note that more complex access control models, such as fine-grained attribute-based models, can be used as-well without limitations.

We represent an access policy as a tuple $AP = \langle AR, tn, V \rangle$, where AR denotes an access rule, tn denotes the issue time of the authorization policy and V denotes the validity period of the rule. Each access rule identifies the clients that can access to a service as the tuple $AR = \langle C, S \rangle$, where client C can access service S. The validity period defines the time range when the access rule is to be considered valid and is represented as $V = \langle nb, na \rangle$, where nb and na denote the *not before* and *not after* time instants, similarly to x509 Web certificates.

An access token is a cryptographic token defined as the tuple $AT = \left\langle AP, \sigma_{AT,A} \right\rangle$, where AP is the authorization policy that authorizes the client owning the token to access the platform and $\sigma_{AT,A} = sign_{SK_A}(AP)$ is the digital signature of the authorization policy computed by using the secret key SK_A of the authorization service denoted as A (that is, the access token can be verified by anybody that knows public key PK_A).

3.2. Architecture overview

We describe the proposed architecture by referring to Fig. 3. It extends the reference architecture for delegated authorizations (see Section 2, Fig. 2) by requiring all parties to exchange and maintain additional *cryptographic attestations*, designed to ensure non-repudiation of operations.

The architecture includes all the types of data managed in the reference architecture: *authorization policies*, *authorization grants* and *access tokens*. Moreover, the domain platform B and the inter-domain platform maintain additional key-pairs, that we denote as $\langle PK_B, SK_B \rangle$ and $\langle PK_{IDP}, SK_{IDP} \rangle$, respectively. We assume that each secret key is only known by its owner, while public keys are known by all domain and inter-domain platforms. Instead, each operational platform knows only the two public keys associated with its domain platform and with the inter-domain platform. In the considered scenario, we focus on platform *A.1*, that knows PK_A and PK_{IDP}. We observe that the considered assumptions for known public keys are meant to model realistic contexts. It seems feasible for an authority to install its public key on its operational platforms. Instead, installing public keys of other authorities would be non realistic.

The security of the proposed approach is based on two main design choices. First, the system requires domain and inter-domain platforms to authenticate each other messages through *cryptographic attestations*, allowing each platform to prove misbehavior of other platforms to third parties. This involves that a platform A that detects authorizations misuses by the delegated platform B can prove platform B's misbehaviors to third parties. Second, the inter-domain service logs authorization material by using authenticated data structures. As a result, the only security assumption on the authenticated log service is that it is always available, that is, that it answers to all requests. If the service tries to misbehave by returning incorrect answers, the other parties are able to detect it.

For ease of presentation, we distinguish two types of cryptographic attestations: *request attestations* and *signed response timestamps*. The *request attestations* authenticate the data sent by a party in a request. This type of attestation includes *authorization policy attestations* (APA). The *signed response timestamps* assess the acceptance of a request that applies modifications on the state of the service, such as data insertion or update. This type of attestation includes *signed policy timestamp* (SPT) and *signed grant timestamp* (SGT). All signed response timestamps are computed with regard to a request req as the tuple $\langle \mathcal{H}(req), nb, \sigma \rangle$, where $\mathcal{H}(\cdot)$ is a deterministic collision-resistant hash function, such as SHA256, nb is the *"not before"* timestamp that defines at which time instant the modifications required by the request are to be considered

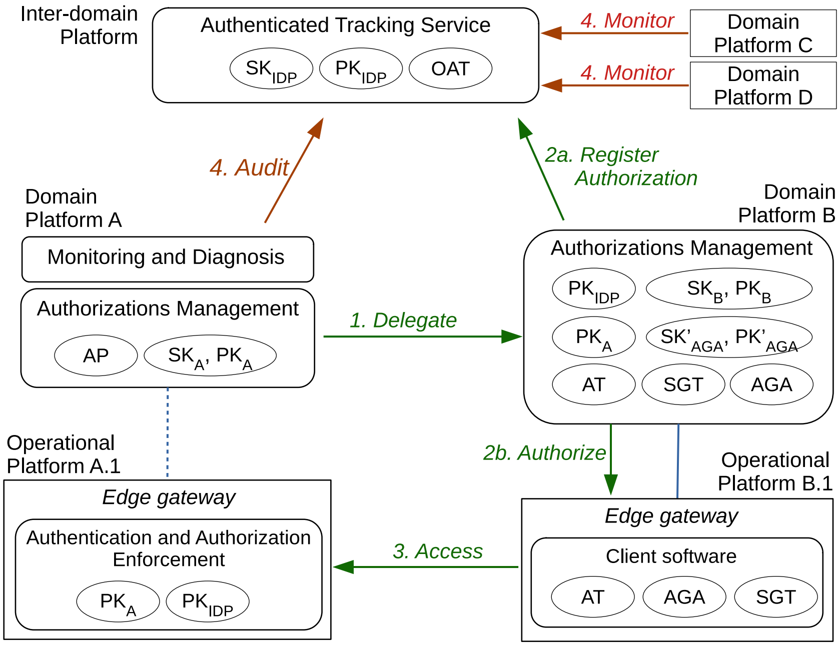

Fig. 3. Authenticated delegated authorization architecture.

applied at the service, and σ is the digital signature of the tuple $\langle \mathcal{H}(request), nb \rangle$ computed by the service that responds to the request.

We do not limit our proposal to specific standards and protocols. Access tokens and attestations can be implemented by using popular standards, such as x509 certificates, JOSE and COSE schemes [13], OpenStack Fernet tokens, and possibly others. In real contexts, we expect that some components might support multiple schemes and choose the most appropriate depending on the underlying communication protocols (as an example, JOSE schemes in text-based protocols and COSE in binary protocols). Similarly, when discussing protocols operated over non-constrained networks we might assume existence of secure channels established via standard protocols without assuming any particular choice. Communicating parties can authenticate each other's identity by using schemes incorporated within secure channel protocols when public keys are known. Typical secure channel protocols might be the TLS and DTLS standards, which allow identity authentication via authenticated DH key-exchange. We do not assume existence of secure channels for communications between operational platforms, which may be based on constrained networks. Instead, as we discuss in the following section, our proposal is able to support peculiar strategies already suggested by authorization protocols for constrained environments [7].

3.3. Operations details

As represented in Fig. 3, the architecture involves four main operations, that are: *(1) delegation*, *(2) authorization*, *(3) access*, and *(4) audit and monitoring*.

(1) The *delegation protocol* allows platform B to obtain the due authorization material to control and release authorizations to platform $A.1$. To request a delegation, B generates a key-pair that we denote as $\left\langle SK'_B, PK'_B \right\rangle$. Then, it builds an *Authorization Delegation Request* (ADR) as:

$$DR = \left\langle \left\langle AP_{AB}, PK_B, PK'_B \right\rangle, \sigma_{DR,B'} \right\rangle, \tag{1}$$

$$ADR = \left\langle \langle DR \rangle, \sigma_{ADR,B} \right\rangle, \tag{2}$$

where AP_{AB} denotes the authorization policy for the requested delegation and, abusing notation for the sake of clarity, $\sigma_{DR,B'}$ denotes the signature generated by using secret key SK'_B. Then, B sends the ADR to A through a secure channel (as an example, a TLS connection), and A validates the request, including: legitimacy of the authorization policy AP (e.g., whether B can indeed access $A.1$ and in which terms); verification of signature $\sigma_{DR,B'}$ by using the public key PK'_B included in the same request, that acts as a proof of possession for secret key SK'_B.

To delegate authorization privileges, platform A generates a unique authorization id (id) as a random byte-string of a length that must be equal to the chosen security level (e.g., 16 bytes for a 128-bit security level). Then, it builds two attestations that we denote as *Delegated authorization Policy Attestation* (DPA) and *Authorization Grant Attestation* (AGA), computed as:

$$DPA = \left\langle \left\langle id, AP_{AB}, PK_B \right\rangle, \sigma_{DPA,A} \right\rangle, \tag{3}$$

$$AGA = \left\langle \left\langle id, PK'_B \right\rangle, \sigma_{AGA,A} \right\rangle, \tag{4}$$

where both $\sigma_{DPA,A}$ and $\sigma_{AGA,A}$ are computed by using secret key SK_A. The first attestation, DPA, authenticates the plaintext delegation information and binds it to the public identity of B, represented by its public key PK_B. The second attestation, AGA, represents the delegation to the possessor of the secret key associated to PK'_B, that is B. The two attestations are used selectively in the following phases of the protocol to build the proof system that does not disclose access control policy information to illegitimate parties. We observe that, from a high-level perspective, the proposed design is similar to the procedure used for requesting a certificate to a certificate authority: in the proposed protocol, the authorization delegation request ADR acts as a certificate sign request, whereas the delegated authorization policy attestation DPA acts as the released certificate. We observe that the validity period of an authorization is much shorter than that of a typical certificate, making it similar to short-lived certificates that are typical of automotive networks. On the other hand, the protocol differs from existing procedures related to certificate authorities because while the policy is associated to DPA, the actual key that will be used to access resources is AGA, which includes key PK'_B and not key PK_B that is known by all parties. Finally, we also observe that the proposed design fits protocols based on well-known Web certificates based on certificate chains, but protocol modifications might allow adoption of other certificate paradigms, such as implicit certificates [14], to reduce the size of the attestations. We leave the investigation of such improvements as a future work.

(2) The *authorization protocol* regulates how organization *B* releases access tokens for allowing its platforms *B.1* to access platform *A.1*. To release an access token to platform *B.1*, platform *B* must build a proper authorization policy AP_{B1} that does not violate the authorization scope granted by platform *A*, that is encoded in AP_{AB} included in the delegation policy attestation DPA (see Eq. (3)).

Platform *B* builds an *Obfuscated Authorization Grant* (OAG) to log the authorization in the inter-domain platform as:

$$OAG = \left\langle \left\langle \mathcal{H}(id), \mathcal{H}\left(AP_{B1} \parallel id\right), PK'_B \right\rangle, \sigma_{OAG,B'} \right\rangle, \tag{5}$$

where signature $\sigma_{OAG,B'}$ is computed by using secret key SK'_B. The first element $\mathcal{H}(id)$ is used as a unique identifier of the attestation within the authenticated data structure of the inter-domain platform. It allows platform *A* to ask for all the authorizations released for a certain identifier id without knowing all the authorization policies associated with access tokens released by platform *B*. The second element of the attestation $\mathcal{H}\left(AP_{B1} \parallel id\right)$ represents a hiding commitment [15] of the authorization operation to the public key of *B* associated to the delegation (PK'_B). Without knowing the unique identifier of the delegation (id), which by construction is a random byte-string that can resist random guesses and enumeration attacks, it is not possible to know the content of the delegation. The third element, which is public key PK'_B, is necessary to make it cheaper for the accessed operational platform *A.1* to operate the verification phase (we postpone the details to the description of that phase below).

The inter-domain platform accepts OAG only if public key PK'_B is valid to verify signature $\sigma_{OAG,B'}$. If valid, it returns a response attestation, that we denote as *signed grant timestamp* (SGT), computed as:

$$SGT = \left\langle \mathcal{H}(OAG), nb_{SGT}, \sigma_{SGT,IDP} \right\rangle, \tag{6}$$

where $\sigma_{SGT,IDP}$ is computed by using secret key SK_{IDP}. The attestation SGT confirms insertion of the value OAG in the authenticated log service within time instant nb_{SGT}.

Finally, platform *B* computes the access token AT as:

$$AT = \left\langle \left\langle AP_{B1}, \sigma_{OAG,B'} \right\rangle, \sigma_{AT,B'} \right\rangle. \tag{7}$$

We observe that including signature $\sigma_{OAG,B'}$ allows to save computation during verification, as we discuss below. Platform *B* releases to platform *B.1* the tuple $\left\langle PK'_B, AT, AGA, nb_{SGT}, \sigma_{SGT,IDP} \right\rangle$.

We observe that the protocol allows domain platform *B* to release quite flexible authorizations to its operational platforms, without any further interactions with platform *A* until policy AP_{AB} is valid. On the other hand, the authorization released to the operational platform should be fine-grained and time-limited, to minimize issues in case of compromised or malicious behaviors by *B.1*, that is more exposed to cyber-attacks.

(3) The *access protocol* defines how platform *B.1* requests access to resources available on *A.1*, and how *A.1* verifies the received material to decide whether to grant or deny access. Platform *B.1* accesses *A.1* by sending a valid access token AT, the authorization grant attestation AGA and the signed grant timestamp SGT (and the payload of the request, which is not of interest to the protocol). Platform *A.1* accepts the request if and only if:

- AGA includes a valid signature $\sigma_{AGA,A}$, verified by using the known public key PK_A;
- AT includes valid authorization policy AP_{B1} (as an example, the access control rule and validity fields are both valid) and digital signature $\sigma_{AT,B'}$ (verified by using the public key PK'_B in AGA and thus approved by platform *A*);
- the verification of signature $\sigma_{SGT,IDP}$ succeeds. To verify it, *A.1* must first rebuild the tuple OAG (see Eq. (5), please note that all the due material is already available within the received data). Then, *A.1* must verify the validity of the digital signature $\sigma_{SGT,IDP}$ against the tuple $\langle \mathcal{H}(OAG), nb_{SGT} \rangle$ by using the known public key PK_{IDP}.

We observe that the most expensive operations of this procedure are the three signature verification operations ($\sigma_{AGA,A}$, $\sigma_{AT,B}$ and $\sigma_{SGT,IDP}$). However, we highlight that our design choices allow to avoid a fourth signature verification, that is, that of $\sigma_{OAG,B'}$. In a somehow simpler alternative design where signature $\sigma_{OAG,B'}$ is not included, the platform *A.1* must verify that the OAG inserted within the log has indeed been produced by platform *B.1*. Our design avoids such verification operation by relying on the fact that the attestation OAG received by platform *A.1* is explicitly bound to public key PK'_B, and that the inter-domain platform would refuse to accept any invalid OAG.

If the access request is operated over constrained networks that do not support standard secure channels, the access token can be used as a first message to operate a challenge–response protocol, where platform *A.1* challenges platform *B.1* by using the public key PK'_B. This is a known approach to provide some protection against replay attacks in constrained environments. For further details, please refer to proof of possession tokens as described in proposed extensions for Authentication and Authorization for Constrained Environments to the OAuth2 standard [7].

(4) Audit and monitor protocols allow domain platforms to control the correct behavior of each domain platform and of the inter-domain platform. In the considered architecture, platform *A* performs audit operations to control operations authorized by platform *B*, and the other domain platforms (*C* and *D*) to detect misbehaviors operated by the inter-domain platform on logged information.

Platform *A* can control authorizations released by *B* with regard to a certain authorization delegation by using the authorization grant id (id), decided by *A* itself in the delegation protocol by querying the inter-domain platform. Authenticated data structures are able to provide proofs of membership for all the information stored within the log. If the service stores authorization material, it returns OAG data structures including obfuscated delegation information signed by platform *B*. To obtain plaintext information, platform *A* queries *B* with the retrieved OAG. Platform *B* must comply with the protocol by returning the authorization policy AP_{B1} that allows to recompute $\mathcal{H}\left(AP_{B1} \parallel id\right)$ (see Eq. (4)). If the policy AP_{B1} violates policy AP_{AB} included in the DPA with same id (see Eq. (3)), then platform *B* misbehaved.

The inter-domain platform updates the authenticated data structures periodically, publishing a digest of the data structure that acts as a representative of the updated state (as an example, the root of a Merkle Hash Tree). As the digest is also digitally signed and thus non-repudiable by the inter-domain platform, it acts as a proof for all the data maintained by the platform itself. To verify the correct behavior of the platform, platforms that act as monitors must retrieve the data stored in the authenticated data structure and verify the correctness of the digest. Our design choice allows any domain platform within the system to monitor the inter-domain platform without getting information about authorizations that do not involve them, thus the system also protects the confidentiality of the authorization information from parties that are not involved.

4. Analyses and evaluations

We analyze the benefits of the proposed architecture by comparing it against related approaches (Section 4.1). Then, we measure its performance and we analyze its applicability to IIoT scenarios (Section 4.2).

4.1. Comparison with alternative approaches

For comparison purposes, we consider the *trusted platforms* described in Section 2, where both domain and inter-domain platforms are trusted, and two families of approaches: a *blockchain-based* approach where released authorizations are stored in a public distributed

Table 1
Advantages and disadvantages of related families of approaches.

Approach	Advantages	Disadvantages
Trusted platforms	Lightweight	Security and trust issues
Blockchain-based log	Cryptographic proofs High availability Prevent misuse	High latencies for write operations Large storage requirements Internet-connected devices
Intrusion detection system	Wider features No modifications to devices	No cryptographic proofs
Proposal	Cryptographic proofs Support offline devices Confidentiality	Security trade-offs Small overhead on access requests

ledger maintained by all the platforms and devices through a state-replication consensus protocol; an *intrusion detection system* where authorization misuses are detected by inspecting network traffic. A summary of the comparative analysis is reported in Table 1, and discussed below.

The first advantage of the trusted platforms approach is its lightweight performance, because it only requires domain platforms to send information to the inter-domain platform, which is trusted in storing them and in not violating confidentiality of authorization information. However, this approach has severe security issues because it does not match the considered threat model where domain and inter-domain platforms are untrusted.

Storing all authorization information on a blockchain-based system would inherit the advantages of the blockchain including high availability of the information and the capability of enforcing verification of authorizations at insertion time by using smart contracts [16]. On the other hand, this system inherits the disadvantages of the blockchain that can introduce high delays at insertion time and costs (e.g., [17]). Moreover, all devices that must verify correctness of the requests would require large storage to maintain the blockchain and Internet-connectivity to guarantee updates.

An intrusion detection system is a quite different approach, but it might guarantee similar attributes. Its main advantages include the ability to operate without introducing modifications to devices and to detect a much wider range of attacks. However, it is unable to guarantee a cryptographic proof of misbehavior.

Our proposal does not require to trust domain and inter-domain platforms thanks to cryptographic assertions exchanged by the parties. Misbehaviors by the inter-domain platform can be detected thanks to authenticated data structures. The inter-domain platform cannot access confidential information because domain platforms encrypt and obfuscate data before storing them. The cryptographic schemes on which all protocols are based, which are message authentication codes, hash functions and digital signatures, can be supported by many classes of devices. Even more important, IIoT devices are allowed to operate offline because they do not require frequent updates nor to maintain a large amount of data. As a result, the proposed approach represents the best trade-off in terms of performance, security and flexibility for securely attributing authorizations misuse in industrial scenarios.

4.2. Experimental evaluation

We analyze the performance of the authenticated log service that represents a potential bottleneck in the proposed architecture. To this aim, we consider deploying the proposal on Google Trillian [18], that implements an authenticated data structure based on Merkle Hash Trees (MHT). As described in Section 3, the service receives new Obfuscated Authorization Grants (OAG) and returns Signed Grant Timestamps (SGT). For performance reasons, a new OAG is not inserted synchronously within the MHT, but the service inserts it periodically through batch merge operations. This service guarantees the clients that a received OAG will be merged within a specific time window through the *not before* (nb) value within SGT. The delay required to merge an operation is called *maximum merge delay* (MMD).

Authorization mechanisms have different requirements in terms of service operation latencies and maximum merge delays. As an example, a scenario may require a log service to grant authorizations within few milliseconds and involve audit operations every few days or weeks. Other scenarios may have opposite requirements, such as allowing long authorization delays (e.g., programmable interventions) but short audit times (e.g., the operation is critical and an invalid authorization must not be allowed or be detected within very short time intervals).

The trade-off between performance and costs should be clear. As authorizations can be audited only after merging of authorization grants, higher MMD values denote lower costs and lower quality of the service. On the other hand, lower MMD values are more expensive because MHT update operations are characterized by low degree of parallelism. As a result, merge delays must be sized appropriately depending on the throughput of insertion operations. In our evaluation, we consider different amounts of operations and we measure the service latency for each of them. To the aim of analyzing the behavior of the proposed system and of determining the best candidate values for different maximum merge delays, we evaluate the performance for increasing amounts of concurrent authorization operations.

We consider a real scenario represented by authorization systems for air-gapped environments, where operators and machines (e.g., automated trolleys) must request access to resources that cannot always verify authentication and authorization material online. In these environments, a moving device must obtain authorization grants from an authorization service, move to the air-gapped location, and present these grants to offline devices, that must decide to accept or reject the material. We estimate the number of automated devices that move in industrial plants to be highly variable, depending on the size of the environment and the types of productions that can range from few units to tens of requests. In the emulated environment, that use fine-grained and short-lived authorizations, the number of requested authorizations can be in the order of hundreds per second. We evaluate the performance of the system by measuring:

- the time required to confirm the acceptance of a new authorization from the domain platform. All operations timings are measured through the Trillian testing utility;
- the total *authorization log time*, that is, the time required by the service to merge new authorizations in the MHT and make them available for auditing. The log time is evaluated by measuring merge delays through querying the hash of the log transaction and extracting the *queue timestamps* and the *integration timestamp* that are two metadata included in the MHT. Then, we compute the log time as the difference between the two values;

For both measures, we perform multiple evaluations that are based on the following configurations:

- we consider increasing operation workloads to evaluate the scalability of the system, including 10, 100 and 200 insert operations per second. The requests are sent until 10 000 logs are integrated into the log tree. In order to avoid measurement biases we repeat the evaluation 50 times and consider the average value.

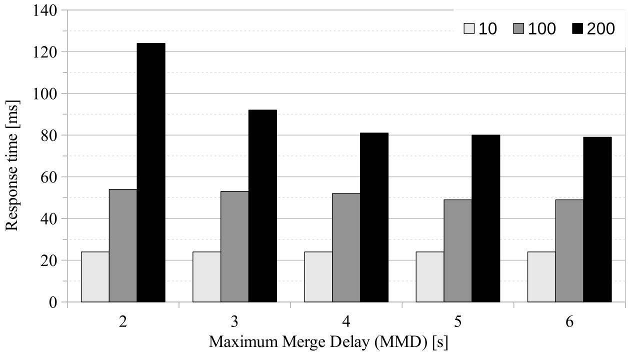

Fig. 4. Latency of the authorization insertion operations.

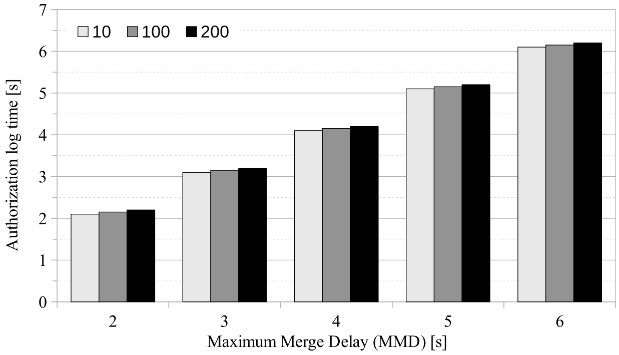

Fig. 5. Total log time of new authorizations.

- we evaluate the influence of MMD on the system performance by considering five MMD configurations: 2, 3, 4, 5 and 6 s;
- we measure the integration time by combining the operation latencies and the maximum merge delay.

The server machine executing the authenticated log service is configured with five virtual CPUs, 50 GB of RAM and an SSD hard drive. We observe that such an amount of RAM has been chosen to avoid potential IO bottlenecks, and the proposed evaluation would probably work with a lower amount of RAM and achieve the same performance. Fig. 4 shows the response times of the service for new authorizations. The y-axis reports the average response times in ms. The x-axis denotes the results for the three workloads and increasing MMD values. The average response times typically range from 22 ms to 80 ms as a function of increasing workloads. We can observe that low values of MMD, in the order of 2 s or less, greatly affect the performance of the system in the case of high workloads. Indeed, the system tends to saturate for configurations with $MMD = 2$ s and more than 100 operations per second. This result evidences that for a similar system, adequate performance is achieved when the service is configured with values of MMD greater the 3 s.

Fig. 5 reports the *authorization log time*, where the x-axis and the y-axis denote the three workloads for increasing MMD values, and the log times expressed in seconds, respectively. This figure confirms that the total log time of the proposed mechanism is unaffected by the required number of operations per second, as the log time is almost equal to the choice of the MMD value. In practice, some milliseconds are added to the basic MMD value that is in the order of seconds. These results confirm the exceptions about the proposed system, when MMD is a reliable setting to control authorizations even in an industrial setting where smart devices have low computational capacities.

Based on previous results and four main applications, we can claim that the proposed system is applicable to any realistic scenario. The performance of the system is characterized by two features: the access time that is, the amount of time required to complete the procedure related to granting access; the log time (represented as MMD) that is the time taken to provide reason to the end-user in the case of an adverse decision to an access.

Table 2
IIoT application latency assuming 3-seconds MMD.

IIoT application	Access time	Log time
Smart Car	3.5–4 s	24 h
Smart Locker	3.5–4 s	3.5–4 s
Smart Lab	Few weeks	3.5–4 s
Smart Warehouse	Few days	24 h

For example, in the case of a Smart Car belonging to the factory fleet a typical employee can directly approach a car to borrow it for commuting. In a similar case, the time required to complete the procedure for granting access should be small. On the other hand, if the access is denied, the employee may move to the next available car (or even take his owned vehicle) and check the logs for denial later. In this case, the log time can be large.

The case for Smart Lockers is similar to the case for Smart Cars. The time required for access setup is very short as the employee is already at the locker requesting for access. If the access is denied, the employee needs to know the reason immediately as he may have used the locker to keep his stuff.

Now consider the case of a Smart Lab, where the (team of) employees typically begin the booking process in advance, so there is sufficient time for the completion of access related procedure. If the access is denied, the employees would like to know the reason immediately so that the problem can be solved and they can avoid delay or rescheduling issues.

For the Smart Warehouse application, let us consider the case where an online retailer is going to deliver a package at the warehouse, but no warehousemen is available for delivery processing/acknowledgment. Thus, an employee creates the access rule for the delivery person. The access time to setup the delegation is high. When the delivery person arrives, the access is denied. There is no remedy but to leave the package outside unattended or to take back. Later on, the delivery person fills a complaint report about the access denial, thus allowing some time for logs.

Table 2 summarizes the four smart applications with an estimated access time and log time characteristics.

For an asymptotic analysis, we have to observe that a tree can manage a maximum of 2^{63} transactions. If n denotes the number of delegating domain platforms, k the number of devices, and x stands for the number of transactions generated by each device per day, we can easily evaluate that a saturation may occur after some thousands of years. Even in a large industry with hundreds of delegators, thousands of devices and hundreds of thousands of transactions per day, we can reach some dozen of billions of log records per day that remain largely below 2^{63}. This analysis evidences meaning that our solution never saturates in practice.

5. Related work

The proposed architecture relates with authorization architectures and protocols for distributed systems that are governed by multiple authorities, including federated identity systems such as the well-known OpenID [19] or the so-called self-sovereign identity systems [20], and authentication and authorization protocols specifically designed for constrained environments [7,21]. Standard solutions for authorization services protect the environment from illegitimate accesses from unauthorized personnel, but they typically do not consider malicious internal attackers that have high privileges, such as admins that have complete accesses. As internal attacks are increasing, our proposal that allows auditing and attribution of illegitimate behaviors increases the security of the IIoT systems.

Detecting violations within outsourced services is related to the capability of detecting integrity of algorithms and of data. Security proposals include expensive cryptographic protocols for verifiable outsourcing [22], which however are unsuitable because of high computational overhead. Other approaches for detecting correctness of algorithms and protocols are based on secure hardware enclaves (also known as *trusted execution environments*), such as SGX [23]. These approaches allows to guarantee integrity of algorithms operated in adversarial environments, making them a practical alternative solution to cryptographic protocols for verifiable outsourcing. Since they already achieve practical performance and are already available in modern CPUs, they are considered a promising approach to increase the security of existing systems. As a disadvantages, these technologies seem still vulnerable to attacks [24] and thus their deployment in critical systems should be carefully evaluated. To the best of our knowledge, no solution for verifiable authorization protocols based on trusted execution environments has been proposed. Investigating such an approach could be an interesting research area.

Our proposal is also related to decentralized security solutions designed for multiple root authorities, including blockchain technologies [25,26], secure multi-party computation [27], and transparency logs [28–30]. Blockchain protocols based on consensus and secure multi-party proposals cannot be used in many industrial environments due to device and network constraints. The proposed architecture is based on *transparency logs* [28] that allow the system to outsource a log service to semi-trusted parties without affecting the security of the system. These approaches are based on known cryptographic schemes to guarantee data integrity, such as digital signatures, MAC primitives, and so-called authenticated data structures to enable efficient operations for dynamic workloads [31,32]. We exclude proposals that are characterized by high computational, storage and traffic requirements that are unsuitable to IIoT environments. Proposals deploying lightweight and flexible authentication and authorization schemes in constrained environments [7,21] do not guarantee strong auditability. To the best of our knowledge, no existing proposals based on transparency logs consider the challenges related to auditing authorizations of IIoT systems as we considered in this paper.

The most popular *transparency log* solution is Certificate Transparency [28] that monitors Certification Authorities by requiring browsers to verify that all certificates fetched from Web sites have been stored in approved authenticated logs. Literature also proposes a more general approach called *key transparency* [29] that prevents the so-called *equivocation*, which is an attack based on binding different cryptographic keys to the same identity. This solution cannot be trivially applied to the considered industrial scenario due to the different characteristics of the delegated authorization protocols, including management of secret information and updates to the authorization policies. Finally, a proposal based on transparency logs has been proposed by the same authors for smart cities [30]. It allows users to detect malicious behaviors of cloud services that act as delegated authorization services, but it cannot be used in federated environments characterizing industrial systems where users and authorization service belong to the same trust domain and might collude to avoid detection.

We observe that variants of the proposed architecture might be designed on other types of transparent logs, including verifiable transparent logs based on other technologies, including trusted execution environments [33] or distributed ledger technologies [25]. However, the convenience of each alternative approach should be considered with regard to the requirements of the considered scenario. In the considered industrial-based environment, our proposal already achieves practical performance, does not require devices connected to the Internet or with high storage and computational resources, and does not assume security of hardware enclaves.

Security solutions that are less related to our proposal, but that also contribute to guarantee security of IIoT architectures, include network segregation based on firewalls and data diodes (e.g., [34]), and intrusion detection systems (e.g., [35]). Unfortunately, industrial systems are characterized by legacy industrial protocols, network protocols with small packet sizes that cannot be naïvely integrated with security measures, and network-enabled sensors with low computational power that cannot operate many standard security protocols. In literature, there are proposals that are designed for IIoT systems, such as network intrusion detection systems for automotive networks [36] and automation systems [12]. Intrusion detection systems can cover a wider range of attacks other than those related to authentication and authorization procedures, but they are unable to provide strong evidence of misbehaviors. Moreover, their adoption in possibly disconnected networks can delay audit and detection of threats, and there are useless in the case of encryption network protocols.

Other proposals that are complementary to ours include several research proposals for scalable identification systems, such as physical unclonable functions [37], strong authentication protocols to control remote access [38], architectures for offloading expensive cryptographic operations [39] and lightweight asymmetric cryptographic primitives and protocols [40,41]. Additional challenges are related to the security of the software operated by the IoT devices [42], including the capability of verifying whether a device has been compromised and the possibility of fixing vulnerabilities through software patching. Major proposals include remote attestations to verify the integrity of the executed software [43], and scalable and reliable architectures for long-term software updates [44]. The issues related to the security of a specific device are out of the scope of this paper. The proposed approach is orthogonal to security solutions deployed within the industrial network. Our proposal requires that IoT devices authenticating access requests must support digital signatures, but it is aware that these devices may have limited computational and storage capabilities. Hence, it minimizes the number of cryptographic operations and requires the storage of a small number of keys.

6. Conclusions

We propose a system that allows to audit authorization procedures operated in industrial IoT environments, characterized by highly secure air-gapped systems and devices with low resources placed in constrained environments. The proposed design is compliant with standard authorization and network communication protocols and can leverage existing software services and libraries for a reliable deployment. Its security is based on established cryptographic protocols, such as standard digital signature schemes and hash functions, and allows each party involved in the system to prove misbehaviors publicly, incentivizing each industrial party involved in a collaboration to adopt the best security practices to avoid misbehaviors due to internal or external adversaries. The proposed experimental evaluation operated by using a prototype implementation based on the established Google Trillian project, shows the feasibility of the system even in presence of intensive operation workloads.

CRediT authorship contribution statement

Luca Ferretti: Conceptualization, Methodology, Formal analysis, Validation, Writing - original draft, Writing - review & editing. **Francesco Longo:** Conceptualization, Methodology, Formal analysis, Validation, Writing - original draft. **Giovanni Merlino:** Conceptualization, Validation, Writing - original draft , Writing - review & editing. **Michele Colajanni:** Supervision, Project administration, Writing - original draft. **Antonio Puliafito:** Supervision, Project administration, Resources, Writing - original draft. **Nachiket Tapas:** Software, Investigation, Validation, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Laurie B, Langley A, Kasper E. Industrial internet of things volume g1: reference architecture. IIC PUB, Industrial Internet Consortium; 2019.

[2] Jazdi N. Cyber physical systems in the context of industry 4.0. In: 2014 IEEE international conference on automation, quality and testing, robotics. IEEE; 2014, p. 1–4.

[3] Ting S, Tse Y, Ho G, Chung S, Pang G. Mining logistics data to assure the quality in a sustainable food supply chain: A case in the red wine industry. Int J Prod Econ 2014;152:200–9.

[4] Waller MA, Fawcett SE, science Data, analytics predictive. Data science predictive analytics and big data: a revolution that will transform supply chain design and management. J Bus Logist 2013;34:77–84.

[5] Ivanov D, Tsipoulanidis A, Schönberger J. Digital supply chain, smart operations and industry 4.0. In: Global supply chain and operations management. Springer; 2019, p. 481–526.

[6] Aceto G, Persico V, Pescapé A, Industry. 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0. J Ind Inf Integr 2020;18:100129.

[7] Seitz L, Selander G, Wahlstroem E, Erdtman S, Tschofenig H. Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth). Internet-draft 2018.

[8] Fauri D, d. Wijs B, den Hartog J, Costante E, Zambon E, Etalle S. Encryption in ics networks: A blessing or a curse?. In: IEEE international conference on smart grid communications (SmartGridComm). 2017.

[9] Asharov G, Orlandi C. Calling out cheaters: Covert security with public verifiability. In: International conference on the theory and application of cryptology and information security. Springer; 2012, p. 681–98.

[10] Hong C, Katz J, Kolesnikov V, Lu W-j, Wang X. Covert security with public verifiability: Faster, leaner, and simpler. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2019, p. 97–121.

[11] Laurie B, Langley A, Kasper E. Industrial internet security framework. IIC PUB, Industrial Internet Consortium; 2016.

[12] Fauri D, Kapsalakis M, dos Santos DR, Costante E, den Hartog J, Etalle S. Leveraging semantics for actionable intrusion detection in building automation systems. In: International conference on critical information infrastructures security. Springer; 2018, p. 113–25.

[13] Schaad J, Cellars A. RFC 8152: CBOR object signing and encryption (COSE). RFC; 2017.

[14] Pollicino F, Stabili D, Ferretti L, Marchetti M. An experimental analysis of ECQV performance on VANETs. In *Proc. 92th IEEE vehicular technology conf.* 2020.

[15] Damgård IB, Pedersen TP, Pfitzmann B. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In: Annual international cryptology conference. Springer; 1993, p. 250–65.

[16] Tapas N, Merlino G, Longo F. Blockchain-based iot-cloud authorization and delegation. In *Proc. IEEE int'l conf. smart computing*, 2018.

[17] Magnanini F, Ferretti L, Colajanni M. Efficient license management based on smart contracts between software vendors and service providers. In: 2019 IEEE 18th international symposium on network computing and applications (NCA). IEEE; 2019, p. 1–6.

[18] Google. Trillian: General transparency. 2018, https://github.com/google/trillian/tree/67395f8e7cf7f94ef80ac1846ec10c49a4d6550f.

[19] Recordon D, Reed D, Openid. Openid 2.0: a platform for user-centric identity management. In: Proceedings of the second acm workshop on digital identity management. ACM; 2006, p. 11–6.

[20] Mühle A, Grüner A, Gayvoronskaya T, Meinel C. A survey on essential components of a self-sovereign identity. Comp Sci Rev 2018;30:80–6.

[21] Pereira PP, Eliasson J, Delsing J. An authentication and access control framework for coap-based internet of things. In: IECON 2014-40th annual conference of the ieee industrial electronics society. 2014.

[22] Parno B, Howell J, Gentry C, Raykova M. Pinocchio: Nearly practical verifiable computation. In: IEEE symp. security and privacy. 2013.

[23] Karande V, Bauman E, Lin Z, Khan L. SGX-log: Securing system logs with sgx. In *Proc. ACM Asia conf. computer and communications security*, 2017.

[24] Lee S, Shih M-W, Gera P, Kim T, Kim H, Peinado M. Inferring fine-grained control flow inside sgx enclaves with branch shadowing. In *Proc. 26th USENIX security symp.*, 2017.

[25] Tomescu A, Devadas S. Catena: Efficient non-equivocation via bitcoin. In *Proc. 38th IEEE symp. security and privacy*, 2017.

[26] Novo O. Blockchain meets IoT: An architecture for scalable access management in IoT. IEEE Internet Things J 2018;5.

[27] Ben-David A, Nisan N, Pinkas B. FairplayMP: a system for secure multi-party computation. In *Proc. 15th ACM conf. computer and communications security*, 2008.

[28] Laurie B, Langley A, Kasper E. RFC6962: certificate transparency. RFC; 2013.

[29] Bonneau J. Ethiks: Using ethereum to audit a coniks key transparency log. In: Proc. int'l conf. financial cryptography and data security. Springer; 2016.

[30] Ferretti L, Longo F, Colajanni M, Merlino G, Tapas N. Authorization transparency for accountable access to iot services. In *Proceedings of the third international conference on internet of things (ICIOT 2019)*, 2019.

[31] Miller A, Hicks M, Katz J, Shi E. Authenticated data structures, generically. In *Proc. 41st ACM SIGPLAN-SIGACT symp. principles of programming languages*, 2014.

[32] Ferretti L, Marchetti M, Andreolini M, Colajanni M. A symmetric cryptographic scheme for data integrity verification in cloud databases. Inform Sci 2018;422:497–515.

[33] Paccagnella R, Datta P, Hassan WU, Bates A, Fletcher C, Miller A, Tian D. Custos: Practical tamper-evident auditing of operating systems using trusted execution. In: Network and distributed system security symposium. 2020.

[34] Jones DW, Bowersox TC. Secure data export and auditing using data diodes. Technology 2006;6:7.

[35] Colajanni M, Marchetti M. A parallel architecture for stateful intrusion detection in high traffic networks. In *Proc. of the IEEE/IST workshop on" monitoring, attack detection and mitigation"(MonAM 2006)*, Tuebingen, Germany, 2006.

[36] Marchetti M, Stabili D. Anomaly detection of CAN bus messages through analysis of ID sequences. In: 2017 IEEE intelligent vehicles symposium (IV). IEEE; 2017, p. 1577–83.

[37] Suh GE, Devadas S. Physical unclonable functions for device authentication and secret key generation. In: 2007 44th ACM/IEEE design automation conference. IEEE; 2007, p. 9–14.

[38] Srinivas S, Balfanz D, Tiffany E, Czeskis A, Alliance F. Universal 2nd factor (u2f) overview. FIDO Alliance Propos Stand 2015;1–5.

[39] Aazam M, Zeadally S, Harras KA. Offloading in fog computing for iot: Review, enabling technologies, and research opportunities. Future Gener Comput Syst 2018;87:278–89.

[40] Renes J, Schwabe P, Smith B, Batina L. μKummer: efficient hyperelliptic signatures and key exchange for microcontrollers. In: Gierlichs B, Poschmann A, editors. Cryptographic hardware and embedded systems – CHES 2016. Lecture notes in computer science, vol. 9813, Springer-Verlag Berlin Heidelberg; 2016, p. 301–20, Document ID: b230ab9b9c664ec4aad0cea0bd6a6732, http://cryptojedi.org/papers/#mukummer.

[41] Ferretti L, Marchetti M, Colajanni M. Fog-based secure communications for low-power iot devices. ACM Trans Internet Technol 2019;19:27:1–21. http://dx.doi.org/10.1145/3284554, URL: http://doi.acm.org/10.1145/3284554.

[42] Da Xu L, He W, Li S. Internet of things in industries: A survey. IEEE Trans Ind Inform 2014;10.

[43] Nunes IDO, Eldefrawy K, Rattanavipanon N, Steiner M, Tsudik G. Vrased: A verified hardware/software co-design for remote attestation. In *Proc. 28th USENIX security symposium*, 2019, p. 1429–46.

[44] Villegas MM, Orellana C, Astudillo H. A study of over-the-air (ota) update systems for cps and iot operating systems. In: Proceedings of the 13th european conference on software architecture - Volume 2, ECSA '19. New York, NY, USA: ACM; 2019, p. 269–72. http://dx.doi.org/10.1145/3344948.3344972, URL: http://doi.acm.org/10.1145/3344948.3344972.