Computers & Security 121 (2022) 102866

Contents lists available at ScienceDirect

Computers & Security

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

Footsteps in the fog: Certificateless fog-based access control

Eugene Frimpong [a,*], Antonis Michalas [a,c], Amjad Ullah [b,c]

[a] Tampere University, Korkeakoulunkatu 7 Kampusareena, Tampere, Finland
[b] Edinburgh Napier University, Edinburgh, Scotland, United Kingdom
[c] University of Westminster, London, United Kingdom

ARTICLE INFO

Article history:
Received 17 July 2021
Revised 22 June 2022
Accepted 31 July 2022
Available online 1 August 2022

Keywords:
Access control
Fog computing
Attribute-based access control
Internet of things
Certificateless cryptography

ABSTRACT

The proliferating adoption of the Internet of Things (IoT) paradigm has fuelled the need for more efficient and resilient access control solutions that aim to prevent unauthorized resource access. The majority of existing works in this field follow either a centralized approach (i.e. cloud-based) or an architecture where the IoT devices are responsible for all decision-making functions. Furthermore, the resource-constrained nature of most IoT devices make securing the communication between these devices and the cloud using standard cryptographic solutions difficult. In this paper, we propose a distributed access control architecture where the core components are distributed between fog nodes and the cloud. To facilitate secure communication, our architecture utilizes a Certificateless Hybrid Signcryption scheme without pairing. We prove the effectiveness of our approach by providing a comparative analysis of its performance in comparison to the commonly used cloud-based centralized architectures. Our implementation uses Azure – an existing commercial platform, and Keycloak – an open-source platform, to demonstrate the real-world applicability. Additionally, we measure the performance of the adopted encryption scheme on two types of resource-constrained devices to further emphasize the applicability of the proposed architecture. Finally, the experimental results are coupled with a theoretical analysis that proves the security of our approach.

1. Introduction

Access control (AC) is a fundamental component in the IoT paradigm where authorized access to potentially hundreds of thousands of devices is of utmost importance (Servos and Osborn, 2017). The rapid rise in the deployment of IoT systems has led to increased interest from academia and industries in effectively integrating access control systems in IoT systems. It is expected that there will be roughly 36 billion IoT devices in 2025 and 50 billion devices by 2030 (Research, 2020). Most of the attention directed at the IoT computing paradigm stems from the promise of potentially providing users, communities, and industries with the capability to harvest substantial amounts of data, whilst providing low-costs and low-latency communication. This, however, has inadvertently led to a significant rise in security and privacy related concerns in the use of IoT (Sandhu, 2000), especially in the area of AC (Michalas et al., 2011a; 2011b; Sandhu, 2000). Although a lot of interest is being garnered in this area, majority of the current works have been focusing on the theoretical aspect of design and enforcement of different access control models (Alshehri and Sandhu, 2017; Ouaddah et al., 2017; Ravidas et al., 2019; Salonikias et al., 2016). Demonstrating the feasibility of conducted studies requires directing attention at providing implementation results.

Our analysis of existing AC schemes for the IoT highlighted two main approaches. First, a cloud-based (centralized) approach where all components of the AC system run in a remote cloud server (Alshehri and Sandhu, 2017; Ouaddah et al., 2017; Ravidas et al., 2019). Second, a decentralized approach where all components of the AC system are implemented in the IoT devices (Ravidas et al., 2019; Tasali et al., 2017). Both approaches have their intrinsic problems. A completely centralized approach, introduces substantial communication overhead and a single point of failure, while in a decentralized one, implementing a complete AC system in an IoT device is challenging, as it requires substantial computational power which is usually unavailable in these IoT devices. In both approaches, achieving a scalable IoT environment, an important requirement for IoT applications (Ravidas et al., 2019), becomes challenging. Finally, most commercial implementations of AC for IoT frameworks enforce coarse-grained access control policies which offer limited functionalities to administrators and users.

* corresponding author.
E-mail addresses: eugene.frimpong@tuni.fi (E. Frimpong), antonios.michalas@tuni.fi (A. Michalas), a.ullah@napier.ac.uk (A. Ullah).

https://doi.org/10.1016/j.cose.2022.102866

Our Contribution: In this work, we propose a distributed access control architecture that distributes the core components of an AC system between fog computing nodes and the cloud. Fog computing is generally described as a virtualized platform that provides services such as computing, storage, networking and control closer to edge devices (Frimpong et al., 2020). The key advantages of using fog computing nodes are reduced latency overhead, reliable operation and eliminating the need for devices to continually connect to the centralized cloud. Our contributions can be summarized as follows;

- We propose a distributed access control architecture centred around the use of fog computing nodes in tandem with cloud computing nodes. Our architecture is not entirely reliant on a central authorization authority for access decisions, and uses the popular Attribute-Based Access Control (ABAC) model (Hu et al., 2013).
- We strengthen our architecture by utilizing a certificateless hybrid signcryption scheme to secure the communication between the IoT devices and fog nodes/users.
- Subsequently, we introduce the Footsteps in the Fog (FitF) protocol to demonstrate the practical implementation of our approach, and identify real-world applications that can benefit from our proposed system.
- Finally, we provide an experimental test bed that realistically imitates both the commonly used cloud-based centralized approach and our proposed architecture, and conduct various experiments as well as a comparative analysis of the efficiency of both approaches.

Organization

The rest of the paper is organized as follows: In Section 2, we present the most important related works, while providing brief background information on access control system in Section 3, as well as discussing the proposed system model. Our protocol and architecture are then presented in Section 4, with detailed threat model and security analysis in Section 5. In Section 6, we evaluate the performance of the protocol, followed by a discussion of its application domain in Section 7, and an extended threat model in Section 8. In Section 9, we discuss some limitations of Attribute-Based Encryption, and finally conclude the paper in Section 10.

2. Related work

The concept of achieving secure access control via fog computing nodes has been well studied with studies on different approaches, challenges and state of the art in the field (Aleisa et al., 2020; Xu et al., 2020; 2021). For example, authors in Aleisa et al. (2020) recently outlined a number of challenges with access control in fog computing as well state of the art in fog-based access control. Authors in Xu et al. (2021) presented a secure-cloud-fog bilateral access control system based on a lightweight matchmaking encryption cryptographic tool. In this desgin, a sender can specify a decryption policy to control receivers while a receiver can also specify source identification policy to discard undesirable ciphertexts without costly data decryption. Additional schemes based on novel cryptographic techniques such as Attribute-Based Encryption (ABE) have also been highlighted as foundations for efficient access control architectures (Damgård et al., 2016; Zhang et al., 2018; Zhao et al., 2021). One such scheme is the work by authors in Zhao et al. (2021). In this work, authors proposed an efficient Ciphertext-Policy ABE (CP-ABE) scheme with attribute revocation capability in a fog enabled E-health ecosystem. This work focused primarily on the design of an efficient CP-ABE scheme that utilized fog computing as a method of reducing communication latency as compared to our work where we integrate the fog computing nodes as part of the access control process. Additional limitations of the ABE scheme are discussed and outlined in Section 9.

In further works, Salonikias et al. (2016), studied the operational characteristics of a proposed Intelligent Transportation System (ITS) (Selvarajah et al., 2012) paradigm which utilizes fog computing, and identify distinct access control issues. They proposed a theoretical ABAC-based AC system suitable for the fog computing paradigm. In this work, the Policy Enforcement Point (PEP) is implemented on the IoT device and multiple Policy Decision Points (PDP) are located in the fog with the Policy Information Point (PIP) in the cloud. Note that the authors implemented a PIP that performs the functions of both a traditional Policy Administration Point (PAP) and that of the PIP (i.e. the PIP contains subject attribute information as well as policy information). Contrary to this work, we propose an architecture that implements the core AC functions external to the IoT device while propagating contents of the PAP to components in fog nodes. Additionally, we provide experimental evaluations to show the feasibility of our proposal.

Alshehri and Sandhu (2016) also proposed an architecture for the deployment of an authoritative family of access control models in cloud-enabled IoT devices. Policy storage, enforcement and decision making are conducted in the cloud services layer, while policy administration is conducted in the application layer. This work has been further extended by Alshehri and Sandhu (2017), who developed operational and administrative access control models that support the described Access Control Oriented (ACO) architecture. These access control models were implemented under the assumption that the ACO architecture uses a topic-based publish/subscribe communication method (Johnsen and Bloebaum, 2012) for all interactions between entities. For their operational models, authors utilized a combination of Access Control Lists (ACL) (Barkley, 1997) and ABAC model while using another combination of ACLs, Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC) for their administrative models. While this work provided a comprehensive architecture for cloud-based IoT devices, it is purely theoretical and did *not* provide any experimental evaluation. Furthermore, researchers in Tasali et al. (2017) proposed a fine-grained access control based framework for Health-based IoT applications, which supports multiple authorization levels for the same device. However, as compared to Alshehri and Sandhu (2017, 2016), authors provide a proof of concept implementation focusing on the Medical Device Coordination Framework (MDCF) (Yadav and Meghashree, 2017). In their architecture, the components of the AC system are situated in the network controller component of the MDCF. Their framework, by utilizing ABAC, and implementing the core components of the authorization framework outside the IoT device, satisfied all but the performance and latency requirement for access control systems in IoT (Ravidas et al., 2019) – a pivotal requirement for such applications

Access Control-as-a-Service (ACaaS) was explored by Ahmad et al in their paper (Ahmad et al., 2018). Their system allowed for the specification and management of policies independent of the CSP while using the enforcement mechanisms offered by the CSP; it automatically analyses and validates policies and translates AC requirements into platform-independent entities and policies. The management of policies is outsourced to a trusted third-party while utilizing the enforcement mechanisms provided by the CSP. This implementation is entirely reliant on the CSP to make access decisions with no provisions for enforcing a secure communication between resource constrained edge devices as compared to our work where we limit the complete reliance on the CSP with added security measures. Finally, Blockchain technology has also been explored as a means of providing efficient access control systems (Gilani et al., 2020; Shafagh et al., 2017; Zyskind et al., 2015). In these approaches, access control policies are treated as

transactions immutably written on the blockchain and publicly auditable. Access rights can be granted based on data type, and access granted to the requester after being verified against the data in the blockchain.

3. Background and system model

In this section, we provide some background information on Access Control and Attribute-based Access Control model as well as a description of the system model considered for this paper.

Access Control System (ACS) An access control system typically consists of three core components (Ravidas et al., 2019): *(i)* the *policies*, which define and outline security requirements governing access to a given resource, *(ii)* the *model*, which serves as a formal representation of the defined policies, and *(iii)* the *mechanism*, a low-level implementation or enforcement of the specified security controls defined in the policy. Most existing ACS are based on the Object Model Architecture and Mechanism framework (OM-AM) (Sandhu, 2000). In OM-AM, the OM layers address the security objectives of an ACS, define the access control policy, the aim of the system, and interpretation of real-world security policies. The AM layers, deal with how to meet the set requirements in the OM layers and the interaction between the various entities, workflows and the low-level implementation and enforcement of the defined policies. *Our work focuses on the AM layers with particular emphasis on the ABAC model.*

The mechanism component of an ACS, consists of subcomponents that play various roles in the evaluation and enforcement of a specified policy. These are implemented in a variety of ways based on well-publicized methods (Ouaddah et al., 2017) such as policy-based, token-based, and hybrid architectures. In this work, we consider the policy-based architecture as described in the eXtensible Access Control Markup Language (XACML) standard (Oasis, 2013). Our architecture (Fig. 1) consists of the following components: *(i)* Policy Enforcement Point (PEP) – responsible for enforcing any policy-based decision, *(ii)* Policy Decision Point (PDP) – evaluates all access requests against defined access control policies and makes a decision on whether to grant or deny access, *(iii)* Policy Administration Point (PAP) – provides the policy repository and facilitates the management of policies, and iv) Policy Information Point (PIP) – provides subject and object information repository as well as information to the PDP to enable policy evaluation.

High level Architecture Overview When a user or application attempts to access a resource, the PEP sends information on the attempted access to the PDP. This information is in the form of an Authorization Decision Request (ADR). The PDP evaluates the ADR against policies and attributes available to it and returns an Authorization Decision (AD) to the PEP. PEP's main task, is to enforce any AD returned from the PDP. During the process of sending information on the attempted access to the PDP, the PEP may obtain attributes of both subject and device from online Attribute Authorities (AA) or Attribute Repositories (AR) into which AAs have stored attributes. Additionally, the PDP augments the information provided by the PEP, with more attributes obtained from the AAs or ARs. Subsequently, the PDP obtains policy information from PAP or Policy Repositories (PR) into which PAP has stored an arbitrary number of policies (Fig. 1).

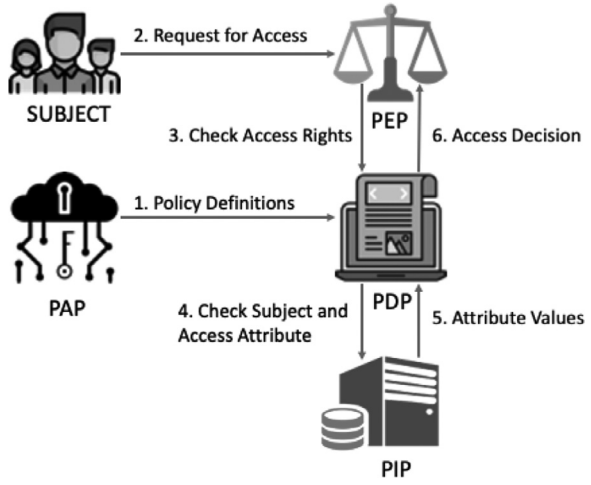


Fig. 1. Policy-based Architecture Diagram.

Attribute-Based Access Control (ABAC) ABAC is an access control model where a subject's request to access a resource is granted or denied based on the assigned attributes of subject, object, environmental conditions and a set of policies specified in terms of the attributes and conditions (Hu et al., 2014). ABAC supports the definition of policies based on existing attributes of both users and objects without the need to manually assign new attributes.

The attributes in the ABAC model can be classified into: *(i)* **User Attributes** – all attributes related to a user (e.g. name, role, date hired, relationship to another user etc.), *(ii)* **Device Attributes** – all aspects of a device (e.g. device metadata, serial number, last synchronized, device owner, etc.), and *(iii)* **Contextual Attributes** – all derived attributes (e.g. time of day, number of users accessing the device, physical location, current state of the device, etc.).

The Access Decisions in ABAC are a result of the evaluation of boolean statements that are the translations of real world policies. The actual decision relies on the value of attributes at a specific point in time. The administrator, who defines the policy, only needs to know a subset of the potential attributes of users, device and contextual information. This particular feature is what makes ABAC an ideal model for the IoT – it provides both fine-grained control and support for the heterogeneous nature of IoT deployments.

3.1. System model

Below, we provide a detailed description of the system model for our distributed access control architecture (Figs. 2 and 3). The setup consists of five primary entities: *(i)* Key Generation Center (KGC), *(ii)* IoT Sensor Devices, *(iii)* Fog Nodes, *(iv)* Users, and *(v)* Cloud Service Provider (CSP). **Key Generation Center (KGC):** This entity is responsible for generating the necessary public parameters and keys for the proper run of the security algorithms used by the registered entities. The KGC's role is limited to securing the communication medium between different entities.

Cloud Service Provider (CSP): We consider a top level cloud computing service equipped with a PAP and PIP. The contents of the PAP are pushed to the PR, while the contents of the PIP are pushed into the AR component of the fog nodes based on a set rule or schedule. The PIP is the subject and object information repository. System administrators and IoT device owners are able to access the CSP in order to set and define policies using an administrative interface. These policies are stored in the PAP, while PIP, operates as a persistent medium, storing all attributes related to users and devices. Additionally, the CSP is responsible for propagating the contents of the PAP and PIP to fog nodes in its domain. Communication between all entities for the propagation and synchronization of attributes and policies is secured via TLS (Fig. 3).

End Users: Let $\mathcal{U} = \{u_1, \ldots, u_x\}$ be the set of end users registered in our environment. Each user is allowed to perform various computations, generate access requests and receive status updates on sensed events. Each user possesses a list of attributes that is evaluated during an access request. This list is generated dur-

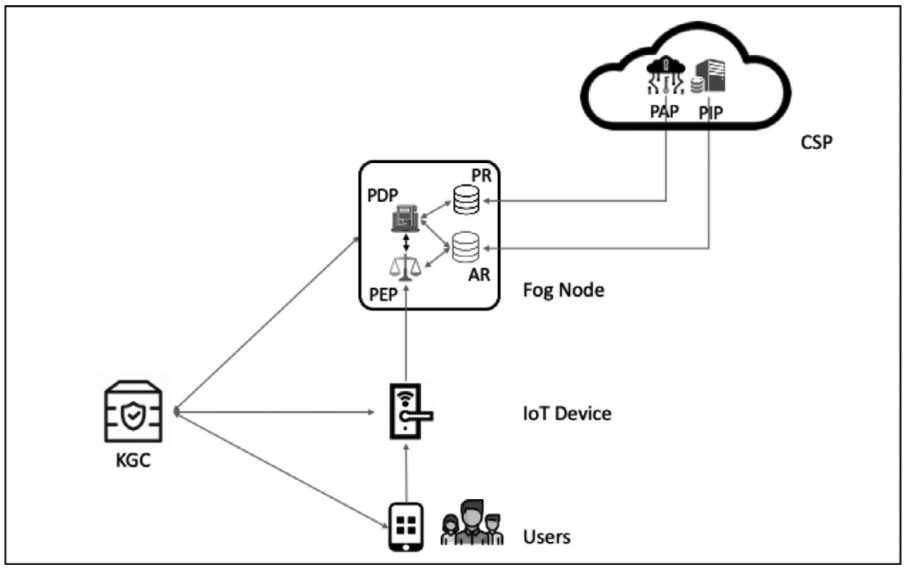


Fig. 2. Proposed Architecture.

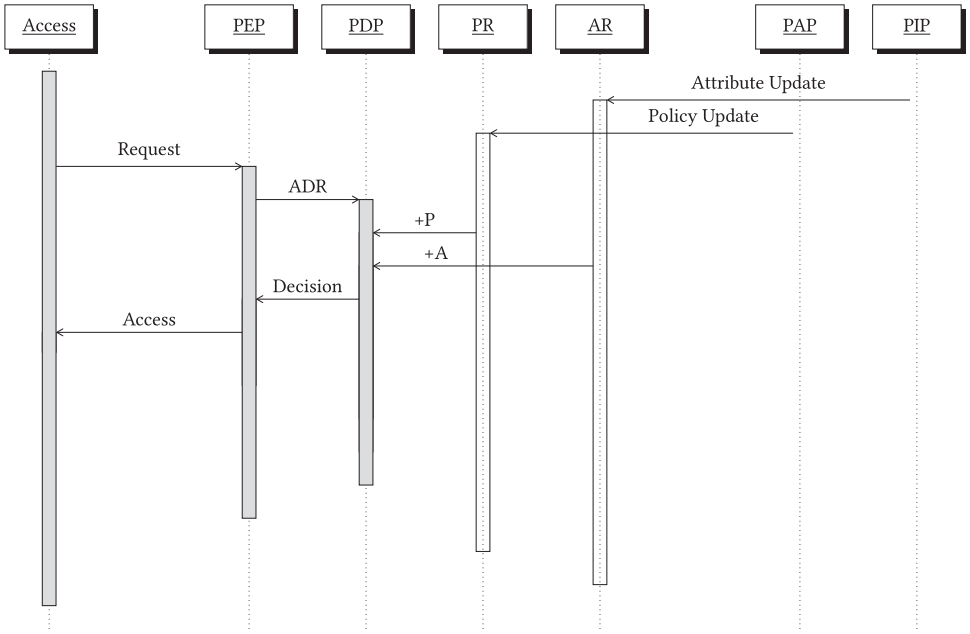


Fig. 3. Access Control Process.

ing a registration phase and can be updated on demand. A user's attributes are assigned in the form of a *User Attribute Assignment (UAA)* (Servos and Osborn, 2017):

$$\{a \in \mathcal{A}, u \in \mathcal{U}, v \in \mathcal{V}\} \in UAA$$

where $\mathcal{A}$ is the attributes space and $\mathcal{V}$ is the set of all values assigned to a given user and attribute pair.

IoT Sensor Devices: Let $\mathcal{D} = \{d_1, \ldots, d_y\}$ be the set of all IoT devices deployed in our setup. An element of $\mathcal{D}$, can either register the occurrence of various environmental events or provide users access to specific resources. Furthermore, a device is assigned attributes in the form of a *Device Attribute Assignment (DAA)* (Servos and Osborn, 2017):

$$\{a \in \mathcal{A}, d \in \mathcal{D}, v \in \mathcal{V}\} \in DAA$$

When a device $d_j \in \mathcal{D}$ receives an access request from a user u_i, it establishes a secure communication session with the PEP component of a nearby fog node.

Fog Nodes: Let $\mathcal{F} = \{f_1, \ldots, f_z\}$ be the set of all fog nodes deployed in our setup. Each fog node is equipped with a PEP, PDP, PR and AR (Figs. 2 and 3). The PDP retrieves a list of attributes and defined access policies from AR and PR respectively. When the PEP receives an access request and attributes bundle from a device d_j, it generates an ADR which contains the user, device and contextual attributes, and forwards it to the PDP for an access decision. The PDP retrieves an additional list of attributes relating to both the device and user from the AR, and corresponding access policies from PR (+P and +A in Fig. 3).

4. Footsteps in the fog

In this section, we present Footsteps in the Fog (FitF) – a protocol that constitutes the core of this paper's contribution. FitF can be divided into two logically disjointed protocols: (i) the secure communication protocol, based on the CL-HSC scheme

presented in Seo and Bertino (2013), which allows registered entities in our architecture to securely communicate with one another, and (ii) the main distributed access control protocol, which utilizes fog nodes, and policy and attribute propagation from a CSP to achieve an efficient access control management system.

4.1. FitF secure communication

CL-HSC is a semantically secure certificateless hybrid signcryption scheme that allows us to provide a lightweight security solution to secure the communication between the resource constrained IoT device and fog nodes/users. We utilize CL-HSC in an effort to meet necessary security requirements such as authenticated key exchange and non-repudiation. Additionally, by eliminating the need to manage certificates, CL-HSC is suitable for use in our resource-constrained setting. The scheme comprises eight probabilistic algorithms as defined below:

1. **CL_Setup** : This algorithm is run by the KGC. It takes as input, a security parameter λ, and outputs a master secret key x and system parameters Ω.
   We denote this by: $(\mathsf{x}, \Omega) \leftarrow \mathsf{CL\_Setup}(\lambda)$.
2. **CL_GenSecretValue** : This algorithm is run by each registered user. It takes as input, Ω and the identity u_i, and outputs the user's secret value $\mathsf{x_i}$ and the corresponding public value $\mathsf{P_i}$. We denote this by: $(\mathsf{x_i}, \mathsf{P_i}) \leftarrow \mathsf{CL\_GenSecretValue}(\Omega, \mathsf{u_i})$.
3. **CL_PartialPrivKeyGen** : The KGC runs this algorithm to generate the user's partial private and public key pair. It takes as input Ω, x, and the identity of the user u_i, and outputs the user's partial key pairs $(\mathsf{s_i}, \mathsf{R_i})$.
   We denote this by: $(\mathsf{s_i}, \mathsf{R_i}) \leftarrow \mathsf{CL\_PartialPrivKeyGen}(\Omega, \mathsf{x}, \mathsf{u_i})$
4. **CL_PrivKeyGen** : Each registered user runs this algorithm to generate it's full private key. The algorithm takes as input the user's partial private key $\mathsf{s_i}$ and secret value $\mathsf{x_i}$, and returns the user's full private key $\mathsf{sk_i}$. We denote this by:
   $(\mathsf{sk_i}) \leftarrow \mathsf{PrivKeyGen}(\mathsf{s_i}, \mathsf{x_i})$
5. **CL_PubKeyGen** : Each registered user runs this algorithm to generate it's full public key. It takes as input the user's public key $\mathsf{P_i}$ and partial public key $\mathsf{R_i}$, and returns the user's full public key $\mathsf{pk_i}$. We denote this by: $(\mathsf{pk_i}) \leftarrow \mathsf{CL\_PubKeyGen}(\mathsf{P_i}, \mathsf{R_i})$
6. **CL_SymKeyGen** : This is a symmetric key generation algorithm run by a user u_i wishing to establishing a secure session with another user u_m. The algorithm takes as input, the sender's identity u_i, full public key $\mathsf{pk_i}$, full private key $\mathsf{sk_i}$, message recipient's identity u_m, and recipient's full public key $\mathsf{pk_m}$. It then returns the symmetric key $\mathsf{K_{im}}$ and an internal state information ω not known to u_m. We denote this by: $(\mathsf{K_{im}}, \omega) \leftarrow \mathsf{CL\_SymKeyGen}(\mathsf{u_i}, \mathsf{pk_i}, \mathsf{sk_i}, \mathsf{u_m}, \mathsf{pk_m})$
7. **CL_Encapsulation** : This is a key encapsulation algorithm run by the sender u_i to obtain an encapsulation φ. The algorithm takes as input, the internal state information ω corresponding to $\mathsf{K_{im}}$, an arbitrary tag τ, the sender's identity u_i, full public key $\mathsf{pk_i}$, and full private key $\mathsf{sk_i}$. The sender sends φ and τ to the recipient. We denote this by: $(\varphi) \leftarrow \mathsf{CL\_Encapsulation}(\omega, \tau, \mathsf{u_i}, \mathsf{pk_i}, \mathsf{sk_i})$
8. **CL_Decapsulation** : This is a key decapsulation algorithm run by the intended recipient of a message in order to retrieve the key $\mathsf{K_{im}}$ encapsulated in φ. The algorithm takes as input, the encapsulation φ, an arbitrary tag τ, the sender's identity u_i, full public key $\mathsf{pk_i}$, the recipient's identity u_m, full private key $\mathsf{sk_m}$ and full public key $\mathsf{pk_m}$. It then returns the symmetric key $\mathsf{K_{im}}$ or an invalid error. We denote this by: $(\mathsf{K_{im}}) \leftarrow \mathsf{CL\_Decapsulation}(\varphi, \tau, \mathsf{u_i}, \mathsf{pk_i}, \mathsf{u_m}, \mathsf{pk_m}, \mathsf{sk_m})$.

4.2. FitF access control management

The Access control management in FitF is divided into three main phases: *Setup and Initialization*, *Key Generation*, and *Access Request*. To provide a detailed description of each phase, we consider a scenario where a registered user u_i seeks to access a resource secured by an IoT device d_j. In the assumed scenario, there exists a nearby fog node f_k assigned to evaluate access requests and generate access decisions, and a KGC responsible for generating the public security parameters and assisting registered entities to generate the necessary keys. The user u_i and device d_j are assigned a list of attributes A_i and A_j, respectively, upon set-up and registration. Each phase is described below:

FitF - Setup and Initialization: The KGC runs the CL_Setup algorithm at the beginning of the protocol to generate a master secret key and system parameters. The algorithm returns the system parameters, Ω, and the master secret key x (Algorithm 1). These

Algorithm 1 CL_Setup.

Input: λ
Output: $\mathsf{x}, \Omega, \mathsf{P_{pub}}$

KGC
choose $x \in \mathbb{Z}^*$.
$P_{pub} = xP$
Output Ω to all entities

system parameters are public and accessible to each registered entity partaking in the protocol.

FitF - Key Generation: Each registered entity (i.e. the user u_i requesting access, the IoT device d_j, and the PEP component in the fog node f_k) runs the CL_GenSecretValue algorithm to generate a secret value (Algorithm 2). On successful run of this al-

Algorithm 2 CL_GenSecretValue.

Input: $\Omega, \mathsf{u_i}$
Output: $\mathsf{x_i}, \mathsf{P_i}$

for each entity **do**
  choose $x_i \in \mathbb{Z}^*$
  $P_i = x_i P$
  Send u_i and P_i to KGC

gorithm, each entity sends its identity and public key to the KGC for the generation of a *partial* private and pubic key pair. Upon receiving the identity of a registered entity the KGC runs the CL_PartialPrivKeyGen algorithm to output a partial private and public key pair for that entity (Algorithm 3). We assume that the

Algorithm 3 CL_PartialPrivKeyGen.

Input: $\mathsf{x}, \Omega, \mathsf{u_i}$
Output: $\mathsf{s_i}, \mathsf{R_i}$

for each entity **do**
  choose $r_i \in \mathbb{Z}^*$
  $R_i = r_i P$
  $s_i = r_i + xH_0(u_i, R_i, P_i) \mod q$
  return R_i, s_i

partial keys generated by the KGC are transferred securely to the respective entity.

Upon receiving the partial key pairs, each registered entity runs the CL_GenPrivKey and CL_GenPubKey to generate a full public/private key pair. CL_GenPrivKey returns the full private key

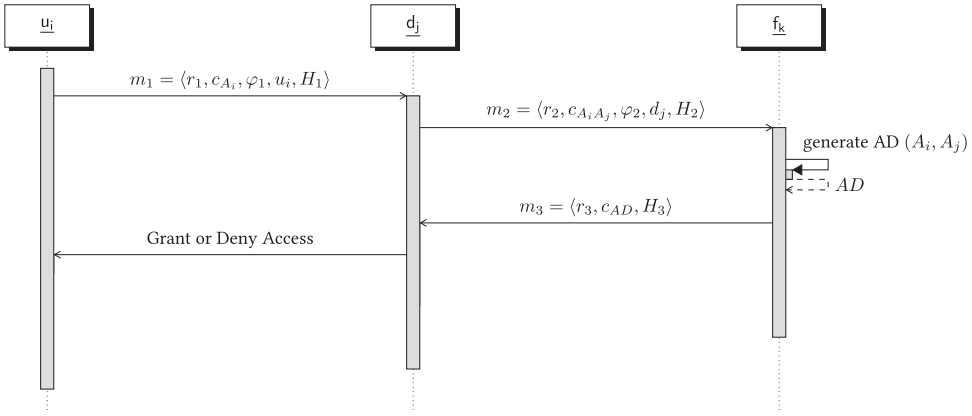


Fig. 4. FitF Access Request Phase.

Algorithm 4 CL_GenPrivKey.

Input: d, x
Output: sk

1: **for** each entity **do**
2: $sk_i = (s_i, x_i)$

(Algorithm 4), while CL_GenPubKey returns the full public key (Algorithm 5). We assume that each entity makes its full public

Algorithm 5 CL_GenPubKey.

Input: R, P
Output: pk

1: **for** each entity **do**
2: $pk_i = (R_i, P_i)$

key publicly available to all other entities partaking in the protocol.

FitF - Access Request: Access to a resource is granted or denied based on the access policy defined by the system's administrator (Fig. 4). To request access to a resource secured by d_j, the user u_i first generates a random number r_1, and runs the CL_SymKeyGen algorithm to generate a symmetric session key $\mathsf{K_{ij}}$. $\mathsf{K_{ij}}$ will be shared with d_j to secure the communication between these two entities (Algorithm 6). Using $\mathsf{K_{ij}}$, u_i encrypts the list of

Algorithm 6 CL_SymKeyGen.

Input: $\mathsf{u_i}$, $\mathsf{pk_i}$, $\mathsf{sk_i}$, $\mathsf{d_j}$, $\mathsf{pk_j}$
Output: $\mathsf{K_{ij}}$, ω

$\mathbf{u_i}$ Runs:
1: choose $l_i \in \mathbb{Z}^*$ at random.
2: $U = l_i \cdot P$.
3: $T = l_i \cdot H_0(d_j, R_j, P_j) \cdot P_{pub} + l_i \cdot R_j \mod q$.
4: $\mathsf{K_{ij}} = H_1(U, T, l_i \cdot P_j, d_j, P_j)$
5: $\omega = (l_i, U, T, u_i, pk_i, sk_i, d_j, pk_j)$
6: $\mathsf{c_{A_i}} = \mathsf{Enc}(\mathsf{K_{ij}}, \mathsf{a_i})$

its attributes A_i to obtain a ciphertext $\mathsf{c_{A_i}}$. To ensure that d_j can decrypt any ciphertext received from u_i, u_i uses CL_Encapsulation to encapsulate $\mathsf{K_{ij}}$ and obtains an encapsulated tag φ_1 (Algorithm 7). u_i then sends the following message to d_j: $m_1 = \langle r_1, c_{A_i}, \varphi_1, u_i, H_1 \rangle$ where $H_1 = H_{K_{ij}}(r_1 || c_{A_i} || A_i || \varphi_1)$. Upon receiving m_1, d_j executes the CL_Decapsulation algorithm to generate the symmetric session key $\mathsf{K_{ij}}$ (Algorithm 8). Finally, d_j decrypts $\mathsf{c_{A_i}}$ to obtain the list of u_i's attributes A_i and verifies the freshness and integrity of m_1.

Algorithm 7 CL_Encapsulation.

Input: ω, $\mathsf{c_{A_i}}$, $\mathsf{u_i}$, $\mathsf{pk_i}$, $\mathsf{sk_i}$, $\mathsf{pk_j}$
Output: φ

$\mathbf{u_i}$ Runs:
1: $H = H_2(U, c_{A_i}, T, u_i, P_i, d_j, P_j)$
2: $W = s_i + l_i \cdot H + x_i \cdot H$
3: $\varphi_1 = (U, W)$

Algorithm 8 CL_Decapsulation.

Input: φ, $\mathsf{c_{A_i}}$, $\mathsf{u_i}$, $\mathsf{pk_i}$, $\mathsf{d_j}$, $\mathsf{pk_j}$, $\mathsf{sk_j}$
Output: $\mathsf{K_{ij}}$

$\mathbf{d_j}$ Runs:
1: $T = s_j \cdot U$
2: $H = H_2(U, c_{A_i}, T, u_i, P_i, d_j, P_j)$
3: $\mathsf{K_{ij}} = H_1(U, T, x_j \cdot U, d_j, P_j)$
4: $\mathsf{A_i} = \mathsf{Dec}(\mathsf{K_{ij}}, \mathsf{c_{A_i}})$

Subsequently, d_j generates a fresh random number r_2 and uses the CL_SymKeyGen and CL_Encapsulation algorithms to generate and encapsulate a second symmetric session key $\mathsf{K_{jk}}$. Using $\mathsf{K_{jk}}$, d_j encrypts its attributes A_j along with A_i and a set of contextual attributes to produce $c_{A_iA_j}$. Then, d_j sends the following message to f_k for an access decision: $m_2 = \langle r_2, c_{A_iA_j}, \varphi_2, d_j, H_2 \rangle$ where $H_2 = H_{K_{jk}}(r_2 || c_{A_iA_j} || A_i || A_j || \ \varphi_2)$. Upon reception, f_k runs CL_Decapsulation to generate the session key $\mathsf{K_{jk}}$, decrypts $c_{A_iA_j}$ and verifies the freshness and integrity of m_2. On successful decapsulation and message verification, f_k forwards the retrieved list of attributes to the PDP for an access decision AD. The PDP generates an AD on the access request based on the attributes received from d_j, additional attributes from the AR, and the defined access policies in the PR. Once an access decision has been made, f_k encrypts the AD with $\mathsf{K_{jk}}$ to obtain c_{AD}, generates a fresh random number r_3, and sends the following message back to d_j: $m_3 = \langle r_3, c_{AD}, H_3 \rangle$ where $H_3 = H_{K_{jk}}(r_3 || c_{AD} || AD)$. Depending on the content of AD, d_j either grants or denies access to u_i. The complete overview of FitF is presented in Fig. 5.

5. Threat model and security analysis

CL-HSC Security The CL-HSC scheme adopted by FitF has been proven to satisfy confidentiality (indistinguishable against adaptive chosen ciphertext and identity attacks – IND-CCA2) and unforgeability (EUF-CMA). The scheme considers three types of ad-

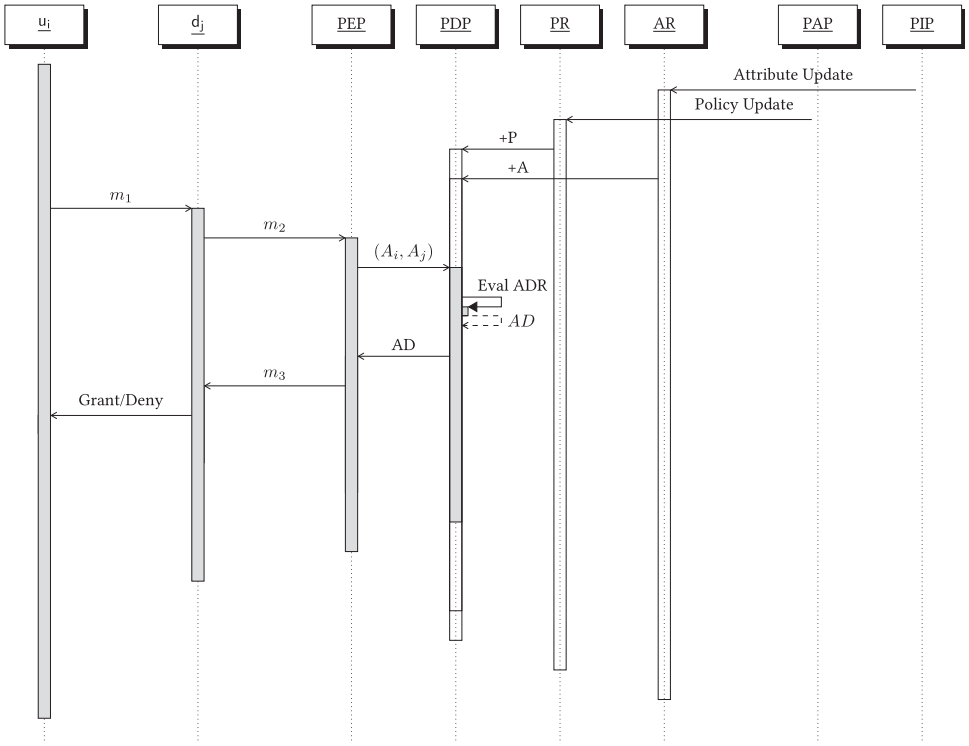


Fig. 5. FitF: The Complete Picture.

versaries: i) a malicious user or device who can replace another user's public keys, ii) a dishonest KGC which has knowledge of the KGC's master secret key, and iii) a previously legitimate user whose partial private/public keys have been revoked by the KGC. To this end, we focused on analyzing the security of the FitF protocol and did not examine the security of the CL-HSC scheme itself, as detailed security analysis and proofs have been presented in Seo and Bertino (2013).

Assumptions about Attacker's Capabilities We consider a powerful adversary $\mathcal{ADV}$ capable of performing a variety of sophisticated attacks such as replay old messages, manipulate exchanged messages between two or more legitimate entities, alter the revocation list, etc. $\mathcal{ADV}$ can corrupt any user from the set of all users $\mathcal{U}$ and any device from the set $\mathcal{D}$. All corrupted users and devices can perform attacks by following the Dolev-Yao adversarial model (Dolev and Yao, 1983). To this end, we assume that $\mathcal{ADV}$ is in full control of the network configuration, can overhear, create, replay and destroy all the exchanged messages between the CSP and their resources as well as with other entities in our system model. Considering the fog nodes and the CSP, we assume that both of them can be corrupted but their behaviour should follow the semi-honest adversarial model. This is needed in order to avoid cases where a corrupted CSP can collude with a malicious fog node. Additionally, we make the realistic assumption that each node is able to verify the owner of a public key. With this assumption, we eliminate the possibility of trivial man-in-the-middle attacks. Based on these assumptions, we exclude attacks from a corrupted CSP and fog capable of performing attacks by following the Dolev-Yao adversarial model. Notwithstanding, we detail such corrupted CSP fog-based internal attacks and their possible countermeasures in Section 8.

Cryptographic Security We assume encryption schemes are semantically secure and the $\mathcal{ADV}$ cannot obtain the plaintext of encrypted messages without knowing the corresponding secret key. Furthermore, we assume that the probability of $\mathcal{ADV}$ guessing a generated random number is negligible. Finally, we explicitly exclude denial-of-service attacks from our adversarial model and we focus on $\mathcal{ADV}$ that aims to compromise the confidentiality of data by forging existing access policies generated by the corresponding data owners.

Finally, we extend the above assumptions on our threat model by defining a set of attacks available to $\mathcal{ADV}$.

Attack 1 (Unauthorized Access Attack)**.** Assume a legitimate user u_i who has access to a device d_j. Let $\mathcal{ADV}$ be an adversary who is able to overhear all communication when u_i requests access to d_j. $\mathcal{ADV}$ successfully performs an Unauthorized Access Attack if she manages to gain access to d_j by tampering with the ciphertext in an access request message in a way that the recipient cannot tell the difference.

Attack 2 (Revocation Deflection Attack)**.** Let $\mathcal{ADV}$ be an adversary who previously had access to a device d_j, but whose access rights have either expired or been revoked. $\mathcal{ADV}$ successfully launches a Revocation Deflection Attack if she manages to convince the responsible fog node that she still has access to d_j by disrupting communication between the CSP and fog node and sending an old access list.

5.1. Security analysis

In this subsection, we prove the security of FitF in the presence of the previously defined adversary.

Proposition 1 (Unauthorized Access Attack Soundness)**.** *Assume a legitimate user u_i who has access to a device d_j and an adversary $\mathcal{ADV}$ who overhears all communication when u_i requests access to d_j. Then $\mathcal{ADV}$ cannot successfully perform an Unauthorized Access Attack.*

Proof. To successfully perform an Unauthorized Access Attack $\mathcal{ADV}$ needs to tamper with the ciphertext in either m_1, m_2, or m_3 of our Request Access phase (Fig. 4). The main goal of $\mathcal{ADV}$, would be to get a positive AD, and fool the recipient into being unable to distinguish a tampered message from a legitimate one. To do so, $\mathcal{ADV}$ has three options available:

Opt1on 1 (Compromise m_1): To successfully launch an unauthorized access attack with this option, $\mathcal{ADV}$ has to replace c_{A_i} in m_1 with another ciphertext c'_{A_i} of her choice such that she receives a positive *AD* at f_k. To do so, $\mathcal{ADV}$ can choose to replay a message m_1 from a previous capture or simply reuse an old ciphertext c_{A_i} and encapsulated tag φ_1 from a previously successful run of the protocol. Although the structure of the replayed message is correct, d_j is able to verify the message is not fresh and aborts the protocol. In the first instance, when $\mathcal{ADV}$ replays an old message m_1, d_j can verify that r_1 is part of an old message it previously received and aborts the protocol. Finally, $\mathcal{ADV}$ can use an old ciphertext c'_{A_i} and φ'_1 such that $m_1 = \langle r_1, c'_{A_i}, \varphi'_1, u_i, H_1 \rangle$. Upon receiving m_1, d_j proceeds to generate K_{ij} and verifies the freshness of m_1. However, as $H_1 = H_{K_{ij}}(r_1||c_{A_i}||A_i||\varphi_1)$, the verification fails and d_j aborts the protocol.

Opt2on 2 (Compromise m_2): Launching an unauthorized access attack with this option has the same limitations as option 1, as both m_1 and m_2 have the same structure (Fig. 4). This fails also.

Opt3on 3 (Compromise m_3): Here we consider a scenario where $\mathcal{ADV}$ launches an unauthorized access attack by tampering with the ciphertext in m_3. In this attack, $\mathcal{ADV}$ either replays an old m_3 from a positive access request or substitutes r_3 and c_{AD} in m_3. Even though the structure of the message is correct with a valid tag, d_j can easily identify that the received message is *not* fresh. In the first instance, d_j is able to tell that the replayed message uses r_3 that is part of an old run of the protocol. In the second instance, r_3 is not the same as the random number in $H_3 = H_{K_{jk}}(r_3||c_{AD}||AD)$. Hence, the verification and integrity check of m_3 fails and d_j aborts the protocol.

□

Proposition 2 (Revocation Deflection Attack Soundness)**.** *Let $\mathcal{ADV}$ be an adversary who previously had access to a device d_j, but whose access has either expired or been revoked. Then $\mathcal{ADV}$, cannot successfully launch an Update Disruption Attack.*

Proof. To successfully launch the Revocation Deflection Attack, $\mathcal{ADV}$ needs to disrupt the communication between the fog node and the CSP thus preventing the fog node from receiving access policies and attributes updates which would deny it access d_j once its access has expired or is revoked. In our architecture, all fog nodes are assumed to be synchronized (i.e. have the same up-to-date policies and attributes information from the CSP at any given time). Hence this attack fails. We note that the propagation of attributes and policies from the cloud to fog nodes raises several continuity and availability concerns in an environment with multiple fog nodes. As such the synchronization of information contained by each fog node may be difficult to achieve due to mitigating factors such as network connectivity issues and/or unavailability of the fog node itself. Therefore, we make a further assumption that any updates in the cloud will be propagated to fog nodes based on a set of rules or predefined schedule with validated and protected timestamps incorporated. In the case where a fog node is unable to access the CSP, it synchronizes AR and PR from nearby fog nodes with the most current timestamp. □

6. Experimental setup and evaluation

This section presents the experiments carried out for the evaluation of our work. The aim of our experiments was twofold:

1. Evaluate the performance of our architecture and the access control mechanism and compare it with the commonly used cloud-based centralised approach;
2. Implement the utilized certificateless hybrid signcryption scheme and measure its performance on two different kinds of resource constrained devices: one that provides hardware acceleration for several cryptographic primitives (Zolertia Remote[1]) and one that does not offer such hardware (TelosB Mote[2]).

To this end, our experiments were categorized into two phases based on the aims described above. Phase 1, focused on the evaluation of the proposed architecture without integrating the adopted certificateless key exchange scheme, while Phase 2, focused on the performance of the certificateless hybrid signcryption scheme on the chosen IoT hardware.

6.1. Phase 1: Architecture performance evaluation

In this phase, we evaluated the proposed architecture in a concrete way by running the same set of experiments using two different architectures and comparing the results. More precisely, we deployed a standard centralized cloud only approach (Fig. 6 a) and our distributed fog-based approach (Fig. 6 b). For the implementation of the PAP, PIP and PDP components, we used Keycloak[3] – a popular open-source identity and access management system. For the PEP component, we utilized a Python-based REST API service, which was specifically developed for the paper.

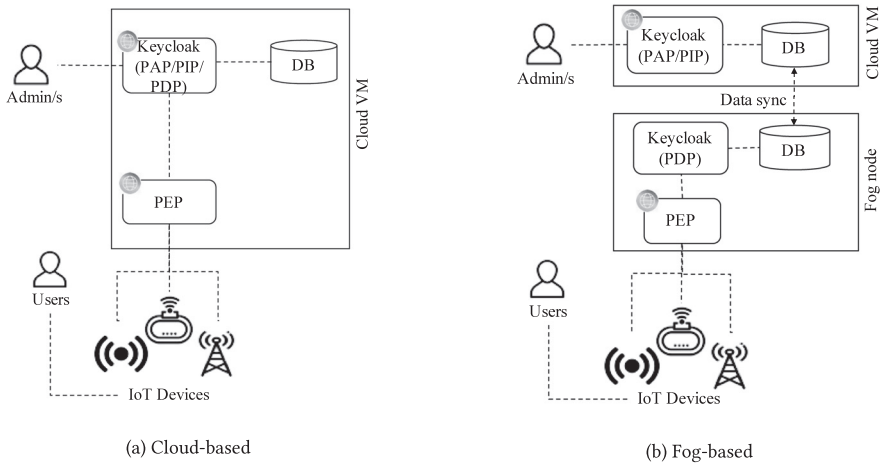
In this phase of the experiments, we simulated the concurrent use of multiple IoT devices that interact with a PEP component to evaluate the end user's access request to a specific device. Apache JMeter[4] – a load testing tool used to analyze the performance of web-based systems – was used to simulate concurrent access requests from multiple IoT devices. Upon receiving access requests, The PEP component evaluates them and outputs an authorization decision (i.e. grant/deny access) by performing the following steps:

Step 1. Authenticate users from Keycloak. If a user is valid, the PEP receives an access token and proceeds to the next step. Alternatively, it denies the access request.
Step 2. Prepare a Keycloak specific ADR.
Step 3. Send ADR to Keycloak to evaluate and return the result through a specialised token called Requesting Party Token (RPT) which contains user specific authorization related information to the IoT devices.
Step 4. Process RPT token and decide whether the access to the underlying IoT device will be granted or denied.

As described in Section 4, the communication between the IoT devices and the PEP is secured via CL-HSC. However, for this phase of our experiments, the communication between a device and PEP in both scenarios, was over TLS with *mutual* client authentication. Additionally, the communication between PEP and PDP was secured using native Keycloak SAML assertions.

Based on our described architecture (Section 3.1), PIP and PAP propagate their contents to the PR and AR components of a fog node. Such data transfer is achieved practically via modern database systems (e.g. MariaDB or PostgreSQL). Identity and access management systems, such as Keycloak, support different kinds of databases including MariaDB and PostgreSQL. These databases provide data synchronization mechanisms across different servers. The details of such methods are considered to be beyond the scope of

[1] https://github.com/Zolertia/Resources/wiki/RE-Mote.
[2] https://telosbsensors.wordpress.com/.
[3] https://www.keycloak.org/.
[4] https://jmeter.apache.org/.



(a) Cloud-based (b) Fog-based

Fig. 6. Experimental setup.

this paper and therefore, no performance analysis of synchronization aspects is included. For further details on database replication, please refer to Galera cluster documentation.[5] The testbed for this phase is detailed below:

Testbed: Phase 1

- **CSP:** A virtual machine (VM) on Microsoft Azure [4 virtual CPUs, 16GB RAM] running Ubuntu 18.04 LTS.
- **Fog:** An iMac desktop computer [4 CPUs, 24GB RAM] running MacOS.
- **IoT Devices:** A laptop, running Apache JMeter to simulate the concurrent use of multiple IoT devices. Configuration: [4 CPUs, 12GB RAM] running Ubuntu 18.04 LTS.

During the entire run of these experiments, the Cloud VM for the cloud-based scenario was deployed in the West Europe region with access requests being generated from Glasgow, UK. In the proposed fog-based scenario, the fog node and machine hosting access requests component (i.e. Apache JMeter) were connected through a Local Area Network (LAN).

For the purposes of our experiments, we configured each IoT device as a resource in Keycloak and created a repository of multiple users. Each user was assigned a specific role which granted access to the IoT devices. Such configurations are defined in Keycloak as authorization policies. The PEP component, with the help of the PDP, enforces the authorization constraints based on users and device attributes. These configurations were fully synchronized across cloud and fog node before the experiments. To test the performance of both approaches (i.e. the proposed fog-based vs cloud-based) the following set of experiments were conducted:

6.1.1. Communication overhead and system performance

The primary aim of this experiment was to identify the differences in the communication overhead and its impact on system performance in both set-ups. To do so, we measured the following metrics:

- Connect.time: The average time required to establish a connection between the IoT device and PEP;
- Proc.time: The processing time required for PEP to perform authorization decisions;
- Latency: The time interval from the instant when a device generates and sends an AR to the instant it receives an authorization decision from PEP.

We simulated an IoT device that repeatedly generated and sent access requests to the PEP component. The obtained results are summarized in Fig. 7, where each measurement in the plot indicates the average results for 10 runs. This figure shows that the Proc.time in both instances had a negligible difference. This was expected since the PEP and PDP settings in both scenarios were identical. On the other hand, the Connect.time (i.e. a key parameter in terms of determining the communication overhead) was much lower for our proposed fog-based approach as compared to the cloud-based approach. This has a direct correlation to the latency (i.e. performance) of the system. We measured an average of 103 milliseconds for the fog-based approach and 257 milliseconds for the cloud-based approach.

6.1.2. System performance during various stress levels

As a next step, we wanted to evaluate the performance of both approaches under various stress situations. To do so, we considered a scenario where multiple IoT devices concurrently generated and sent access requests to the PEP component. In these experiments, we gradually increased the number of simulated devices from 200 to 1000. Furthermore, we considered three different Sample.time settings, i.e. 5, 10, and 15 s. We defined Sample.time as the duration required to run the threads responsible for imitating the concurrent IoT devices.

For each Sample.time setting, we recorded the average latency and the percentage of error (i.e. percentage of unsuccessful requests) while varying the number of IoT devices. An access request was considered unsuccessful if the PEP was unable to process it for any particular reason. The obtained results (for an average of 10 runs) for latency and error are illustrated in Figs. 8 and 9 respectively.

It is evident from the plots in Fig. 8 that the system performance of fog-based approach in each settings of Sample.time, i.e. (5, 10 or 15 s), was significantly better than the cloud-based centralised approach. For example, with Sample.time of 5 s and 200 IoT devices, the average latency in the case of fog-based approach was 0.12 s, whereas, in the case of cloud-based approach, it was approximately 3.2 s. It is clear that the increase in the number of concurrent IoT devices had an exponential degrading effect on latency in the case of cloud-based approach. However, in the case of fog-based approach, the effect on latency was comparatively more gradual rather than exponential. Furthermore, it is worth mentioning that there were cases where the latency was not affected at all. For example, when Sample.time setting was 15 s and the number of concurrent IoT devices was $\leq$ 600. The key reason behind this compelling difference in the performance of the two approaches is

[5] https://galeracluster.com/library/documentation/docker.html.

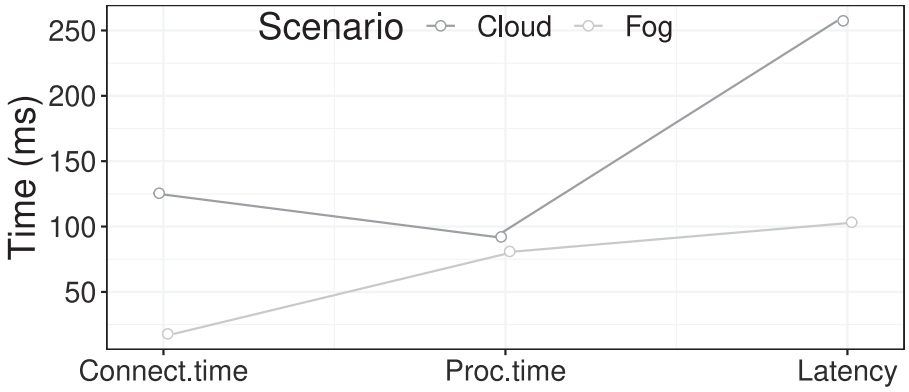


Fig. 7. Communication overhead and system performance.

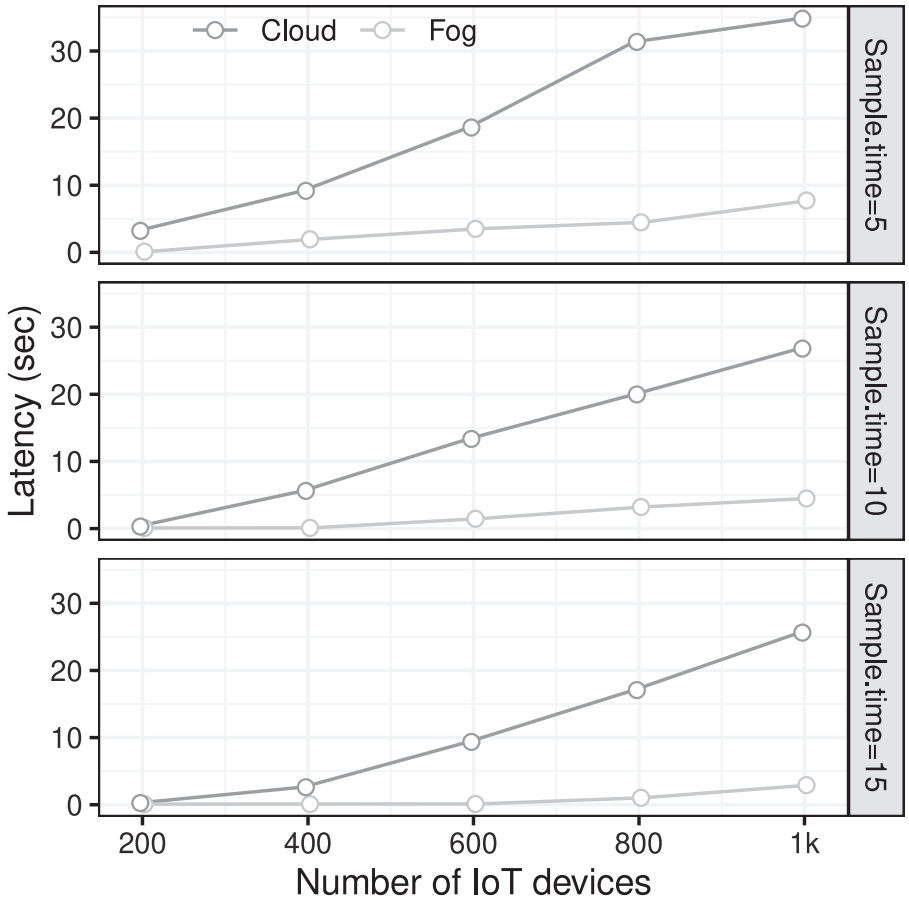


Fig. 8. System performance with concurrent use of varying number of IoT devices using different applied Sample.time settings.

the communication between the machine hosting the Apache Jmeter component, i.e. the IoT devices, and the machine hosting the PEP and PDP components. For the fog-based approach, this communication is within the same vicinity, with both machines connected through LAN. On the other hand, in the case of cloud-based approach, the Apache Jmeter component, i.e. the IoT devices, interact with the VM located in the West Europe region.

Fig. 9 presents the results of the error, i.e. the unsuccessful requests, observed during each case of applied Sample.time setting against concurrent use of varying number of IoT devices. It can be seen that for both approaches there were no unsuccessful requests when the number of IoT devices was $\leq$ 400 (despite the Sample.time setting). However, in scenarios where the number of IoT devices was $\geq$ 400, and the applied Sample.time was 5 and 10 s, the number of unsuccessful requests was comparatively high in the case of fog-based approach. The reason for this was the TLS handshaking between the IoT devices and PEP component. This occurred primarily because the PEP component ran in development mode as a single instance without any load balancing capabilities. This resulted in an increased number of errors, when the number of concurrent IoT devices was comparatively large. However, this behaviour was *not* replicated when the incoming access requests load was comparatively lower, e.g. there were no occurrences of unsuccessful requests when Sample.time was 15 s and the number of concurrent IoT devices was $\leq$ 800.

6.2. Phase 2: IoT performance evaluation

For this phase of the experiments, we shifted our focus to the performance of the CL-HSC scheme on resource-constrained IoT

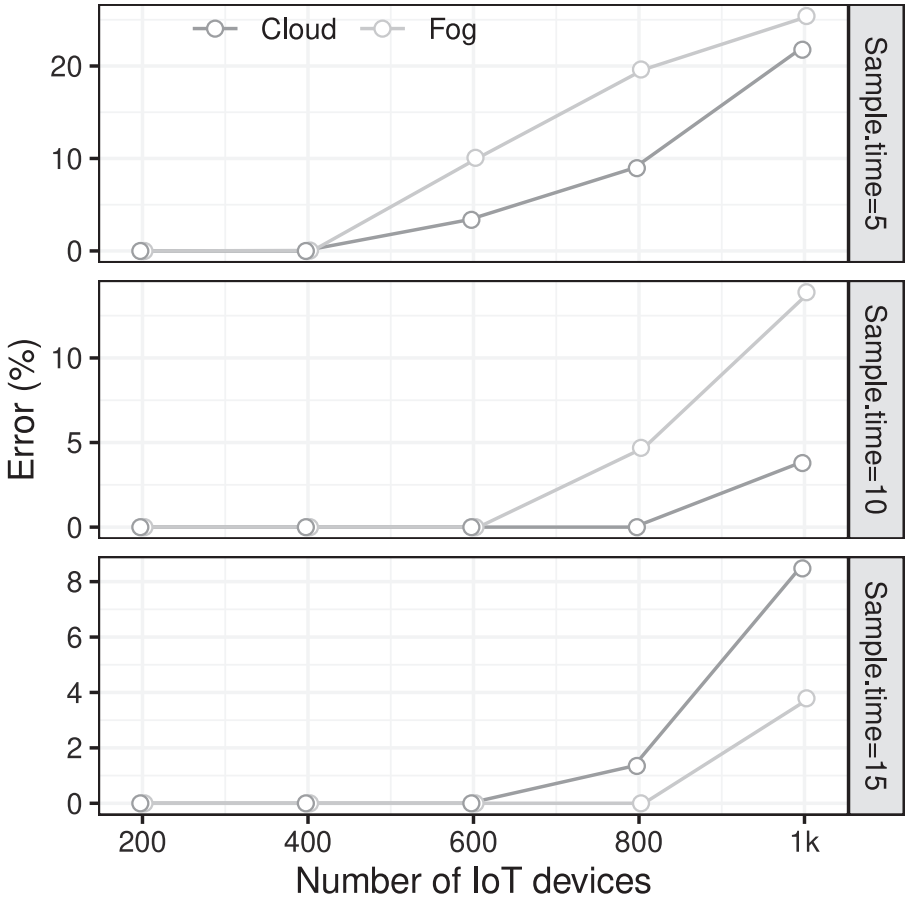


Fig. 9. Observed error (i.e. unsuccessful access requests).

devices. To comprehensively measure the performance of each CL-HSC algorithm, we repeatedly executed them 50 times on a Zolertia board and TelosB board. Additionally, we calculate the energy consumed during the execution of these algorithms. These boards were chosen to demonstrate the feasibility of the scheme on resource-constrained devices of varying specifications. We used Tiny OS Group (2005) as our IoT platform for the TelosB board and, Contiki-NG (2019) for the Zolertia board. In both instances, TinyECC library (Liu and Ning, 2008) was used for the basic ECC operations needed to implement the scheme.

Testbed: Phase 2

- **Support for Crypto Hardware Acceleration:** A Zolertia Remote board with 512KB programmable flash and 32KB RAM.
- **No Hardware Acceleration:** A TelosB Mote with 48KB programmable flash and 10KB RAM.

Execution Time : During the course of these experiments, we observed that the execution time for each algorithm correlated directly to the number of Elliptic Curve (EC) point multiplications performed by the device. The experiments focused on measuring the performance of the following algorithms executed by the constrained devices: CL_GenSecretValue, CL_GenPrivKey, CL_GenPubKey, CL_SymKeyGen, CL_Encaps ulation, and CL_Decapsulation. From our results, we observed that the most computationally expensive operation for both devices was the CL_SymKeyGen algorithm, for which the execution time was approximately 14.343 s for the TelosB device and 14.125 s for the Zolertia device. This algorithm involves 5 EC point multiplications. The cheapest algorithm for both devices was CL_Encapsulation; its execution times were approximately 0.016 and 0.002 s for the TelosB and Zolertia device respectively. As shown in Table 1, this algorithm involves no EC point multiplication. The complete results are provided in Table 1.

Energy Consumption: We calculated the energy consumption of each algorithm by using the CPU current consumption and voltage values in the datasheets of both devices. CPU current in the context of our experiments constitutes the current drawn by the device in active CPU and radio listening modes. As expected, the CL_SymKeyGen algorithm consumes the most energy with 847mJ for the Zolertia device and 1067mJ for the TelosB board. On the other hand, CL_Encapsulation consumes the least amount of energy with 0.12mJ and 1.19mJ respectively (Table 1). To conclude, the overall performance of the selected CL-HSC algorithms on both devices suggests that it would be more efficient to compute the expensive CL_SymKeyGen function off-line. In such a setup, the smart lock would run the CL_SymKeyGen function to generate a session key, K_{jk}, between itself and a nearby fog node before the actual run of the FitF protocol (refer to the FitF – Request Access phase of Section 4.2). This would reduce the total runtime of the proposed protocol significantly.

7. Application domain

Our architecture can be employed in a wide range of real world IoT applications across a few of its super-domains (smart homes, smart cities etc). For the purpose of this paper we chose to describe our system in the context of the Smart Hotel Management and Smart Home lock IoT application domains.

Smart Hotel Management: In this scenario, we consider a hotel management system where each hotel room door is equipped with

Table 1
Constrained Device Performance.

		TelosB		Zolertia Re-mote	
	EM	Time (sec)	Energy (mJ)	Time (sec)	Energy (mJ)
CL_GenSecretValue	1	3.518	261.73	2.804	168.24
CL_PrivKeyGen	0	0.203	15.1	0.007	0.42
CL_PubKeyGen	0	0.199	14.8	0.006	0.36
CL_SymKeyGen	5	14.343	1067	14.125	847.5
CL_Encapsulation	0	0.016	1.19	0.002	0.12
CL_Decapsulation	2	7.218	537.02	6.367	382.02

a smart lock device that grants or denies access to guests based on a combination of attributes defined in the hotel's access policy. The combination of attributes may include attributes of the guest (e.g. name and reservation details), attributes of the smart lock (e.g. door number, location, and serial number), and contextual attributes such as the time of day. With this set-up, once a guest pays and reserves a room at the hotel, they are assigned specific rights and attributes and given the system parameters to generate the requisite security keys. Additionally, each IoT smart lock is also assigned specific attributes and given the system parameters needed for key generation during installation or scheduled updates. We assume that for each hotel or group of hotels in a specific location, there is one fog node that processes all access requests generated when guests try to access a hotel room. Guests establish a secure session with the smart lock when trying to access their rooms and subsequently, the smart locks establish a secure session with the PEP components in that fog node. Once the secure sessions have been established, guests' access requests are evaluated at the fog node based on the hotel's defined policies. This application offers a contactless and secure management of hotel rooms with a continuous communication work-flow that ensures real-time control and oversight.

Smart Home Lock: To illustrate the advantages of FitF in a smart home lock application, we will consider the August Smart Lock[6], a standard smart home lock, which employs a simple Lock or Unlock operation mode with a history log that shows who locked or unlocked the device. The August lock architecture consists of the *smart lock, mobile application* and *remote server*. The smart lock is equipped with Bluetooth Low Energy (BLE) with no internet functionality and only communicates with the user's mobile application. The smart lock enforces all defined policies while the remote server provides policy administration, storage, and decision-making functions. By supporting only BLE, the smart lock can only contact the remote server when a user is nearby. Additionally, the August Smart Lock employs a centralized Group-Based Access Control (GBAC) (Bakar et al., 2009) model with an access management system that resides in the cloud-based remote server. There are two types of defined user groups: Owner and Guest. The owner group has a higher authority, where members can perform all operations available to a user such as the provisioning of guest users or other users in the owner group, updating other user roles, and specifying timeslots for the guests to access the smart locks. The use of FitF, would introduce a revamped secure, and distributed architecture for the smart home lock. The smart lock would be able to generate access decisions independent of the user's mobile application in a secure and privacy-preserving way while also supporting fine-grained access control policies.

8. Extended threat model

The proposed architecture and protocol have been proven to be efficient and secure against a number of identified external attacks. However, we acknowledge that our protocol may be weak against internal attacks. While we believe this category is out of the scope of this work[7], it is important to consider such malicious behaviors. This will not only inspire our next steps towards building a more secure and therefore complete scheme, but will also allow other researchers to contribute to our work by pointing out inefficiencies in our scheme and formulating open research challenges. To this end, we extend the threat model in Section 5 to consider a corrupted CSP and fog capable of performing attacks by following the Dolev-Yao adversarial model. Through this extension, we seek to identify malicious internal attacks that our scheme is susceptible to and elaborate on possible enhancements aimed at securing the scheme. It is worth noting that these enhancements may introduce additional computational and communication overhead with negative impact on the scheme's efficiency. The following attacks were identified:

Attack 3 (Insider Modification Attack)**.** Let $\mathcal{ADV}$ be an adversary who has successfully compromised a device d_x. $\mathcal{ADV}$ successfully launches the Insider Modification attack if she successfully modifies the user attribute list in m_1 received from user u_i in the access request phase.

This attack can be mitigated by modifying the *Request Access* phase of our protocol in Section 4.2. In this modified version, a user u_i requesting access to a device d_x first generates a hash of its attributes and timestamp, and signs it with their full private key, sk_i: $\sigma_i = sig_{\mathsf{sk}_i}(H(A_i||t_i))$. The signed value is sent as part of m_1 and m_2 to a nearby fog node f_k ensuring that the fog node can always verify the attributes of the user. Hence, if a malicious device wishes to tamper with the attributes of u_i in an indistinguishable way, it should also forge a new signature, signed with the secret key of u_i. However, given the EUF-CMA security of the signatures, this can only happen with negligible probability. We can further extend this solution by requiring the device to also sign a hash of the list of its attributes and user attributes, and timestamp. However, this solution introduces additional computational cost from the signing and verification operations.

Attack 4 (CSP/Fog Manipulation Attack)**.** Let $\mathcal{ADV}$ be an adversary who has successfully compromised a legitimate CSP or fog node. $\mathcal{ADV}$ successfully launches the CSP/Fog Manipulation Attack if she successfully manipulates the contents of the PR, AR, PIP or PAP to ensure that a user u_i's access request is explicitly denied or granted.

To potentially address this attack, we can enhance the features of the CSP and fog nodes to include support for a Trusted Execution Environment (TEE). TEEs are tamper-resistant processing environments that guarantee the authenticity of executed code, the integrity of runtime states, and the confidentiality of code, data and

[6] https://august.com/products/august-smart-lock-pro-connect.

[7] The main purpose of this paper is to verify the applicability of the access control mechanism we designed and elaborate on the efficiency and benefits of the proposed architecture.

runtime states stored on a persistent memory (Sabt et al., 2015). By using a TEE, we can guarantee a level of policy and attribute repository trustworthiness that is currently not available to either the CSP or fog nodes. Thereby ensuring that the CSP manipulation attack is unsuccessful.

Attack 5 (Collusion Attack)**.** In a collusion attack, two dishonest entities attempt to deduce private information about legitimate users in order to gain unauthorized access to a resource by actively sharing and/or combining information available to them.

Overall, there are four (4) entities that actively partake in the access control management of FitF. Assuming that a malicious adversary $\mathcal{ADV}$ can compromise any of these entities, we end-up having the following five (5) possible collusion attacks:

- User and User collusion: We consider a scenario where two users u_y and u_z attempt to combine their attribute lists A_y and A_z to gain unauthorized access to a device d_x. This is a possible limitation that can be addressed by extending the proposed system model in such a way that a Registration Authority (RA) will be included – similar to the the approach followed in Michalas (2019). RA will be responsible for the registration of users' attributes. Additionally, RA will secure users' attributes by encrypting them and signing them. Therefore, two corrupted users trying to combine their attributes in order to get unauthorized access to a resource will fail since *(1)* they will not have access to plaintext attributes and *(2)* they will not have access to the secret key used by RA to sign the maliciously generated list. Finally, it is worth mentioning that RA can run as a separate third party or can be also implemented as part of the CSP.
- CSP and User collusion: In this attack, we consider an adversary $\mathcal{ADV}$ who has corrupted a user u_c and the CSP. The attack involves the collusion of the two entities with the main aim of granting u_c unauthorized access to a resource (i.e. user's malicious/invalid access request is explicitly granted). For this attack to be successful, the CSP needs to deviate from the protocol and modify the contents of the PAP and PIP, thereby enabling a fog node grant access requests from u_c. This attack can be addressed by enhancing the CSP with TEE support. Doing this ensures the CSP can be attested (remotely or locally) during the launch phase (Paladi et al., 2017; Santos et al., 2009). We can then get certain guarantees about the trusted state of the CSP and be sure that it will not deviate from the protocol.
- User and Device collusion: A collusion attack between a dishonest user and a dishonest device is considered unrealistic as this kind of attack renders the entire access control system irrelevant. The dishonest user will always gain access to a dishonest device if they collude.
- Device and Fog Node collusion: For this attack, we consider a dishonest device actively colluding with a dishonest fog node to ensure a user's access request is either explicitly denied or granted. This attack can be addressed by signing user access requests as discussed in Attack 3 and enhancing fog nodes with TEE support as discussed in Attack 4. Doing this ensures that a fog node is unable to maliciously tamper with the contents of the PR and AR, as well as providing security guarantees on the state of the fog node. Signing user access requests also serves to provide a form of access request verification.
- Fog Node and CSP collusion: In our architecture, fog nodes operate as extension of the CSP. As such, a collusion between a dishonest fog node and dishonest CSP is the same as a single dishonest CSP or fog node. A possible solution to this attack would be to have encrypted and publicly verifiable policies. This should be coupled with a TEE-based CSP and fog node, as discussed in Attack 4. Following this approach, a malicious Fog node would not be able to change attributes of users according to policies or use a non-legitimate policy. Apart from that, by doing this, we can provide further security to our scheme since the fog nodes will not be able to see policies and users' attributes in clear text. Hence, important information about users will remain private. However, it is worth mentioning that such an approach is not straightforward and requires a thorough redesign of our protocol. It addition to that, such a solution will significantly affect the performance and the applicability of our approach.

9. Discussion/Comparison with attribute-based encryption

During the last years, we have seen some great developments in the area of cryptographic access control with most notable being systems utilizing the promising concept of Attribute-Based Encryption (ABE). ABE is a form of public-key cryptography that evolved from Identity-Based Encryption (Boneh and Franklin, 2001). *The main concept of ABE is simple and fascinating:* Data is encrypted based on a public key and an access policy. Then, a unique private key is generated for each user. This key is generated based on a list of attributes (e.g. Date of Birth, Country of Residence, etc.). Consequently, a user is able to decrypt a file that is associated with a certain policy *if and only if* the attributes of her key satisfy the underlying policy. In other words, one single ciphertext can be decrypted by multiple different keys. Access policies are usually captured as Boolean expressions of attributes using `AND, OR, NOT` and threshold (`k out of n` attributes hold) operations. ABE is used to implement data access control schemes, including role-based access control, where the access control scheme is (effectively) "incorporated" into the encrypted data. For the generation of users' unique keys based on personalized attributes, trusted attribute authorities have been considered in the literature in order to verify that an authenticated user has the proper attributes before allowing any decryption. Attributes may refer to any kind of information describing a user, a piece of sensitive data or some other entity or notion. While ABE schemes offer great flexibility in terms of access management, they suffer from certain drawbacks. Most importantly, the size of the produced ciphertexts and the time required to both encrypt and decrypt data grows with the complexity of the underlying policy (Green et al., 2011). As a result, online access-control systems that are solely based on such schemes are currently considered as inefficient. Hence, while the concept of ABE has the potential to unleash new, creative, useful and emerging applications, from a practical perspective, it still holds a *largely unfulfilled* promise. In light of this, it is only natural that we discuss why we opted for a certificateless cryptographic scheme as compared to an ABE scheme.

The goal of FitF is to achieve a distributed, efficient, secure, and lightweight attribute-based access control system. As a result, we opted against the use of ABE schemes primarily due to the computational overhead it would incur on the resource-constrained devices we consider for this work. Furthermore, in contrast to our work, ABE schemes are limited in terms of specifying policies and managing user attributes. Decryption keys support user attributes that are organized logically as a single set. As such, the user can only use all possible combinations of attributes in a single set as issued in their keys. In contrast, the certificateless cryptographic scheme we adopted and the ABAC model allows us to use a fully expressive set of attributes. However, and based on the fact that we see a great potential in utilizing efficient ABE schemes in modern access control systems, as a next step, we plan to work towards implementing lightweight ABE schemes that can smoothly run on constrained devices. We believe that

achieving this, will allow us to further improve and modernize our architecture.

10. Conclusion

This paper presented a fog-based distributed access control solution for IoT systems. Our proposed architecture distributes core components of the access control system between fog computing nodes (i.e. closer to the IoT devices) and the cloud. By utilizing fog nodes, we eliminate the need for IoT devices and users to constantly connect to the cloud, thus reducing the communication overhead. To secure the communication between all our components, we adopted and implemented a certificateless hybrid signcryption scheme (CL-HSC) between the IoT device and fog node/users. Furthermore, to test the effectiveness and performance of our proposed solution, we performed various comprehensive experiments and outlined their results. Observed results support the hypothesis that our proposed architecture performs considerably better than the commonly used centralized cloud-based architectures.

With this work we hope to pave the way towards more distributed and secure access control systems that can be directly applied to a wide range of domains. We believe this work has the potential to kick-start a period of accelerated research in the field of distributed access control and give the opportunity to both architecture and protocol designers to improve the effectiveness of their designs by providing insight into possible flaws and inefficiencies.

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.

CRediT authorship contribution statement

Eugene Frimpong: Conceptualization, Formal analysis, Writing – original draft, Writing – review & editing, Software, Investigation. **Antonis Michalas:** Conceptualization, Formal analysis, Supervision, Project administration, Funding acquisition. **Amjad Ullah:** Software, Investigation.

Acknowledgment

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 952071 (DIGITbrain) and the Technology Innovation Institute (TII), Abu Dhabi, United Arab Emirates, for the project ARROWSMITH: Living (Securely) on the edge.

References

Ahmad, T., Morelli, U., Ranise, S., Zannone, N., 2018. A lazy approach to access control as a service (ACaaS) for IoT. 23rd ACM on Symposium on Access Control Models and Technologies - SACMAT 18.

Aleisa, M.A., Abuhussein, A., Sheldon, F.T., 2020. Access control in fog computing: challenges and research agenda. IEEE Access 8, 83986–83999. doi:10.1109/access.2020.2992460.

Alshehri, A., Sandhu, R., 2017. Access control models for virtual object communication in cloud-enabled IoT. In: IEEE International Conference on Information Reuse and Integration doi:10.1109/iri.2017.60.

Alshehri, A., Sandhu, R., 2016. Access control models for cloud-enabled internet of things: a proposed architecture and research agenda. In: 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC).

Bakar, A.A., Ismail, R., Ahmad, A.R., Manan, J.l.A., Jais, J., 2009. Group based access control scheme: proof of method for secure access control architecture in mobile ad-hoc networks, pp. 446–451. doi:10.1145/1821748.1821833.

Barkley, J., 1997. Comparing simple role based access control models and access control lists. In: Proceedings of the Second ACM Workshop on Role-Based Access Control - RBAC '97 doi:10.1145/266741.266769.

Boneh, D., Franklin, M.K., 2001. Identity-based encryption from the weil pairing. In: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. Springer-Verlag, London, UK, UK, pp. 213–229. http://dl.acm.org/citation.cfm?id=646766.704155.

Contiki-Ng. Contiki-NG: Documentation. 2019. https://github.com/contiki-ng/contiki-ng/wiki.

Damgård, I., Haagh, H., Orlandi, C., 2016. Access control encryption: enforcing information flow with cryptography. Theory Cryptogr. 547–576. doi:10.1007/978-3-662-53644-5_21.

Dolev, D., Yao, A., 1983. On the security of public key protocols. IEEE Trans. Inf. Theory 29 (2), 198–208.

Frimpong, E., Bakas, A., Dang, H.V., Michalas, A., 2020. Do not tell me what i cannot do! (the constrained device shouted under the cover of the fog): implementing symmetric searchable encryption on constrained devices. In: Proceedings of the 5th International Conference on Internet of Things, Big Data and Security doi:10.5220/0009413801190129.

Gilani, K., Bertin, E., Hatin, J., Crespi, N., 2020. A survey on blockchain-based identity management and decentralized privacy for personal data. In: 2020 2nd Conference on Blockchain Research; Applications for Innovative Networks and Services (BRAINS) doi:10.1109/brains49436.2020.9223312.

Green, M., Hohenberger, S., Waters, B., 2011. Outsourcing the decryption of ABE ciphertexts. In: SEC'11: Proceedings of the 20th USENIX Conference on Security. USENIX Association, Berkeley, CA, USA, p. 34. http://dl.acm.org/citation.cfm?id=2028067.2028101.

Group, T.T.x.W., 2005. TinyOS 2.0. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems - SenSys '05.

Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., Cybersecurity, S., 2013. Guide to attribute based access control (ABAC) definition and considerations.

Hu, C.T., Ferraiolo, D.F., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., 2014. Guide to attribute based access control (ABAC) definition and considerations.

Johnsen, F.T., Bloebaum, T.H., 2012. Topic discovery for publish/subscribe web services. In: 8th International Wireless Communications and Mobile Computing Conference.

Liu, A., Ning, P., 2008. TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In: 2008 International Conference on Information Processing in Sensor Networks (IPSN 2008).

Michalas, A., 2019. The lord of the shares: combining attribute-based encryption and searchable encryption for flexible data sharing. In: SAC'19: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. Association for Computing Machinery, New York, NY, USA, pp. 146–155. doi:10.1145/3297280.3297297.

Michalas, A., Komninos, N., Prasad, N.R., 2011a. Mitigate dos and DDos attack in mobile ad hoc networks. Int. J. Digit. CrimeForensics (IJDCF).

Michalas, A., Komninos, N., Prasad, N.R., 2011b. Multiplayer game for DDos attacks resilience in ad hoc networks. In: 2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace; Electronic Systems Technology (Wireless VITAE) doi:10.1109/wirelessvitae.2011.5940931.

Oasis. Introduction to XACML. 2013. https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html.

Ouaddah, A., Mousannif, H., Elkalam, A.A., Ouahman, A.A., 2017. Access control in the internet of things: big challenges and new opportunities. Comput. Netw. 112.

Paladi, N., Gehrmann, C., Michalas, A., 2017. Providing user security guarantees in public infrastructure clouds. IEEE Trans. Cloud Comput. 5 (3), 405–419. doi:10.1109/TCC.2016.2525991.

Ravidas, S., Lekidis, A., Paci, F., Zannone, N., 2019. Access control in internet-of-things: a survey. Jo. Netw. Comput. Appl.

Research S. Number of connected devices worldwide 2030. 2020. https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/.

Sabt, M., Achemlal, M., Bouabdallah, A., 2015. Trusted execution environment: what it is, and what it is not. 2015 IEEE Trustcom/BigDataSE/ISPA doi:10.1109/trustcom.2015.357.

Salonikias, S., Mavridis, I., Gritzalis, D., 2016. Access control issues in utilizing fog computing for transport infrastructure. In: Critical Information Infrastructures Security Lecture Notes in Computer Science, pp. 15–26.

Sandhu, R., 2000. Engineering authority and trust in cyberspace. In: Proceedings of the Fifth ACM Workshop on Role-Based Access Control - RBAC 00.

Santos, N., Gummadi, K.P., Rodrigues, R., 2009. Towards trusted cloud computing. In: HotCloud'09: Proceedings of the 2009 Conference on Hot Topics in Cloud Computing. USENIX Association, Berkeley, CA, USA. http://dl.acm.org/citation.cfm?id=1855533.1855536.

Selvarajah, K., Arief, B., Tully, A., Blythe, P.T., 2012. Deploying wireless sensor devices in intelligent transportation system applications. Intell. Transp. Syst. doi:10.5772/25993.

Seo, S.H., Bertino, E., 2013. Elliptic curve cryptography based certificateless hybrid signcryption scheme without pairing.

Servos, D., Osborn, S.L., 2017. Current research and open problems in attribute-based access control. ACM Comput. Surv. 49 (4), 1–45.

Shafagh, H., Burkhalter, L., Hithnawi, A., Duquennoy, S., 2017. Towards blockchain-based auditable storage and sharing of IoT data. In: Proceedings of the 2017 on Cloud Computing Security Workshop doi:10.1145/3140649.3140656.

Tasali, Q., Chowdhury, C., Vasserman, E.Y., 2017. A flexible authorization architecture for systems of interoperable medical devices. In: Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies - SACMAT '17 Abstracts doi:10.1145/3078861.3078862.

Xu, S., Li, Y., Deng, R.H., Zhang, Y., Luo, X., Liu, X., 2020. Lightweight and expressive fine-grained access control for healthcare internet-of-things. IEEE Trans. Cloud Comput. 10 (1), 474–490. doi:10.1109/tcc.2019.2936481.

Xu, S., Ning, J., Ma, J., Huang, X., Pang, H.H., Deng, R.H., 2021. Expressive bilateral access control for internet-of-things in cloud-fog computing. In: Proceedings of the 26th ACM Symposium on Access Control Models and Technologies doi:10.1145/3450569.3463561.

Yadav, S.G.S., Meghashree, N., 2017. Design and implementation of MDCF protocol. In: 2017 International Conference on Inventive Systems and Control (ICISC) doi:10.1109/icisc.2017.8068641.

Zhang, Y., Zheng, D., Deng, R.H., 2018. Security and privacy in smart health: efficient policy-hiding attribute-based access control. IEEE Internet Things J. 5 (3), 2130–2145. doi:10.1109/jiot.2018.2825289.

Zhao, J., Zeng, P., Choo, K.K.R., 2021. An efficient access control scheme with outsourcing and attribute revocation for fog-enabled e-health. IEEE Access 9, 13789–13799. doi:10.1109/access.2021.3052247.

Zyskind, G., Nathan, O., Pentland, A.S., 2015. Decentralizing privacy: using blockchain to protect personal data. 2015 IEEE Security and Privacy Workshops doi:10.1109/spw.2015.27.

Eugene Frimpong is a Doctoral Researcher with the Network and Information Security (NISEC) group at the Department of Computing Sciences in Tampere University, Finland. He received his Master's degree in Cyber Security and Forensics from the University of Westminster in London, England and his Bachelor's degree in Computer Engineering from Kwame Nkrumah University of Science and Technology in Kumasi, Ghana. Currently, he is pursuing his PhD in the field of IoT Security, Privacy and Access Control under the supervision of Dr. Antonis Michalas at Tampere University. His research interests extend to Fog, Mist and Cloud computing and their interaction with the IoT. His work seeks to design, implement and analyse various security and privacy preserving protocols that can be utilized in the IoT ecosystem.

Prof. Antonis Michalas received his PhD in Network Security from Aalborg University, Denmark and he is currently working as an Assistant Professor at the Department Computing Sciences, at Tampere University, Finland where he also coleads the Network and Information Security Group (NISEC). The group comprises Ph.D., students, professors and researchers. Group members conduct research in areas spanning from the theoretical foundations of cryptography to the design and implementation of leading edge efficient and secure communication protocols. Apart from his research work at NISEC, as an assistant professor he is actively involved in the teaching activities of the University. Finally, his role expands to student supervision and research projects coordination. Furthermore, Antonis has published a significant number of papers in field related journals and conferences and has participated as a speaker in various conferences and workshops. His research interests include private and secure e-voting systems, reputation systems, privacy in decentralized environments, cloud computing, trusted computing and privacy preserving protocols in eHealth and participatory sensing applications.

Amjad Ullah is a Lecturer at the department of computer science in Edinburgh Napier University, Scotland, UK. He received his MSc in Advanced Distributed System degree from the University of Leicester, UK, in 2011 and Ph.D. degree in Computer science from University of Stirling, UK in 2017. His research interests include cloud-to-edge computing, internet-of-things, dynamic resource provisioning in cloud computing, orchestration and runtime management of applications in cloud-to-edge environments, cloud auto-scaling, deadline-based auto-scaling policies to support batch-based applications in the cloud environment.