Trusted and only Trusted. That is the Access! Improving Access Control allowing only Trusted Execution Environment Applications

Dalton Cézane Gomes Valadares
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE)
59 PUBLICATIONS **127** CITATIONS
SEE PROFILE

Alvaro Sobrinho
Universidade Federal do Agreste de Pernambuco
67 PUBLICATIONS **220** CITATIONS
SEE PROFILE

Newton Carlos Will
Federal University of Technology - Paraná
35 PUBLICATIONS **97** CITATIONS
SEE PROFILE

Kyller Gorgonio
Universidade Federal de Campina Grande (UFCG)
77 PUBLICATIONS **303** CITATIONS
SEE PROFILE

Trusted and only Trusted. That is the Access!
Improving Access Control allowing only Trusted Execution Environment Applications

Dalton C. G. Valadares, Álvaro Sobrinho, Newton C. Will, Kyller C. Gorgônio, Angelo Perkusich

Abstract Security concerns should always be considered when deploying distributed systems that deal with sensitive data. Generally, the software components responsible for storing these sensitive data are protected, having access control systems to allow or deny external requests. A Policy Enforcement Point (PEP) Proxy is one of these systems which allows or denies access to protected data by checking if the requester is authorized and has permission to access. Despite these two validations about the requester (authorization and data access permission), the traditional PEP Proxy does not guarantee anything more about the requester which will process the data. This work proposes an improvement to the PEP Proxy protection in a way that it can also verify if the requester runs on a Trusted Execution Environment (TEE) application. A TEE is responsible for trusted computing, processing data in a protected region of memory, which is tamper-resistant and isolated from external resources, and keeping code and data protected even if the operating system is hacked. The Trusted PEP Proxy (TruPP) performs the remote attestation (RA) process to guarantee that the requester runs on a TEE. We created a Coloured Petri Net (CPN) model to help validate our proposal by checking some security properties.

1 Introduction

The advances in the Internet of Things (IoT) technologies have attracted the attention of industries, researchers/academies, and different kinds of customers. Since IoT devices are increasingly smarter, they are frequently called smart objects [8, 12]. Despite this, these smart objects have specific constraints regarding battery, memory, and processing capabilities. Thus, a requirement for many applications is the use of a more robust device, generally called as gateway, with more processing and memory power and fewer constraints.

According to the local where this gateway is placed/deployed and the preprocessing is performed, we have the concepts of edge and fog computing, which are

Dalton C. G. Valadares
UFCG/VIRTUS RDI Center, Campina Grande, PB, Brazil
Federal Institute of Pernambuco, Caruaru, Brazil, e-mail: dalton.valadares@embedded.ufcg.edu.br

Álvaro A. C. C. Sobrinho
Federal University of the Agreste of Pernambuco, Garanhuns, Brazil
e-mail: alvaro.alvares@ufape.edu.br

Newton C. Will
Federal University of Technology - Paraná, Dois Vizinhos, Brazil, e-mail: will@utfpr.edu.br

Kyller C. Gorgônio and Angelo Perkusich
UFCG/VIRTUS RDI Center, Campina Grande, Brazil
e-mail: kyller@computacao.ufcg.edu.br, perkusic@dee.ufcg.edu.br

alternatives to cloud computing. When the gateway is located together with the IoT devices, with local data processing, this scenario is named edge computing [23]. When IoT applications demand even more processing power, with increasing restricted time requirements, we can deploy specific servers locally to provide the needed resources for gateways and IoT devices instead of sending data directly to the cloud. This scenario is the basis of fog computing, in which IoT devices collect data from the environment and send them to a fog server. This server preprocesses these data before sending them to a cloud server, for instance [24, 9]. Fog computing can reduce the latency when certain situations require fast responses in the field, and it can also decrease costs with cloud processing and communication.

Regardless of edge or fog computing, IoT applications often have to deal with sensitive data, like Personally Identifiable Information (PII), such as medical, financial, or educational information. These kinds of information can include name, fingerprint, telephone number, and social security number, which requires special care regarding overall data security. A particular type of PII is named Protected Health Information (PHI), which is related to health information from patients in healthcare organizations. This information has become an attractive target for criminals since the PHI records do not include only health data but also valuable personal information (e.g., address or social security number).

The interest in these data is even greater because they do not use to change, unlike credit card and bank account numbers, which can be canceled. In addition, while credit card numbers can be sold for US$1 on the black market, and PII can be sold for US$10 up to US$20, medical information such as PHI can reach between US$20 and US$50, being the most valuable kind of information [14]. The number of PHI breaches reported by the US federal government in 2016 reached the health information of more than 15 million people, with the biggest case happening in Arizona through a network server hacking attack, which involved more than 3.7 million patients' health information [18]. Also, in the same year, a successful ransomware attack forced a hospital in Los Angeles to pay US$17000 in order to get its resources and network released.

Another known scenario that demands data protection, although it does not deal with PII, is a smart metering application that is responsible for periodically measuring the energy consumption of a place. Suppose an adversary gets access to a data set containing energy consumption measurements from a specific period (e.g., a month). In that case, this adversary can use a Non-Intrusive Appliance Loading Monitoring (NIALM) technique to estimate what appliances were being used at a specific time and, by doing this, estimate how many people were in that place and what these people were doing [15, 24].

Security mechanisms should be adopted to avoid this kind of problem, considering, for example, data and communication channel protection. Encryption techniques can be used to protect data, and the communication channel can be secured through TLS (Transport Layer Security). Besides, access to sensitive data must be controlled, allowing only authenticated and authorized parties. A common way to control data access is deploying a Policy Enforcement Point (PEP) Proxy before the data service that needs to be secured. A PEP Proxy is responsible for intercepting requests to a protected service and checking if this request comes from an authenticated and authorized party. The intercepted request usually contains a token that should be validated with an Identity Management (IdM) system to check if the requester was authenticated and is authorized. In a positive case, the requester gets access to the data service; otherwise, the access is denied, and the data remain safe [24].

Although data can be accessed only by authenticated and authorized requesters, there is no reason to trust their machines where data are processed since there is the risk that an adversary can get access to their operating systems with high-level privileges and thus get access to sensitive data. We established two research questions as guidance to deal with this concern and achieve a solution:

1. How to improve security by decreasing the distrust in the data consumers/requesters?
2. How to improve access control in a way that only trusted parties can access sensitive data?

In this sense, Trusted Execution Environments (TEE) have become frequently adopted to increase the security of data and systems. A TEE is enabled due to an extension of the processor instructions set, which allows the creation of shielded memory spaces isolated from the rest of the system. The two TEE technologies more common in the market and the scientific communities are ARM TrustZone and Intel Software Guard Extensions (SGX). The protected memory regions are commonly called "secure world" (TrustZone) or "enclaves" (SGX), and even an adversary with high-level privileges should fail when trying to access these specific memory regions. Attestation protocols allow third parties to validate that an application is running on a valid TEE. These protocols provide specific information about the secure world/enclave, verifying its authenticity by enabling third parties to check and validate this information [13, 27].

In this work, considering both research questions, we propose an improvement to the PEP Proxy in a way that it can also perform the attestation process. Thus, after authentication and authorization checking, the attestation process is initialized to verify if the requester application, which is interested in sensitive data, runs on a TEE and, therefore, is considered a trusted entity. We created a Coloured Petri Net (CPN) model representing the communication flow with a PEP Proxy that also performs the attestation process besides authentication and authorization. With the CPN model, we can validate security properties. Thus, the main contribution of this solution is that we improve the trust in data consumers/requesters by allowing sensitive data distribution only to trusted entities, which means that these entities were previously authenticated, authorized, and attested.

The remainder of this paper is organized as follows: we present the background in Section 2; we describe the fundamentals of our proposal, a trusted access control, in Section 3; we demonstrate the CPN model for our proposal in Section 4; in Section 5, we present the related works, and we finish this paper presenting some final considerations in Section 6.

2 Background

This Section briefly describes the essential concepts we use, such as Policy Enforcement Point (PEP) and Trusted Execution Environment (TEE). Besides, we also describe the Coloured Petri Nets as we used this formalism to model our proposal and verify security behaviors.

2.1 Policy Enforcement Point (PEP) Proxy

A PEP Proxy works as a gatekeeper controlling the access to a protected service or resource, verifying and validating the requesters to allow or deny their accesses. Generally, a PEP acts together with some policy-based access management system, which can be a Policy Decision Point (PDP) or an Identity and Access Management (IAM) system.

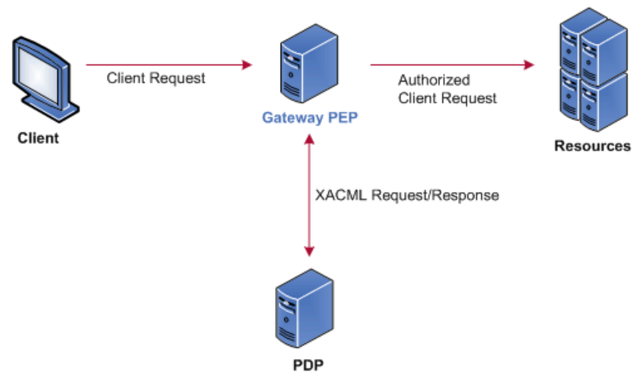

Fig. 1: Architecture with a Policy Enforcement Point [17].

Fig. 1 shows a basic architecture using a PEP. When the PEP proxy/gateway receives a request, it checks the access policies with the PDP or IAM system to determine whether the requester has permission to access the protected resources. Since the PEP enforces the access policies application, it forwards the request to the protected service or sends an unauthorized message to the requester, respectively, whether the access was allowed or denied. An IAM is also known as IdM (Identity Management system).

2.2 Trusted Execution Environment and Intel SGX

Despite the many definitions found in the literature, a TEE can be defined as a processing environment, tamper resistant, that runs on separated kernel space [19, 25, 26]. Besides guaranteeing an isolated execution, TEE also provides secure storage and an attestation mechanism that proves its trustworthiness for third parties [30, 22].

Software Guard Extensions (SGX) is the TEE implementation from Intel. SGX is a set of instructions and mechanisms to provide data confidentiality and integrity. Sensitive data and code are placed in a container called *enclave*, which is a cipher memory area protected by the CPU against external code, even the BIOS, kernel, or hypervisor code [30].

SGX has a reduced Trusted Computing Base (TCB) involving a piece of hardware and software. The developer chooses which software parts will be placed inside the enclave and that data are protected even if the operating system or hypervisor is compromised. The communication between trusted and untrusted parts of the application is performed by the edge routines, which are defined by the developer [7].

Each enclave has a self-signed certificate by the enclave author, which contains information that allows the SGX to detect if any part of the enclave was tampered with and allows an enclave to prove that it was correctly loaded and can be trusted. The Enclave Measurement computation is a unique 256 bits hash code, which identifies the code and initial data to be placed within the enclave, the order and position in which they are to be placed, and the security properties of those pages. A change in any of these variables will result in a different measure. [7].

To enable two or more enclaves to share data securely, the SGX also supports Remote Attestation (RA), which is the process of proving that software was properly initialized on a TEE platform. In the case of Intel SGX, RA is the mechanism by which a third party confirms that a particular software entity is running on an Intel SGX-enabled platform and protected within an enclave before provisioning that software with secrets and protected data.

2.3 Coloured Petri Nets

CPN is a graphical and formal language for modeling and validating concurrent systems [11]. CPN extends the classical Petri nets with a functional programming language (i.e., CPN ML language). A CPN model includes elements such as places (graphically represented by an ellipse), transitions (graphically represented by a rectangle), arcs, color sets, variables, and tokens. Modelers can only connect places to transitions and transitions to places using arcs (never places to places and transitions to transitions). The distribution of tokens among all places represents one marking of a CPN model. The state space of a model contains the set of all reachable markings from the initial one. Readers can use the book of Jensen and Kristensen [10] for more technical details on the syntax and semantics of CPN.

The CPN Tools software provides features to develop and formally analyze CPN models. For instance, CPN Tools enables automatic state space analysis using standard and user-defined query functions based on the CPN ML language. Besides, using a specific library (i.e., ASK-CTL) to conduct model checking [5], modelers can verify and ensure that a CPN model satisfies temporal modal logic formulas.

3 Trusted PEP Proxy

As stated before, although a PEP Proxy can be deployed to protect a sensitive data service, nothing can be inferred about the security of the requester machine, which processes the sensitive data. This Section explains our proposal to improve the protection provided by a PEP Proxy, allowing access to protected service only to TEE applications, which securely process sensitive data.

3.1 Application Scenario

A smart metering application is a scenario considered as the base case for this work. In this case, residences have smart meters responsible for measuring energy consumption and sending this information to be processed by the energy supply company. This application can be divided into producers and consumers: producers are the smart meters that generate energy consumption data, and consumers are the applications responsible for processing these data.

As seen before, this kind of application is susceptible to NIALM attacks, which hit the residents' privacy once their behavior can be inferred by someone that analyzes the residences' energy consumption in a specific period.

3.2 Threat Model

As the case base application contains three main components (producers, data storage service, and consumers), the attack surface considers data in transit between any of these components, data in rest inside the data storage service, and data in processing, mainly in the consumers. This work's focus is the improvement of the PEP Proxy. Thus we consider the data are encrypted in the producers and decrypted in the consumers, with the cryptographic keys exchange process out of scope (not described here). With this deliberation, the attack surface for this work considers only the consumers since the main worry is regarding the data to be processed once consumers can be a target of attacks, and attackers can get access to the sensitive data.

3.3 Design Principles

As a standard PEP Proxy works validating just if a specific requester is authorized to access a specific service, we decided to improve in a way the data owner could decrease the concern regarding possible attacks in the requester machine during data processing. For this, the application requires that the data service be accessed only by requesters running TEE applications, ensuring a better security level for data during processing.

Thus, the PEP Proxy should also validate if the requester runs a TEE application, which will perform secure data processing. To accomplish this desired improvement, the PEP Proxy should execute the remote attestation (RA) process to check if the requester runs a valid TEE application. To validate our idea, we created a Coloured Petri Net (CPN) model representing all three main components (i.e., producer, protected data service, and consumer) and the communication flow between them. The CPN model also considers the authentication, authorization, and attestation process at a high level. With this model, we can verify security properties by simulating scenarios where consumers run and do not run with TEE, for instance.

3.4 Trusted PEP Proxy (TRUPP)

Since the improved PEP Proxy verifies if the requester runs on a TEE, allowing only trusted applications to get access to the protected data service, we call it Trusted PEP Proxy (TRUPP). Fig. 2 shows an architecture with the deployment of the TRUPP.

The communication flow between the components seen in Fig. 2, which is enumerated, is described below:

1. The service requester, which is supposed to run a TEE application (inside an Intel SGX enclave, for instance), authenticates with the Identity Management (IdM) System, informing valid credentials;
2. After the authentication process, validating the service requester credentials, the IdM System returns a valid OAuth2 token, which will be used later to check authorization (access permissions);
3. The service requester sends a request to the protected service, which is intercepted by the TruPP, informing the OAuth2 token received from the IdM System after the authentication process;

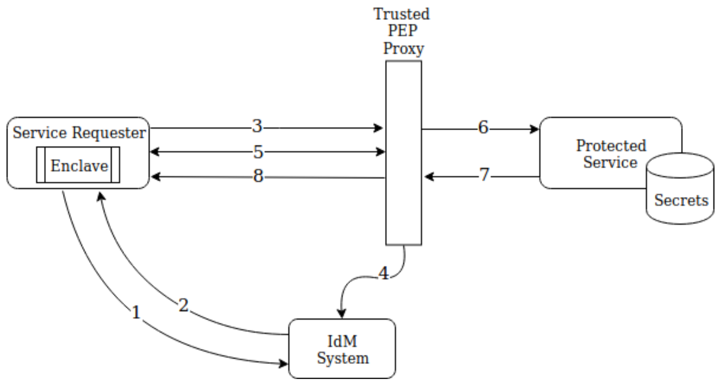

Fig. 2: Architecture with the Trusted PEP Proxy

4. The TruPP verifies the OAuth2 token with the IdM System, checking if the service requester has the right permissions to access the protected service;
5. Once the service requester has access permissions to the protected service, the TruPP starts the RA process to verify if the service requester runs a TEE application;
6. After the RA process is finished, if it succeeded, the request reaches the protected service, which will process the return;
7. The protected service returns the requested data to the TruPP;
8. The TruPP forwards the requested data received from the protected service.

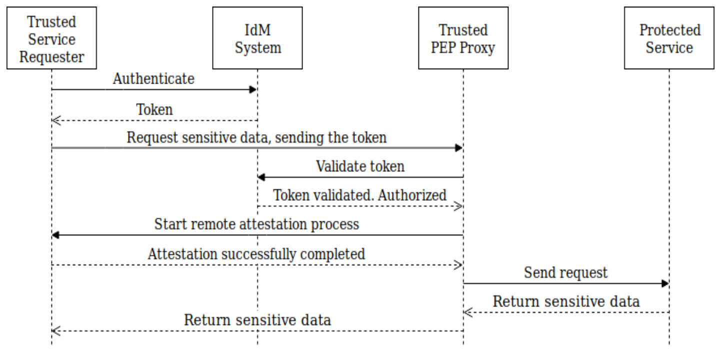
Trusted Service Requester
IdM System
Trusted PEP Proxy
Protected Service
Authenticate
Token
Request sensitive data, sending the token
Validate token
Token validated. Authorized
Start remote attestation process
Attestation successfully completed
Send request
Return sensitive data
Return sensitive data

Fig. 3: Message sequence diagram

The main difference between a standard PEP Proxy and the improved TruPP is step 5, which is the execution of the RA process, checking for a TEE application in the service requester. We can see the message sequence diagram of this communication flow in Fig. 3. As explained, the trusted service requester authenticates in the IdM System and receives a valid OAuth2 token. Then, it sends a sensitive data request to the protected service, which the TruPP intercepts. The TruPP then validates the OAuth2 token with the IdM System and starts the remote attestation process. Once the trusted service requester is successfully attested, the TruPP sends the request to the protected service and receives back the sensitive data requested.

Finally, the TruPP forwards the sensitive data to the trusted service requester, which will securely perform the processing (inside an enclave, i.e., a trusted application).

4 Formal Model and Analysis

We used the CPN formal modeling language to represent and validate the behaviors of our solution. Fig. 4 presents the main module of our hierarchical CPN model. The model comprises four sub-modules: *Service Requester with TEE Protection*, *Trusted PEP Proxy*, *IdM System*, and *Protected Service*. The marking of the *Entity* place represents that there is one valid entity to request sensitive data and one fake entity that should not be able to consume sensitive data.

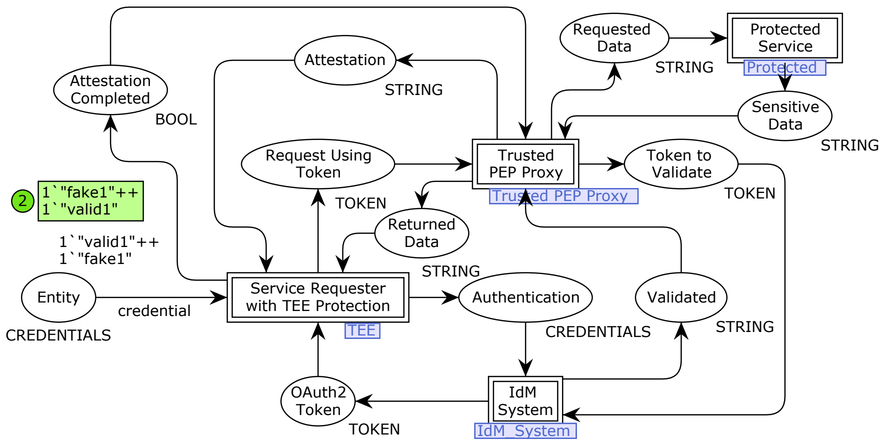

Fig. 4: Main module of the hierarchical CPN model.

Fig. 5 presents the sub-module related to the substitution transition *IdM System*. The IdM system maintains valid identities to enable the validation of credentials. Besides, the system validates the token received from the trusted PEP proxy. The other sub-modules are detailed representations of the solution presented in the previous section (e.g., see Fig. 3). We omitted the remaining sub-modules due to space constraints.

To increase confidence in our solution, we formally verified the model using the state-space analysis tools available from the CPN Tools software. We considered the initial marking of the model that represents one valid entity able to request sensitive data and a list of valid identities registered in the IdM system. Fig. 6 shows two examples of CPN ML functions used to analyze if the model presents desired behaviors, protecting the consumption of sensitive data. CPN ML is a functional programming language defined based on the ML standard and embedded in the CPN Tools software. We defined the CML ML function of Fig. 6a to verify if the consumption of sensitive data is denied when a valid entity sends a request without the existence of a valid TEE (i.e., the entity failed the attestation process). The standard CPN ML function *PreAllNodes* conducts an exhaustive search throughout the state space, verifies the predicate (represented using a new CPN ML function defined by a modeler) for each state, and returns the states satisfying the predicate.

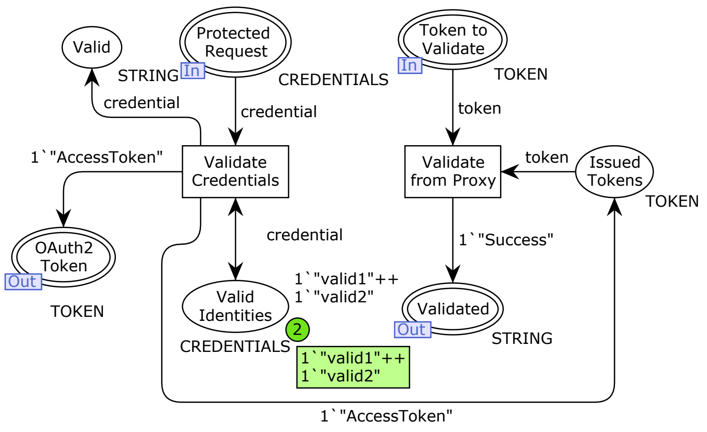

Fig. 5: Sub-module of the hierarchical CPN model, representing the IdM system.

We also defined the CPN ML function of Fig. 6b to analyze the remaining states in which a valid entity requests and consumes sensitive data after passing the attestation process (i.e., the entity can consume sensitive data).

```
fun failedOrReadyForAttestation n =
 let
    val valid = Mark.IdM_System'Valid 1 n
    val result = Mark.Trusted_PEP_Proxy'Result_Att 1 n
    val received = Mark.TEE'Received 1 n
 in
    (valid == 1`"valid1") andalso (result == empty)
     andalso (received == empty)
 end;

val invalid = PredAllNodes (fn n =>
                             failedOrReadyForAttestation n)
```

(a)

```
fun canConsume n =
 let
    val valid = Mark.IdM_System'Valid 1 n
    val result = Mark.Trusted_PEP_Proxy'Result_Att 1 n
    val received = Mark.TEE'Received 1 n
 in
    (result == 1`true) andalso (valid == 1`"valid1")
 end;

val invalid = PredAllNodes (fn n => canConsume n)
```

(b)

Fig. 6: Examples of CPN ML functions for verification of desired behaviors

After running both functions, we observed that all states are covered because, from the 13 existing states, nine represent that the entity is ready for attestation or failed the attestation. Besides, five states represent the entity that passed the attestation and is prepared to consume (or have already consumed) sensitive data. Although we only consider a unique, valid entity in our initial marking of the model (which leads to only 13 states), it is enough to formally analyze the expected behaviors for protecting sensitive data. Adding more valid or fake entities would only increase the number of states in the state space without negatively impacting our previous analysis. Our model also guarantees that only a valid entity can request sensitive data for consumption.

5 Related Work

In this section, we present some works that apply Policy Enforcement Point (PEP) Proxy and Trusted Execution Environment (TEE) to improve the security of sensitive data applications.

Bruno et al. [4] used a PEP Proxy written in Python to improve access control in IoT applications that use the MQTT protocol. PEP Proxy acts between IoT nodes and the MQTT broker, intercepting the messages sent from the nodes to the broker and from the broker to the nodes. A Policy Decision Point (PDP) checks all the requests. The authors performed an experimental evaluation and concluded that their solution presents an overhead of 100ms, considered a viable performance. Achillas et al. [1] used the FIWARE Wilma PEP Proxy to control access in a fleet management system responsible for optimizing task and time management in agricultural material handling operations. Valadares et al. [24] also applied the Wilma PEP Proxy to control access of data producers and consumers to protected services. Only authorized producers and consumers have access to a data broker and a key distribution system in their solution.

Shaghaghi et al. [21] proposed PEP as a Service (PEPS), aiming to protect applications in a Software Defined Network (SDN) domain. The authors implemented a prototype and reported an analysis regarding the challenges and performance of the PEPS, suggesting improvements for future work. Al-Hasnawi et al. [2] presented a Policy Enforcement Fog Module (PEFM), which aims to protect data in fog-based IoT applications. The PEFM employs a PEP and is positioned between fog servers and IoT sensors, controlling data access through the enforcement of privacy policies. They evaluated the PEFM in different scenarios (local and remote) through simulations, and the results showed acceptable performance overhead. Shahraki et al. [20] proposed a decentralized multi-authority attribute-based access control (DMA-ABAC) model to protect healthcare data. The DMA-ABAC uses a PEP too, and enables authorities to control their security settings directly. The proposal aims to resist against replay and third-party storage attacks and also attribute collusion, allowing users to access healthcare data according to security regulations.

Condé et al. [6] used the Intel SGX to implement a novel file protection scheme, protecting authentication credentials in a common authentication framework (PAM). According to the authors, tests with the implemented proposal presented acceptable performance overhead. Will et al. [29] propose the use of TEE to enforce the security of IPC mechanisms, creating a trusted communication channel between the applications and ensuring the confidentiality and integrity of data exchanged between them. Valadares et al. [24] proposed using TEE to establish a trusted architecture for IoT data dissemination with protection, the Trusted IoT Architecture (TIoTA). In the TIoTA, two components run on TEEs: a key distribution system and the data consumer. They conducted tests with a TIoTA implementation that uses Intel SGX, resulting in a low latency overhead (3ms on average). A privacy-preserving IoT data aggregation scheme that ensures data and user privacy in the processing of heterogeneous data and in the presence of heterogeneous devices is proposed by [28].

Nguyen et al. [16] proposed a distributed, scalable, fault-tolerant, and trusted logger for IoT device data using Intel SGX. This proposal, called LogSafe, achieved data confidentiality, integrity, and availability, providing tamper detection and protecting against replay, injection, and eavesdropping attacks. Through experiments, LogSafe presented high scalability and a high data transmission rate. Ayoade et al. [3] presented a decentralized system for data management in IoT applications using Intel SGX and blockchain, enforcing access control with smart contracts and storing the raw data in an SGX application, while the data hashes are stored in the blockchain. According to performed experiments with an implementation of the proposal, the authors conclude that it has an acceptable efficiency.

6 Conclusion

Access control mechanisms are present in many systems to allow access only to authorized parties/entities. A PEP Proxy is often used to intercept requests and verify if the requesters have permission to access the protected data/service by checking tokens, policies, or other attributes. Generally, a PEP Proxy validates if the requester is authenticated and has the authorization to access what is requested.

In this work, we proposed an improvement to PEP Proxies: adding the remote attestation protocol to verify if the requester runs in a TEE, guaranteeing that data will be processed more securely, warranted by the TEE capabilities and protections. This way, only authenticated and authorized requesters running in a TEE can access the protected data/services.

To validate our proposal, we modeled a Coloured Petri Net representing the architecture with the main entities: a trusted service requester, an IdM System, a Trusted PEP Proxy, and a protected service. The CPN model considered the communication flow among the entities and the main operations (authentication, authorization, and attestation). With this model, we verified that the proposal behaves as desired, keeping the security properties and improving the data security provided by the PEP Proxy.

For future work, we suggest implementing a PEP Proxy in a TEE application, ensuring that it processes the requests with more security, and also adding timestamps to attested requesters, with an expiration time, to identify when the PEP should perform a new attestation. Besides, we plan to perform model checking of security properties with the CPN model and a few experiments to measure the time and CPU overhead of the PEP improvements.

References

1. Achillas, C., Bochtis, D., Aidonis, D., Marinoudi, V., Folinas, D.: Voice-driven fleet management system for agricultural operations. Inf. Process. Agric. **6**(4) (2019)
2. Al-Hasnawi, A., Carr, S.M., Gupta, A.: Fog-based local and remote policy enforcement for preserving data privacy in the internet of things. Internet of Things **7** (2019)
3. Ayoade, G., Karande, V., Khan, L., Hamlen, K.: Decentralized IoT data management using blockchain and trusted execution environment. In: Intl Conf on Information Reuse and Integration. IEEE, Salt Lake City, UT, USA (2018)
4. Bruno, E., Gallier, R., Gabillon, A.: Enforcing access controls in IoT networks. In: Intl Conf on Future Data and Security Engineering. Springer, Nha Trang City, Vietnam (2019)
5. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R.: Handbook of Model Checking. Springer, 1st edn. (2018)
6. Condé, R.C.R., Maziero, C.A., Will, N.C.: Using Intel SGX to protect authentication credentials in an untrusted operating system. In: Sym on Computers and Communications. IEEE, Natal, RN, Brazil (2018)
7. Costan, V., Devadas, S.: Intel SGX explained. IACR Cryptology ePrint Archive **2016** (2016)
8. Henze, M., Hermerschmidt, L., Kerpen, D., Häußling, R., Rumpe, B., Wehrle, K.: A comprehensive approach to privacy in the cloud-based internet of things. Future Gener. Comput. Syst. **56** (2016)
9. Hu, P., Dhelim, S., Ning, H., Qiu, T.: Survey on fog computing: Architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. **98** (2017)
10. Jensen, K., Kristensen, L.M.: Coloured Petri Nets. Springer (2009)

11. Jensen, K., Kristensen, L.M.: Colored Petri nets: A graphical language for formal modeling and validation of concurrent systems. Commun. ACM **58**(6) (2015)
12. Kortuem, G., Kawsar, F., Sundramoorthy, V., Fitton, D.: Smart objects as building blocks for the internet of things. IEEE Internet Comput. **14**(1) (2010)
13. Küçük, K.A., Paverd, A., Martin, A., Asokan, N., Simpson, A., Ankele, R.: Exploring the use of Intel SGX for secure many-party applications. In: Wksp on System Software for Trusted Execution. ACM, Trento, Italy (2016)
14. Lee, K.: Despite risks, healthcare IT professionals stick with mobile (2018), https://searchhealthit.techtarget.com/feature/Despite-risks-healthcare-IT-professionals-stick-with-mobile
15. McLaughlin, S., McDaniel, P., Aiello, W.: Protecting consumer privacy from electric load monitoring. In: Conf on Computer and Communications Security. ACM, Chicago, IL, USA (2011)
16. Nguyen, H., Ivanov, R., Phan, L.T.X., Sokolsky, O., Weimer, J., Lee, I.: LogSafe: Secure and scalable data logger for IoT devices. In: Intl Conf on Internet-of-Things Design and Implementation. IEEE, Orlando, FL, USA (2018)
17. Oracle: XACML Policy Enforcement Point (2019), https://docs.oracle.com/cd/E27515_01/common/tutorials/authz_xacml_pep.html
18. Rouse, M., Sutner, S.: PHI breach (protected health information breach) (2018), https://searchhealthit.techtarget.com/definition/PHI-breach-protected-health-information-breach
19. Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted execution environment: What it is, and what it is not. In: Intl Conf on Trust, Security and Privacy in Computing and Communications. IEEE, Helsinki, Finland (2015)
20. Salehi Shahraki, A., Rudolph, C., Grobler, M.: A dynamic access control policy model for sharing of healthcare data in multiple domains. In: Intl Conf on Trust, Security and Privacy in Computing and Communications. IEEE, Rotorua, New Zealand (2019)
21. Shaghaghi, A., Kaafar, M.A., Scott-Hayward, S., Kanhere, S.S., Jha, S.: Towards policy enforcement point as a service (PEPS). In: Conf on Network Function Virtualization and Software Defined Networks. IEEE, Palo Alto, CA, USA (2016)
22. Shepherd, C., Arfaoui, G., Gurulian, I., Lee, R.P., Markantonakis, K., Akram, R.N., Sauveron, D., Conchon, E.: Secure and trusted execution: Past, present, and future - a critical review in the context of the internet of things and cyber-physical systems. In: Intl Conf on Trust, Security and Privacy in Computing and Communications. IEEE, Tianjin, China (2016)
23. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet Things J. **3**(5) (2016)
24. Valadares, D.C.G., da Silva, M.S.L., Brito, A.M.E., Salvador, E.M.: Achieving data dissemination with security using FIWARE and Intel Software Guard Extensions (SGX). In: Sym on Computers and Communications. IEEE, Natal, RN, Brazil (2018)
25. Valadares, D.C.G., Will, N.C., Caminha, J., Perkusich, M.B., Perkusich, A., Gorgônio, K.C.: Systematic literature review on the use of trusted execution environments to protect cloud/fog-based internet of things applications. IEEE Access **9** (2021)
26. Valadares, D.C.G., Will, N.C., Spohn, M.A., de Souza Santos, D.F., Perkusich, A., Gorgônio, K.C.: Confidential computing in cloud/fog-based internet of things scenarios. Internet of Things **19** (2022)
27. Wang, J., Hong, Z., Zhang, Y., Jin, Y.: Enabling security-enhanced attestation with Intel SGX for remote terminal and IoT. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **37**(1) (2018)
28. Will, N.C.: A privacy-preserving data aggregation scheme for fog/cloud-enhanced IoT applications using a trusted execution environment. In: Intl Systems Conf. IEEE, Montreal, QC, Canada (2022)
29. Will, N.C., Heinrich, T., Viescinski, A.B., Maziero, C.A.: Trusted inter-process communication using hardware enclaves. In: Intl Systems Conf. IEEE, Vancouver, BC, Canada (2021)
30. Will, N.C., Valadares, D.C.G., de Souza Santos, D.F., Perkusich, A.: Intel Software Guard Extensions in internet of things scenarios: A systematic mapping study. In: Intl Conf on Future Internet of Things and Cloud. IEEE, Rome, Italy (2021)