Privacy-Preserving Multi-Party Access Control for Third-Party UAV Services

Dominik Roy George
Eindhoven University of Technology
Eindhoven, The Netherlands
d.r.george@tue.nl

Savio Sciancalepore
Eindhoven University of Technology
Eindhoven, The Netherlands
s.sciancalepore@tue.nl

Nicola Zannone
Eindhoven University of Technology
Eindhoven, The Netherlands
n.zannone@tue.nl

ABSTRACT

Third-Party Unmanned Aerial Vehicle (UAV) Services, a.k.a. *Drone-as-a-Service (DaaS)*, are an increasingly adopted business model, which enables possibly unskilled users, with no background knowledge, to operate drones and run automated drone-based tasks. Although these services provide significant advantages, the resources provided by drones are typically owned by multiple parties. Thus, Third-Party UAV services require adopting multi-party access control solutions. In this context, the leakage of the access control policies specified by the data owners might disclose confidential information and, thus, they should be protected as well. In this work, we propose a privacy-preserving multi-party access control solution tailored to the application scenarios of Third-Party UAV Services. Our solution advances an existing privacy-preserving multi-party access control framework based on Secure Function Evaluation to fit the distributed and heterogeneous nature of drone deployments. Through an extensive experimental evaluation, we demonstrate our solution can perform private policy evaluation on constrained devices in a reasonable time while requiring limited communication, memory, and energy overhead.

CCS CONCEPTS

• **Security and privacy → Access control**; • **Theory of computation → Cryptographic protocols**; • **Computer systems organization → Embedded and cyber-physical systems**.

KEYWORDS

Drones; Authorization; Secure Function Evaluation

ACM Reference Format:
Dominik Roy George, Savio Sciancalepore, and Nicola Zannone. 2023. Privacy-Preserving Multi-Party Access Control for Third-Party UAV Services. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies (SACMAT '23), June 7–9, 2023, Trento, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589608.3593837

1 INTRODUCTION

The increasing availability on the market of Unmanned Aerial Vehicless (UAVs), namely, *drones*, is making it even more appealing to use them in various domains [5]. At the same time, regulations applying to drone flights and professional certifications, required to fly such devices, require users to spend significant time and money to use drones safely [28]. In this context, *Third-Party UAV Services*, also referred to as Drone-as-a-Service (DaaS), represents an emerging business model where Service Providers (SPs) lease ready-to-use drones, as well as certified pilots, customizable to the needs of any application scenario [6]. Today, DaaS providers are appearing on the market at a high pace, and recent reports show this trend will increase in the next years [7].

Although the DaaS paradigm significantly reduces the time and effort to integrate drones into business, it also introduces security and privacy concerns. Drones are managed by untrusted SPs, and they access resources owned by multiple parties. Thus, DaaS scenarios should use multi-party access control solutions, which can regulate access to resources through policies provided by multiple parties [27]. At the same time, such policies might contain sensitive information, e.g., relationships among Data Owners (DOs). If leaked, such information might raise privacy issues and, possibly, jeopardize business. Therefore, we need to adopt privacy-preserving multi-party access control solutions for DaaS scenarios.

To the best of our knowledge, no solutions are available today for privacy-preserving multi-party access control for constrained devices. Some solutions available in the literature cover traditional domains (see Sec. 3), but their usage within drones is not straightforward. Indeed, drones feature several constraints in terms of processing, communication, storage, and energy. At the same time, drones frequently operate in remote areas where Internet connectivity is scarce and no other entities are available to assist the processes. Therefore, currently-available solutions need to be re-engineered to fit the requirements of the DaaS scenario.

Contribution. Our work aims to investigate the feasibility of privacy-preserving multi-party access control schemes for Third-Party UAV Services. Our scheme builds on top of an existing approach based on the Secure Function Evaluation (SFE) paradigm [25], further adapting its architecture to meet the constraints and requirements of the DaaS scenario. Through the proposed scheme, drones can evaluate complex multi-party policies without relying on a persistent Internet connection and without inferring user policies specified by data owners. We deployed a proof-of-concept of the proposed scheme on a Raspberry PI 3 and performed an extensive experimental assessment to evaluate the performance of our approach w.r.t. several key metrics for constrained devices. The results show that our scheme can support privacy-preserving multi-party policy evaluation on a processing-constrained device in realistic configurations while requiring a very limited energy toll.

Roadmap. The remainder of the paper is organized as follows. Sec. 2 introduces our scenario, use cases, and requirements. Sec. 3 discusses related work. Sec. 4 provides the details of our solution,

SACMAT '23, June 7–9, 2023, Trento, Italy
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0173-3/23/06.
https://doi.org/10.1145/3589608.3593837

and Sec. 5 discusses security considerations. Sec. 6 reports an experimental evaluation of our solution using a real proof-of-concept implementation. Finally, Sec. 7 concludes the paper.

2 SCENARIO, USE CASES, AND REQUIREMENTS

In this section, we introduce our reference scenario and adversary model (Sec. 2.1). We also provide practical use cases (Sec. 2.2), which are used to extract the main requirements for achieving privacy-preserving multi-party access control in DaaS scenarios (Sec. 2.3).

2.1 Scenario and Adversary Model

The scenario considered in this work is based on the *Third-Party UAV service* architectural paradigm, referred to as DaaS. The DaaS paradigm provides advantages in several application domains, such as surveillance, farming, and construction [2], where individuals and companies cannot afford to buy drones, maintain them, and/or acquire the licenses necessary for flying UAVs.[1]

Without loss of generality, the DaaS paradigm enables a SP to lease one or more drones to its customers to achieve specific tasks, e.g., the inspection of specific equipment or areas [7, 29, 32]. The SP provides the drones and equips them with the necessary tools, sensors, and software applications, depending on the specific purpose the drone is intended for. Moreover, the SP can provide a professional pilot to run operations on the field. The drones and their sensors, hereby referred to as *resources*, are made available for usage and access to several Data Consumers (DCs). To interact with the drone and access resources, DCs use *controllers*, i.e., software applications that can be installed on the user equipment (such as a smartphone) and that allow interacting with the drone, in real-time, through a dedicated (secure) network connection.

Access to the resources provided by drones is regulated through access control policies specified by the DOs. Specifically, DCs can access data collected by the drone and integrated sensors only if the access control policy associated with the resources allows for it (see Sec. 2.2 for practical examples). In specific situations, the DC can also be a DO. DaaS applications may also include other entities, such as the Manufacturer, i.e., the entity that produced the drone, which can be different from the SP. However, given that such entities are not directly involved in the use cases tackled in our work, we do not focus specifically on their functions.

In this work, we focus on resources owned by multiple DOs, each with a different level of authority on the resource. Examples of such resources include data on areas affected by natural disasters, agriculture fields, harbors, and buildings under construction (see Sec. 2.2 for more details). In the IT domain, the protection of such resources is usually achieved through *multi-party access control* [21]. In this paradigm, each DO can specify access constraints on its resources (hereafter referred to as *user policy*), which define under which conditions a subject is allowed to access a given resource. When a DC requires access to a resource, the multi-party access control mechanism accounts for the user policies of all the owners of that resource to determine whether the authorization should be granted or denied, while handling possible conflicts among the DOs' user policies, e.g., based on their level of authority on the resource.

[1]Several countries have regulation in place (e.g., FAA regulations in the US [1]) requiring individuals and companies using UAVs to maintain a valid drone pilot license.

While the protection of co-owned resources is of utmost importance, *user policies* may also be sensitive. If revealed, these policies can leak confidential information, e.g., about the relationship among different DOs, the resource, and the specific DO [25]. The above considerations are even more relevant in the DaaS domain. Indeed, drones might be more easily hijacked and captured than traditional IT systems. Moreover, manufacturers often keep remote access to the deployed vehicles, e.g., for maintenance, and might stealthily access such policies when performing regular maintenance operations. Thus, the policy content should be protected, resulting in the need to deploy a privacy-preserving multi-party access control solution.

While solutions supporting privacy-preserving multi-party access control have been proposed for traditional IT systems [25, 26], their application and contextualization in the DaaS scenario poses additional challenges. Indeed, drones frequently operate in remote locations where connectivity is scarce or not available at all. Thus, drones cannot simply outsource operations to more powerful servers available on the Internet or other network nodes, but they have to perform all computations offline. Moreover, drones are energy-constrained devices, which should preserve energy as much as possible so as not to consume all the available energy too quickly, before the expected end of their planned mission.

2.2 Use Cases

This section presents concrete use cases to illustrate the need for privacy-preserving multi-party access control in DaaS applications. From such use cases, we elicit the requirements that such a solution should meet (cf. Sec. 2.3).

2.2.1 *Disaster Management.*

Disaster management plays a major role nowadays, and the DaaS paradigm is increasingly used in related rescue operations [4, 22].

Application Scenario. Assume a disaster scenario, e.g., a landslide or flooding, where several local forces (e.g., the local police, fire force, and municipality) and state forces (e.g., the military) are involved in rescue operations. Using a drone, such entities can better orchestrate their interventions and optimize the distribution of forces on the field. However, the relatively short duration of the operation might not justify the costs needed to buy and maintain a drone; rather, rescue forces can subscribe to a shared drone through the DaaS paradigm to coordinate their interventions. In this context, the military, local police, fire force, and municipality are both DOs and DCs. Moreover, other entities may require access to the drone and its resources, thus acting as DCs, to participate (in various capacities) in rescue operations, e.g., volunteers, paramedics, civil protection, and news channels. The resources to be accessed include readings of the surrounding environment, e.g., the camera (video feed), heat sensors (sensor data), and environmental sensor measurements, to identify victims and monitor the status of flooding or landslide.

Problem. Assume that, during the operations, the drone flies over a military base near the area where the natural disaster occurred. In such a situation, the military, acting in the capacity of DO, would like to deny access to the video feed and heat map to rescue forces, as they could reveal sensitive information about the base. Therefore, the military force might deploy an access control policy on the drone, explicitly denying access to the video feed to any other entity when the drone flies in the proximity of the military base. However, this policy might contain sensitive information, such as the location of the areas to be protected, which should not be disclosed to entities with access to the drone, including the SP.

As another example, consider the case where the drone discovers dead bodies. For privacy and safety purposes, the rescue forces (police and fire forces, which are DOs) want to deny the news channels access to the video stream. Indeed, not only private images could be leaked, but also news-media operators might crowd the area, obstructing rescue operations. At the same time, the knowledge about an access control policy enforcing such a condition on the drone would immediately reveal to the news channel why they cannot access the video stream.

Challenge. Enforcing privacy-preserving multi-party access control in this use case is particularly challenging. Natural disasters might occur in remote and temporarily hard-to-reach locations, where Internet connectivity is limited. Therefore, the drone cannot rely on a persistent Internet connection, e.g., to delegate policy evaluation to Cloud-based systems. As a result, access control policies should be evaluated in a private way offline, involving only the drone and, if present, locally-available user-owned resources, such as a controller, while dealing with a resource-limited environment.

2.2.2 *Smart Transport/Vessel Inspection.*

The DaaS paradigm can also support port authorities to optimize their inspection procedures while minimizing related operation and deployment costs.

Application Scenario. A harbor has port authorities responsible for handling incoming and outgoing ships, e.g., harbormaster, immigration bureau, and coast guard. We assume that the port authorities (DOs) feature a drone, operated according to the DaaS paradigm, used for inspection activities and other tasks, e.g., validating the crew's identity, inspecting shipment documents, and checking compliance with shipping regulations. In particular, the drone can be used by the harbormaster to coordinate the traffic of vessels and perform vessel inspections. Similarly, the immigration bureau can use the drone to acquire and process the visa documents and passports of crew members. At the same time, we assume that the coast guard runs periodic surveillance operations in the harbor area. To prevent disclosure of the activities performed during these operations, the coast guard may specify an access control policy that restricts the access to the video stream of the drone when it flies in the proximity of the operation area.

Problem. When the drone approaches a vessel for inspection, the harbormaster and the immigration bureau (DCs) might not be able to access the video stream of the drone, as the trajectory of the drone might cross the area currently patrolled by the coast guard (DO). By looking at the policies deployed by the coast guard on the drone, others DOs and the SP might learn sensitive information about the operations run by the coast guard, such as the date, time, and exact area in which such operations are performed.

Challenges. Vessel inspections typically occur some kilometers offshore, where persistent Internet connection is not available. Therefore, the drone should be able to make access decisions offline (i.e., while not being connected to the public Internet), resorting to the aid of only locally-available entities (e.g., a controller), while not featuring connection to any additional network element. In this setting, policy evaluation should not drain too much energy from the drone's battery, as returning to the harbor for battery recharge would result in high costs and delays in the harbor operations.

2.2.3 *Smart Construction.*

In smart construction, we envision that different companies (subcontractors) can rely on the DaaS paradigm to automatize work and reduce costs [14, 23].

Application Scenario. Assume the owner of a construction site contracts multiple subcontractors, specialized in specific areas of the site (e.g., plumbing, installing electricity lines). The site manager can provide the subcontractors with a drone, e.g., to install electricity lines. Additionally, for privacy reasons, the drone only features a connection to the private network of the construction site owner, thus minimizing the information transferred through the public network. Due to safety concerns, the contractor also enables external parties, such as the power line authority, to regulate the usage of some resources. For instance, the power line authority (DO) can specify access requirements to limit power usage by elements carried by the drone. Indeed, abusing electricity usage during line installation operations could lead to a power shortage or a blackout for the entire area. At the same time, the construction site manager (DO) can authorize external entities to access the real-time video feed of the drone. For instance, during the electrical installation, the authority can access the video streaming to oversee if the installed electricity lines comply with regulations. Thus, specific installation operations could be forbidden to the power line operator due to the constraints imposed by the authority regulating the power grid.

Problem. If access control policies are loaded and used in clear on the drone, all parties interacting with the drone can see the information contained in the user policies. At the same time, an adversary capturing the drone might access these policies. Thus, they would know why a specific entity using the drone cannot install the electricity lines, potentially leaking sensitive information. For instance, knowledge of constraints expressed by the power grid authority might be misused to achieve a blackout or shortage in the region.

Challenge. While performing the planned tasks, the drone can only connect to the construction site owner's company network, which might offer limited services. Also, connecting to such a network might require a VPN connection, which is typically hard to support on the drone because of the significant communication, processing, and energy resources it requires [3]. Therefore, to save energy and increase the drone's battery lifetime as much as possible, policy evaluation should be executed locally, either onboard or through the assistance of locally-available users' equipment, while preserving the confidentiality of user policies.

2.3 Discussion

The use cases in the previous section highlight the need for the adoption of a privacy-preserving multi-party access control scheme in the DaaS scenario. On the one hand, multiple entities (DOs) share ownership of resources and can specify access constraints to regulate their access (*Support for Multi-party Data Governance*). On the other hand, to enforce access control policies, the drone needs access to the DOs' policies and the information therein. Such policies, however, can be sensitive. If leaked, unauthorized entities might infer sensitive information about, e.g., the identity of the entities using the drone and their relations. For instance, unauthorized entities might retrieve the location of military bases (use case 2.2.1), spatio-temporal information about joint military operations (use case 2.2.2), and the restrictions on the usage of power lines (use case 2.2.3). Thus, DaaS use cases require an access control solution that allows the privacy-preserving evaluation of DOs' policies (*Policy Confidentiality*). In particular, the access control mechanism should preserve the confidentiality of the access constraints in the user policies (*Attribute Confidentiality*) while remaining capable of collaborative decisions. Moreover, it should prevent someone from linking the access decision to a specific user policy, which can be exploited to derive the user policies (*Decision Untraceability*).

However, the design of such an advanced security mechanism in DaaS applications comes with several constraints. First, drones involved in DaaS applications might not feature persistent Internet connection. Even when equipped with a Subscriber Identity Module (SIM) providing mobile Internet connection, the drone might operate in areas where Internet connectivity is temporarily out of service (e.g., soon after a natural disaster occurred, as in use case 2.2.1), is not available (e.g., offshore, as in use case 2.2.2), or where there are restrictions on the type of connection to be used (as in use case 2.2.3). As a result, drones cannot reliably outsource access control operations to the Cloud or any fully-trusted edge devices. Conversely, they need to evaluate access control policies and enforce access decisions offline, at most, with the assistance of a semi-trusted party available in the field (*Outsourced Third Party*).

In addition, although significant processing and storage resources might be available, drones are typically energy-constrained devices, which should be able to complete all planned tasks before battery exhaustion. For instance, interrupting their tasks to re-charge the battery might cause loss of lives as in use case 2.2.1, economic losses as in use case 2.2.2, and construction delays as in use case 2.2.3. Therefore, the enforcement of privacy-preserving multi-party access control should be computationally *lightweight*, i.e., it should not have a large impact on the time necessary to access resources and the overall lifetime of the drone (*Constrained Environments*).

In Sec. 3, we evaluate to what extent the solutions available in the current literature consider and address the requirements mentioned above, motivating the need and novelty of our solution.

3 RELATED WORK

Several works investigated privacy-preserving access control solutions in the Internet of Things (IoT) domain. We summarize existing proposals w.r.t. the requirements identified in Sec. 2.3 in Table 1.

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) schemes are often proposed to protect resources from unauthorized access in the IoT domain. However, these schemes typically do not protect policy confidentiality. Several works have addressed this gap and proposed schemes that extend CP-ABE with other techniques to also ensure the confidentiality of the policies used to protect the resources. For instance, Kominos et al. [16] propose a privacy-preserving attribute-based encryption scheme that combines CP-ABE with pseudonyms to anonymize identifying attributes in the policy. Nishide et al. [20] combine CP-ABE with wildcards to protect attribute confidentiality. In this scheme, an encryptor uses wildcards to hide attributes that are not relevant to the ciphertext policy. This way, the decryptor does not learn the entire policy but only the attributes that are not "hidden behind the wildcards", thus only partially achieving attribute confidentiality. Zhang et al. [33] combine CP-ABE with hashing to hide the attributes defined in the policy. Other works propose schemes that combine CP-ABE with Linear Secrete Sharing Scheme (LSSS) [13, 31]. For instance, the scheme in [13] uses a one-way anonymous key agreement protocol to anonymize the attributes in the policy and stores the attribute mapping in an LSSS matrix. The matrix is then secretly shared with different parties. An authorized DC needs at least a given number of shares to reconstruct the matrix, and thus, to securely authenticate the attributes if the anonymized ones are present in the matrix. The DC can decrypt the ciphertext only if the anonymized attributes contained in the matrix are securely authenticated, without knowing the entire set of anonymized attributes in the LSSS matrix. A similar solution is proposed by Kan et al. [31], using a Garbled Bloom Filter (BF) to mask the attributes in the LSSS matrix, so that the decryptor is not able to infer the policy used to protect the resource. To achieve attribute confidentiality, Yang et al. [19] propose a solution based on Ciphertext-Policy Attribute-Based Signcryption (CP-ABSC), an attribute-based encryption and signing scheme, in combination with LSSS and Cuckoo Hashing (CH).[2] This solution allows the DO to sign and encrypt a resource using two different policies defined over different sets of attributes hidden in the LSSS matrix using CH.

Although these approaches protect the attributes in the policy attached to the encrypted resource, they are not fully suitable for DaaS applications. First, they can only deal with resources that are disclosed to the DC and do not support the protection of resources that can be only accessed on the drone (e.g., sensors). Moreover, the cryptographic operations underlying CP-ABE and CP-ABSC (e.g., pairing-based decryption, attribute generation, and certificate management) are computationally expensive and, thus, existing schemes tailored to constrained devices typically propose to outsource their computation to a third party. Such outsourcing, however, might not be possible in DaaS applications since persistent Internet connectivity is often unavailable. In addition, the third party is required to be fully trusted. By contrast, in our context, drones operate in unknown and untrusted environments, where they cannot rely on any fully-trusted party. Such schemes also assume that the resources to be protected are governed by a single entity. Although multi-authority-based encryption schemes are available [8, 9], they do not support the protection of co-owned resources. To the best of our knowledge, there are no CP-ABE schemes supporting multi-party data governance.

Another body of research investigated the problem of privacy-preserving policy evaluation using Homomorphic Encryption (HE). For instance, Kapadia et al. [15] use HE together with proxy encryption to preserve attribute confidentiality. The data owner encrypts the attributes in the policy with its public key by using the ElGamal cryptosystem. The server checks whether the attributes of the data consumer match with the encrypted ones and "homomorphically multiplies" the matching encrypted attributes with random values, which are disclosed along with the ciphertext to the client. The DC

[2]CH is similar to BF, which supports the dynamic insertion and removal of elements in a bit array, thus being more efficient and with a lower false-positive rate.

Table 1: Comparison of privacy-preserving access control approaches.

	Crypt. Tech.	Outsourced Third Party		Support for Multi-party Data Governance	Policy Confidentiality		Constrained Environments
		Fully-Trusted	Semi-Trusted		Attribute Confidentiality	Decision Untraceability	
Kominos et al. [16]	CP-ABE	●	○	○	●	○	○
Nishide et al. [20]	CP-ABE	●	○	○	◐	○	○
Zhang et al. [33]	CP-ABE	●	○	○	●	○	○
Fan et al. [13]	CP-ABE	●	○	○	●	○	○
Kan et al. [31]	CP-ABE	●	○	○	●	○	○
Ming et al. [19]	CP-ABSC	●	○	○	●	○	○
Xu et al. [30]	HE	○	●	○	●	○	○
Kapadia et al. [15]	HE	○	●	○	●	○	○
Alishahi et al. [25, 26]	HE; SFE	○	●	●	●	●	○
Our scheme	SFE	○	●	●	●	●	●

decrypts the attributes to decrypt the ciphertext without learning the individual values of the attributes associated with the ciphertext. Xu et al. [30] propose a privacy-preserving Attribute-Based Access Control (ABAC) scheme based on the ElGamal cryptosystem to protect user attributes. This scheme allows a DC to disclose its sensitive attributes in an encrypted form, which are evaluated by the server against an ABAC policy using a privacy-preserving policy matching component without gaining any information about the DC's attribute. Although this scheme does not require a trusted third party to protect user attributes, it only supports the privacy-preserving evaluation of clauses containing only one comparison operator.

These proposals are not suitable for the DaaS scenario because they assume resources owned by a single entity. To the best of our knowledge, the only privacy-preserving solutions based on HE supporting a multi-party governance model are the ones in [25, 26]. Alishahi et al. [26] propose a framework for the privacy-preserving evaluation of multi-party access control policies. The framework allows every party to independently specify policies regulating the access to their resources, which are deployed in the resource server in an encrypted form. The (encrypted) policies are combined according to a predefined governance model and evaluated under encryption, thus preserving the confidentiality of local policies and ensuring decision untraceability. The work in [25] extends [26] by supporting the privacy-preserving evaluation of multi-party access control policies expressed in ABAC. The works in [25, 26] also investigate the use of SFE as an enabler for the privacy-preserving evaluation of multi-party access control policies, showing that SFE provides a more efficient solution compared to HE both in terms of computation time and bandwidth usage. Differently from schemes based on CP-ABE and CP-ABSC, which require a fully-trusted third party when operating on constrained devices, solutions based on HE and SFE only require a semi-trusted party to decouple access control operations, thus preserving policy confidentiality.

Overall, as shown in Table 1, the work that fulfills the most requirements is [25]. However, such a proposal cannot be directly integrated into the DaaS scenario. Such a scheme was proposed to address privacy issues in Online Social Networks (OSNs), where persistent Internet connectivity is always available, and computational capabilities are always at the disposal of both the client and the server. Conversely, in the DaaS scenario, the drone should make access control decisions offline, without connecting to the Internet or featuring a persistent reliable connection to a server in the Cloud. At the same time, the drone should also minimize energy consumption to increase lifetime. Our solution addresses these considerations, contextualizing the solution proposed by [25] in the DaaS scenario and mitigating its limitations in terms of processing and suitability for constrained environments (non-persistent Internet connection and energy limitations). We realized a prototype of our solution and performed a thorough experimental performance evaluation, showing its viability in the DaaS domain.

4 PRIVACY-PRESERVING MULTI-PARTY ACCESS CONTROL IN DAAS

A main problem of existing multi-party access control solutions is that they typically do not account for the protection of the policies, whose disclosure can leak confidential information as well. This problem is addressed in [25], which proposes a framework for the privacy-preserving evaluation of multi-party access control policies. In this work, we rely on that framework for the specification of multi-party access control policies and for the cryptographic primitives for their secure evaluation; based on it, we propose a privacy-preserving multi-party access control architecture for the DaaS scenario. We first present an overview of the framework in [25] (Sec. 4.1); we then describe the architecture of our solution (Sec. 4.2) and the protocol flow (Sec. 4.3).

4.1 Privacy-Preserving Policy Evaluation via Secure Function Evaluation

Our work is built on top of the framework for privacy-preserving multi-party access control proposed in [25]. This framework allows users to provide their policies, expressed in ABAC, in private form and provides secure computation protocols for evaluating multi-party policies, preserving the confidentiality of the user policies forming the multi-party access control policy.

The authors of [25] realized their framework and secure protocol using the SFE paradigm [10]. SFE allows several parties to compute a function on their private inputs without revealing any information besides the result of the function. Given two parties p_1, p_2 and their corresponding inputs x and y, each party creates secret shares for their input, represented as $\langle x\rangle_1$, $\langle x\rangle_2$ and $\langle y\rangle_1$, $\langle y\rangle_2$ respec., such as $\langle x\rangle_1 \oplus \langle x\rangle_2 = x$ and $\langle y\rangle_1 \oplus \langle y\rangle_2 = y$. Each party shares one of the input shares with the remote party and evaluates a given function f using the shares in its possession. The actual result of the function ($z = f(x, y)$) can then be reconstructed by combining the shares of the result, $\langle z \rangle_x$ and $\langle z \rangle_y$, obtained by applying f to the input shares, i.e., $z = \langle z \rangle_x \oplus \langle z \rangle_y$.

By leveraging the SFE paradigm, the work in [25] devises secure computation protocols for target evaluation and policy composition of ABAC policies. These protocols are used for the privacy-preserving evaluation of *multi-party access control policies*. Intuitively, *multi-party policies* are created from the shares of *user policies*, which specify the DOs's access constraints for their resources, according to a *policy template* defining how user policies should be combined based on the relationship between the DOs and the resource. When receiving an access request, the server and a Semi-Trusted Party (STP) evaluate the request, each using its share of the multi-party policy. The access decision is obtained by combining the shares of the decision computed by these entities on their shares of the multi-party policy and request.

Although the proposal in [25] addresses privacy concerns in multi-party settings, it has not been considered or applied in constrained scenarios, where the involved entities are characterized by computational, communication, storage, and energy constraints, and feature a non-persistent Internet connection. We take such a step ahead through our work, as described in the rest of the section.

4.2 Architecture Overview

Fig. 1 presents the architectural view of the proposed privacy-preserving multi-party access control system for the DaaS scenario. The architecture comprises five main entities: the DOs, the SP, the STP, the controller, and the drone.

- The *DOs* are the entities that own the resources in the system. We specifically consider a *multi-owner* scenario, where multiple owners might share possession of a specific resource. In this context, each DO defines *user policies* (in the form of ABAC policies), specifying the access constraints a requester should fulfill to access the protected resource. We assume that DOs are known in advance.
- The *SP* is the entity responsible for the services offered by the drone and for the enforcement of DOs' access constraints.
- The *STP* is a semi-trusted party that assists the SP in the privacy-preserving evaluation of multi-party access control policy.
- The *controller* runs on a device possessed by the DC, equipped with (wireless) communication capabilities, e.g., a mobile phone. Within the controller, we identify two main independent and isolated logical *spaces*: the *client space* and the *STP space*. The client space encompasses the applications used by the DC to access the resources on the drone. The STP space is controlled by the STP and comprises a Policy Decision Point (PDP) aimed to support the privacy-preserving evaluation of multi-party policies.
- The *Drone* is the entity possessing the resource(s) requested by the DC client. When deployed for the planned mission, the drone communicates persistently with the controller, and it does not feature a persistent Internet connection. To protect its resources and services, the drone includes a *Access Control Module* (ACM), which comprises a *Policy Enforcement Point (PEP)*, a *PDP*, a *Policy Information Point (PIP)*. The PEP is responsible for the interaction with the DC's application and enforcing the access decision; the PDP evaluates the DC's requests (with the assistance of the PDP in the STP space); the *PIP* gathers environmental attributes (e.g., from sensors) needed for policy evaluation.

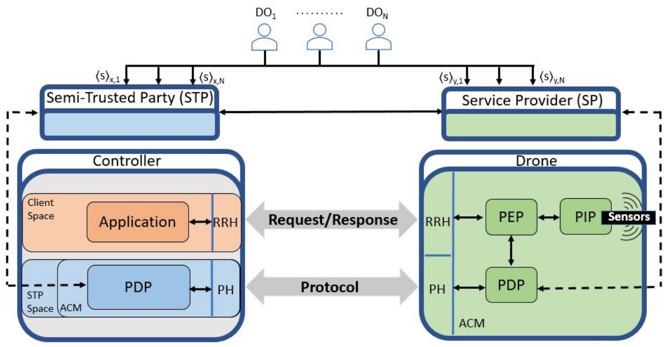

Figure 1: Architecture of the Privacy-Preserving Multi-Party Access Control scheme for the DaaS scenarios.

The DOs, STP, and SP interact offline during the *setup phase* to configure the access control policies for the protection of the drone's resources, such as data and physical capabilities provided by the drone. The DOs create two shares of their *user policy* (denoted as $\langle s \rangle_x$ and $\langle s \rangle_y$ in Fig. 1) and provide one share to the SP and the other to the STP. From these shares, the SP and STP separately construct a *multi-party policy*, specifying the permissions to be enforced on drone resources, based on a predefined *policies templates*.[3] The shares of the *multi-party policy* are then deployed by the SP and STP into the drone and STP space on the controller, respectively.

When operating in the field, a DC might request access to resources or services offered by the drone (through the applications installed in the controller). Access to these resources and services is determined *at runtime* in a privacy-preserving way through the interaction of two entities: the controller and the drone. In particular, the PDPs in the drone and in the STP space of the controller evaluate the request against the share of the multi-party policy, independently. The drone recombines the computed shares of the decision to reconstruct the final access decision to be enforced. Note that the whole protocol flow is executed locally between the controller and the drone, without an online connection to the public Internet.

The communication between the controller and the drone is managed through two interfaces. The Request/Response Handler (RRH) manages the interaction between the application in the client space and the drone, whereas the Protocol Handler (PH) enables the interactions between the drone and the STP PDP for privacy-preserving policy evaluation.

Security Assumptions. We assume an *honest-but-curious* threat model, where the entities behave according to the protocol specification, but they are interested in obtaining more information from their input, intermediary messages and output. In addition, we assume that the SP and the STP are independent entities, which do not collude. We will discuss the implications of collusion attacks in Sec. 5. We also assume that the *client space* and the *STP space* deployed on the controller are independent and do not interact with each other. This can be achieved, e.g., by using containerization techniques that provide built-in isolation among processes.

[3]As in [25], we assume that the policy template is defined by the SP, in agreement with the DOs, based on their relationship with the resources to be protected.

4.3 Protocol Details

Our scheme includes two phases: *setup phase* and *online phase*. A sequence diagram representing the schema is presented in Fig. 2.

Setup Phase. This phase, executed offline before the deployment of the DaaS, involves the following steps:

(1) The DOs generate secret shares of their user policies, i.e., $\langle s \rangle_x$, $\langle s \rangle_y$.
(2) Using a secure connection over the public Internet, the DOs provide one secret share, i.e., $\langle s \rangle_x$, to the SP.
(3) At the same time, the DOs provide the other secret share, i.e., $\langle s \rangle_y$, to the STP using another secure connection.
(4) Using a secure channel, the SP sends to the STP the *policy template* used to construct the *multi-party policy* using DOs' shares.
(5) The STP deploys the multi-party policy into the controller.
(6) Similarly, the SP deploys the multi-party policy into the drone.

Online Phase. In this phase, the DC's controller and the drone interact to determine whether access to the requested resources should be granted. This phase involves the following steps:

(1) The application hosted in the *client space* of the controller sends a request to the drone for accessing a resource.
(2) At reception of the request, the *Drone PEP* requests the *Drone PIP* to provide environmental attributes for policy evaluation.
(3) The *Drone PIP* retrieves the environment attributes from the drone's sensors and sends them to the *Drone PEP*, which augments the access request with those attributes.
(4) The *Drone PEP* forwards the request to the *STP PDP*, which generates two shares of the request, $\langle q \rangle_x$ and $\langle q \rangle_y$, using a share generation algorithm. One share, $\langle q \rangle_x$, is kept on the drone; the other, $\langle q \rangle_y$, is meant for remote evaluation on the STP.
(5) The drone forwards the share $\langle q \rangle_y$ to the *PDP* in the STP space.
(6) The *Drone PDP* and the *PDP* in the STP space evaluate their shares of the request using their shares of multi-party policy, resulting in the shares of the decision, namely, $\langle d \rangle_x$ and $\langle d \rangle_y$ resp.
(7) The *STP PDP* sends its share of the decision, i.e., $\langle d \rangle_y$, to the *Drone PDP*.
(8) The *Drone PDP* re-combines the shares $\langle d \rangle_x$ and $\langle d \rangle_y$, obtaining the final decision d.
(9) The final decision d is sent to the *Drone PEP* for its enforcement.
(10) If the decision is *permit*, the drone grants access to the resource(s). Otherwise, if the decision is *deny*, the drone notifies the requesting application that access is denied.

5 SECURITY CONSIDERATIONS

In the following, we discuss the security properties of the proposed scheme. Overall, the discussed approach is secure against an *honest-but-curious* adversary. This means that the involved parties (DC, DOs, STP, and SP) follow the protocol steps and, at the same time, aim to obtain sensitive information from the exchanged messages. In this context, we can see that none of the involved entities can learn information about the *user policies*. Specifically:

- The SP cannot derive the DOs' *user policies*. This is because the SP knows only one share of the *user policies*, which is insufficient to obtain the user policies thanks to the SFE properties [10]. At the same time, the SP is the entity deploying and managing the drone. Thus, it knows the final decision and the specific request issued by the *application*. This is necessary as the drone needs to provide

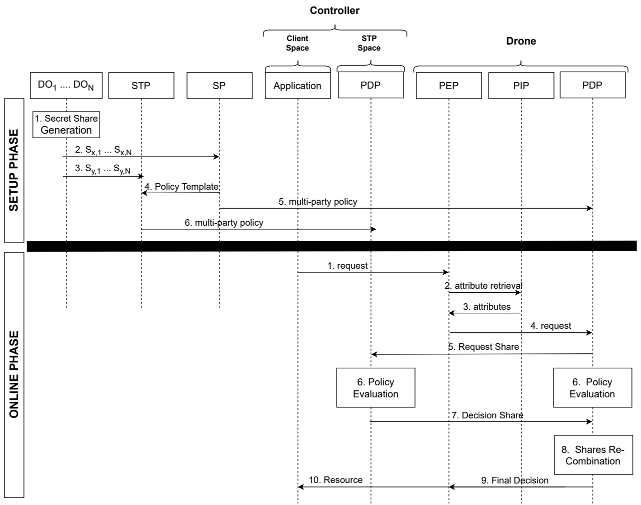

Figure 2: Sequence diagrams of the *Setup Phase* and *Online Phase* of the proposed scheme.

access to the requested resource. However, this information is insufficient to derive the individual *user policies*.
- The STP cannot derive the *user policies* nor the final access decision. As we assume the *STP space* and the *Client space* in the controller are isolated, the STP has only one share of the *user policy* and the *policy template*, not enough to infer the *user policies*.
- On the controller, the *application* knows the final decision. However, the *application* cannot infer the individual *user policies* from it as it does not have any share. However, we highlight that the *application* might attempt to reconstruct the *multi-party policy*, e.g., by issuing all possible requests. However, we highlight that, in our scenario, issuing all possible requests might be difficult, if not impossible, as attributes are retrieved by the PIP at run time.

It is also worth discussing the impact of collusion attacks on the security of our scheme.

Collusion between SP and DOs. Some DOs might collude with the SP to learn the policies of the other DOs. However, the DOs and the SP combined have only one share of the non-colluding DOs' *user policies*. Thus, they cannot derive any information about these policies from intermediary messages.

Collusion between STP and DOs. Even in the case where all but one DOs collude with the STP, they cannot infer the *user policies* of the honest DO, as the STP does not know the final decision. Thus, the STP and the colluding DOs cannot compare such a decision with the known *user policies* to derive the unknown ones.

Collusion between SP and *application*. Neither the SP nor the *application* has access to the other share of DOs' user policies. Thus, even if they collude, they cannot learn them. Both entities know the final decision; thus, neither of them learns new information.

Collusion between STP and *application*. Similarly to the previous case, the STP and *application* do not have both the shares of the *user policies*. Thus, they cannot infer them. By colluding with the *application*, the STP can learn the final decision. Yet, the STP cannot derive DOs' user policies from it.

6 EVALUATION

We evaluated our scheme, presented in Sec. 4, based on its performance and feasibility in the DaaS scenario. Specifically, we are interested in answering the following three research questions:

- **RQ1:** How the level of policy granularity in the scenario affects the communication, processing, and energy cost of the proposed privacy-preserving multi-party policy evaluation scheme on a constrained drone and controller?
- **RQ2:** How the number of stakeholders in the scenario affects the communication, processing, and energy cost of the proposed privacy-preserving multi-party policy evaluation scheme on a constrained drone and controller?
- **RQ3:** How does the proposed scheme perform compared to alternative privacy-preserving solutions available in the literature?

DaaS scenarios are typically highly dynamic. Thus, they require solutions able to adapt to different contexts. In the field of access control, this is usually achieved by employing fine-grained policies, which give users more control over the data. However, supporting a fine granularity often requires access requests to include a large number of attributes for their evaluation, which might impact system performance. **RQ1** investigates the impact of policy granularity (represented as the number of attributes to be evaluated) on the performance of the proposed scheme in terms of processing, communication, and energy costs. Moreover, the number of stakeholders in the scenario (e.g., DOs) may impact system performances, which is the focus of **RQ2**. The number of stakeholders managing the resources is typically reflected in the complexity and size of the access control policy, where larger and more complex policies might require higher execution time, communication overhead, and energy cost for policy evaluation. For both **RQ1** and **RQ2**, we are interested in such performance costs on both the constrained drone and the controller hosting the STP. Knowing beforehand the protocol's demands on both entities can help manufacturers and SPs deploy devices with the necessary processing, communication, and energy capabilities to achieve the planned tasks and minimize deployment costs. In this work, we employ SFE as the underlying privacy-preserving technique for the proposed scheme. **RQ3** aims to assess this choice by comparing our solution against the use of other privacy-preserving techniques available in the literature, applied in traditional IT domains.

6.1 Proof-of-Concept Implementation and Deployment

To answer our research questions, we implemented a proof-of-concept of our proposed solution.[4] For our implementation, we got inspiration from the work in [25], which presents a prototype implementation of their approach for the privacy-preserving evaluation of multi-party access control policies. Their prototype relies on the ABY framework [10] as the main building block. In a nutshell, the ABY framework provides basic modules to design secure two-party computation protocols using Arithmetic, Boolean circuits, and Yao's garbled circuits. However, the prototype in [25] was used to evaluate the performance of the approach using a fully local simulation running on a single machine. In this respect, it does not support the pre-sharing of the user policies and remote connections, as required by the DaaS scenario.

For our proof-of-concept, we followed the same logic of [25] but provided additional functionalities to support a network environment compliant with the DaaS scenario. In particular, we (i) organized the code into several processes, (ii) distributed such processes on several hardware platforms, and (iii) enabled remote communication capabilities using communication sockets. In this context, a major improvement of our implementation is the adoption of the *PutINSharedGate* available from the ABY framework instead of using the *PutINGate* as in [25]. Specifically, using the *PutINSharedGate* allows us to compute and pre-deploy the shares of the user policies during the setup phase rather than providing these policies in plain-text during the online phase, as was done in [25]. To achieve privacy-preserving multi-party policy evaluation, we used 32-bit Boolean circuits and configured the ABY framework to provide a security level of 114 bits.

To answer RQ3, we additionally implemented another proof-of-concept based on the Partially Homomorphic Encryption (PHE)-based protocol proposed in [25]. In this regard, we also considered using Fully Homomorphic Encryption (FHE) in the benchmark protocol. At a first glance, applying FHE might appear a suitable choice for our scenario due to the well-known properties of FHE to carry out both additions and multiplications in the encrypted domain. However, we discarded such an option as, to preserve policy confidentiality, an equivalent scheme based on FHE would also require the deployment of an online third party (not to let the drone know the user policies). When such a third party is involved in the protocol, a solution based on FHE would require more computational overhead than a solution based on PHE due to the well-known high processing toll of FHE schemes. For implementing PHE-based operations, we used the GMP multiprecision library, in the C++ programming language. To enable a fair comparison with our approach, we improved the original implementation using actual remote sockets. Thus, the results reported in Sec. 6.5 consider the same deployment of our proposed approach and the competing solution in terms of hardware, software, and communication setup.

We deployed our proof-of-concept implementations on a real heterogeneous network. To emulate the drone, we used a Raspberry PI 3 Model B+ [24]. This is an embedded platform equipped with a Cortex-A53 processor running at 1.4 GHz, 1GB of RAM, and 16GB of storage. As acknowledged in the literature [17, 18], the computational and bandwidth resources offered by the Raspberry PI are comparable to the ones provided by low-end commercial drones, and, recently, some commercial drones integrating a Raspberry PI as an on-board CPU also became available on the market [12]. Thus, using the Raspberry PI allows us to evaluate our solution in a realistic setting.

On the other hand, we considered two deployment setups for the controller hosting the STP. In the first setup, we used a regular laptop Lenovo Thinkpad P1, equipped with two Intel Core i7-8750H processors running at 2.20 GHz, 32GB of RAM, and 1 TB of SDD Storage, while in the second we used a Raspberry PI 3 Model B+. Hereafter, we refer to the two setups as *Testbed 1* and *Testbed 2*, respectively. These testbeds allow us to investigate the system performance when heterogeneous hardware is used as a controller. When run on the laptop, the STP executes in a Docker virtual environment, while it executes as a native process when run on the Raspberry Pi.

[4]The source code is available at `https://github.com/DominikRoy/PP-DaaS`.

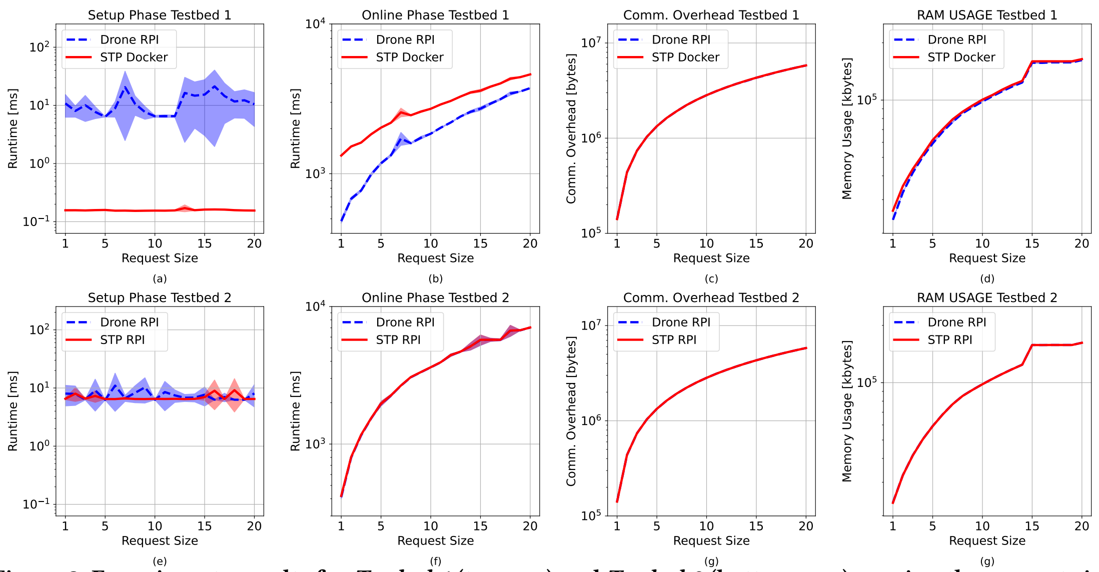

Figure 3: Experiments results for *Testbed 1* (top row) and *Testbed 2* (bottom row) varying the request size

From the communication perspective, the Raspberry Pi modelling the drone and the STP (running either on another Raspberry Pi or on the laptop) communicate over an ad-hoc WiFi network.

6.2 Evaluation Metrics

To assess the practical feasibility of our scheme for the DaaS scenario, we evaluated our proof-of-concept implementations in terms of processing, communication, and energy costs.

We measured the processing costs for evaluating an access request in terms of the execution time of the setup phase (in milliseconds), the execution time of the online phase (in milliseconds), and RAM occupancy (in kilobytes). To carry out time measurements, we leveraged the integrated measurement system provided by the ABY framework. The ABY framework provides built-in benchmarking routines to measure the execution time of its various phases (setup, circuit generation, garbling, OTextension, online). However, these phases do not precisely map to the phases of our scheme. Thus, for measuring the execution time of the setup phase of our scheme, we considered the combined time required for the initialization of the ABY framework (setup), the circuit generation, and the garbling (encryption of the circuit). For the online phase of our scheme, we considered the sum of the time to execute the online phase of the ABY framework and *OTExtension*. Note that the *OTExtension* falls into the online phase of our scheme since this process concerns the generation and distribution of the shares of the access request. To measure RAM occupancy, we used the Linux tool "/usr/bin/time -v", which provides the highest instantaneous RAM consumption of a given process. Such a metric allows us to determine the maximum RAM demands of the proposed approach, thus providing an indication of the necessary RAM to be deployed on the involved devices.

We evaluated communication costs by measuring the communication overhead introduced by our scheme (in bytes). To this end, we leveraged the integrated measurement system provided by the ABY framework, which reports the summation of bytes sent and received during the setup phase, online phase, garbling (encryption of the circuits), and OTtransfer, for both the client and server.

We also measured the energy consumption (in milliJoule) required by the drone and the STP to evaluate an access request. To this end, we used *powertop*, a software-based energy estimation tool provided by the Linux OS. *Powertop* estimates the power (in Watt) of a running process. Then, to compute the energy consumption of the process, we multiply such a power estimate for the (measured) duration of the online phase, as: $E = P \times \Delta t$, being E the estimated energy in Joule, P the power (in Watt) estimated by *powertop* and Δt the duration of the online phase.

6.3 Experiment 1: Impact of the Request Size

This experiment aims to answer **RQ1** by evaluating the impact of policy granularity (represented in terms of request size) on the relevant system performance metrics described in Sec. 6.2 on both the drone and the controller.

Settings. For this experiment, we fixed the policy size to 10 and increased the request size (i.e., the number of attributes provided in the access request) from 1 to 20. For each request size, we generated 200 different access requests. In the results, we report the average for every performance metric.

Results. Fig. 3 reports the average execution time of the setup and online phases, communication overhead, and RAM required by the proposed scheme using the log scale on the y-axis while increasing the request size (x-axis). We also show the 95% confidence interval of the measurements through shaded areas. The figure is organized in two rows and four columns, where the top row refers to the results obtained by running the STP as a docker instance on the laptop (*Testbed 1*), and the bottom row refers to the measurements obtained by running the STP on a Raspberry PI (*Testbed 2*).

We observe that increasing the request size results in an increase of all performance metrics, except for the execution time of the *setup phase*, which remains almost constant regardless of the request size.

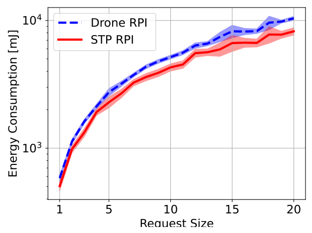

Figure 4: Experimental energy consumption of the controller and drone in *Testbed 2*, varying the request size.

Specifically, the *setup phase* takes between 6 and 20 ms on the Raspberry Pi, whereas it takes between 0.15 and 0.17 ms on the laptop.

The execution time of the *online phase* increases with the request size for both testbeds. On *Testbed 1*, the drone takes, on average, 482 ms for completing the *online phase* with a request size of 1, and the execution time grows to 3, 730 ms for requests of size 20. On the other hand, the STP takes, on average, 1, 322 ms for completing the *online phase* with requests of size 1 and 4, 604 ms for requests of size 20. We believe this result, which might appear surprising, is due to the working logic of Docker containers, which cannot efficiently allocate enough resources to the process. The drone and the STP require the same amount of communication overhead and RAM, ranging from 0.14 and 26 Mbytes, for a request size of 1 to 5.8 and 162 Mbytes, respectively, for a request size of 20. On *Testbed 2*, the drone and the STP take approximately the same time for completing the *online phase*, ranging from 410 ms for requests of size 1 to 7, 045 ms for requests of size 20. Moreover, the drone and the STP require the same amount of communication overhead and RAM required in *Testbed 1*. Comparing Fig. 3(b) with Fig. 3(f), we notice that the drone in *Testbed 2* always requires more time than in *Testbed 1*. Conversely, the STP in *Testbed 2* requires less time than the STP in *Testbed 1* when the request size is lower than 5, but its performance degrades faster for higher values of the request size.

We also measured the energy consumption of our scheme for *Testbed 2* varying the request size.[5] The results are reported in Fig. 4. In line with the results in Fig. 3(f), the energy consumption of the involved entities increases with the increase of the request size. To provide a few examples, for requests of size 8, the drone requires approx. 4, 338.71 mJ, while the STP consumes 3, 603.17 mJ. Such a consumption on the drone translates into 0.158 mAh. Considering that an actual drone such as the DJI Mini 2 features a battery with a capacity of 5, 200 mAh [11], such a consumption corresponds to approx. 0.003% of the overall drone battery capacity, with minimal impact on the drone's lifetime.

6.4 Experiment 2: Impact of Policy Size

The number of stakeholders involved in the management of resources influences the size and complexity of the policies to be evaluated and, thus, system performance. This experiment aims to investigate such impact by assessing how the increase in policy size affects the relevant system performance metrics described in Sec. 6.2 on both the drone and controller.

[5]For *Testbed1* we did not measure the energy consumption, because *Docker* does not has the kernel access for *powertop* to read the energy estimates.

Settings. We fixed the request size to 10 and increased the policy size (i.e., the number of atomic access constraints in the policy) from 1 to 50. For each policy size, we generated 200 different policies. In the results, we report the average for every performance metric.

Results. Fig. 5 reports the average time of the *setup* and *online* phases, communication overhead, and RAM consumption of the proposed solution when increasing the policy size in *Testbed 1* (top row) and *Testbed 2* (bottom row). As for the previous experiments, we also report the 95% confidence interval of all measurements.

Overall, Fig. 5 shows a similar trend compared to the results reported in Fig. 3. All the metrics under investigation increase with the increase of the policy size, but the time for the *setup phase*, which remains constant in the range between 6 and 20 ms.

As for the execution time of the *online phase*, we notice that it increases faster for policies of size between 1 and 10 compared to larger policies. On *Testbed 1*, the drone takes, on average, 1, 849 ms for evaluating a policy size of 6 while requiring a communication overhead of 1.69 Mbytes and 70.4 Mbytes of RAM. At the same time, the STP takes, on average, 2, 864 ms, in line with the findings of Fig. 3. On *Testbed 2*, the drone and the STP take, on average, 2, 378 ms for evaluating a policy size of 6 while requiring the same amount of communication overhead and RAM of *Testbed 1*. As for the previous experiment, we believe these results are acceptable for applying our solution in real settings, as they do not degrade too much the experience of the users and the quality of the offered services. On the other hand, considering the worst-case of a policy size of 50, the *online phase* of the scheme requires an execution time of 12, 619 ms on the drone and 13, 595 ms on the STP (*Testbed 1*).

We also measured the energy consumption of our scheme for *Testbed 2* varying the policy size. The results are reported in Fig. 6. Similarly to the results in Fig. 4, the energy consumption increases with the increase of the policy size. With a policy of size 6, the drone requires approx. 3, 308.98 mJ. Yet, considering the DJI Mini 2, such a consumption corresponds to 0.0023% of the overall drone battery capacity. Such a limited energy toll confirms the minimal impact of our solution on the drone's lifetime.

6.5 Experiment 3: Comparison

In this experiment, we compare the solution described in Sec. 4 against the approach based on PHE proposed in [25].

Settings. In this experiment, for each of the two selected approaches, we fixed the request size to 10 and increased the policy size from 1 to 12. For each policy size, we generated 200 different policies. In the results, we report the average values of each measurement together with the 95% confidence intervals, depicted through shaded areas.

Results. Fig. 7 reports the results of our investigation, focusing on the time required in the online phase by the two approaches, the communication overhead and RAM occupancy.

The results show that the approach proposed in [25] requires, on average, two orders of magnitude more time than our solution. At the same time, the solution in [25] requires less communication overhead than ours, especially on the drone. This is due to the communication overhead introduced by the *OTExtension*, which is used by our scheme during the online phase to generate and exchange the shares of the circuit between the involved entities. Finally, we notice that our solution also requires more RAM than [25]. Indeed,

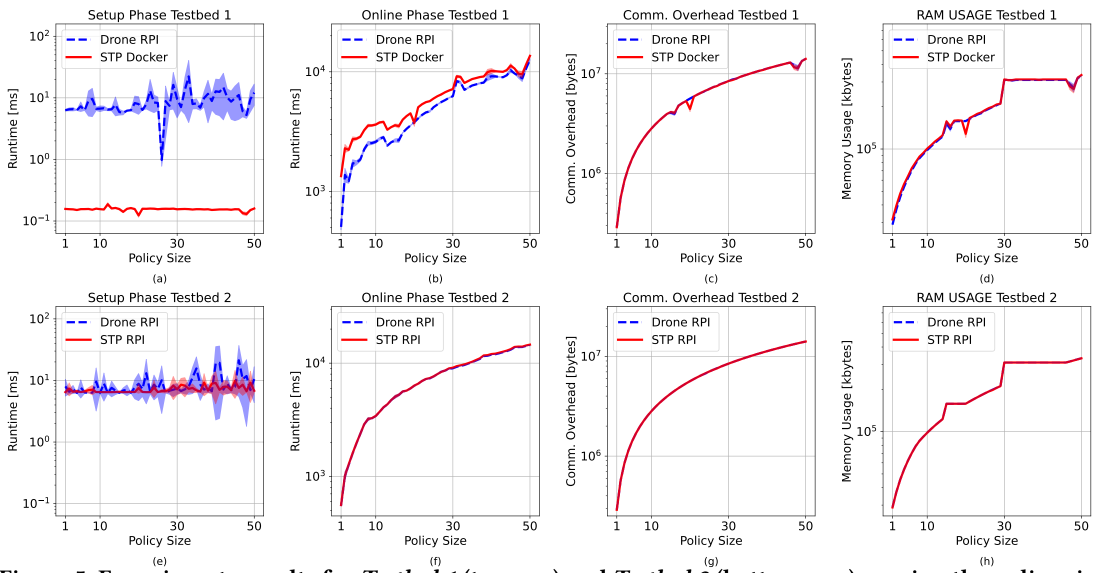

Figure 5: Experiments results for *Testbed 1* (top row) and *Testbed 2* (bottom row) varying the policy size

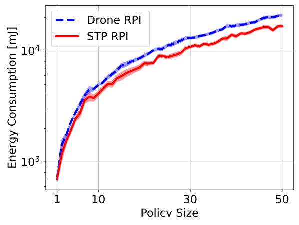

Figure 6: Experimental energy consumption of the controller and drone in *Testbed 2*, varying the policy size.

building, maintaining, and executing operations on circuits requires more RAM than performing additions in the encrypted domain.

6.6 Discussion

The results of our experiments show that system performance degrades when the size of requests and policies increases. Although in extreme cases (request size 20 and policy size 50) the privacy-preserving evaluation of multi-party access control policies on constrained devices is challenging, the experiments demonstrate that the proposed scheme is feasible in real-world DaaS scenarios when a reasonable number of attributes needs to be evaluated and when reasonably complex policies are deployed. For instance, the drone takes approximately 1, 599 ms to complete the online phase for a request size of 8 and a policy size of 10 (cf. Fig. 3). In the same settings, our solution requires approx. 2.23 Mbytes of communication overhead and 85 Mbytes of RAM space, being tolerable for an actual commercial drone. We highlight that in such configurations the consumed energy is approx. 0.003% of the overall battery capacity, not requiring to equip the drone with large batteries.

By comparing *Testbed 1* with *Testbed 2*, we also notice differences in the impact of the request size on the execution time of the online phase on different devices. By comparing Figs. 3(b) and (f) for the STP running in Docker, the time is lower for *Testbed 1* for requests of size 5 or less; for larger requests, lower values are observed in *Testbed 2*. Such an unexpected behavior is caused by the *OTextension* of the ABY framework, as more communication overhead is required between Docker and the host system for larger requests. This is confirmed by comparing Figs. 5(b) and (f), where we do not observe this behavior. Indeed, the request size for such experiments is the same, not affecting the execution time.

Finally, the results reported in Fig. 7 confirm that our approach requires much less time than the PHE-based solution in [26], being more suitable for DaaS scenarios. The reduction in the execution time comes at the cost of an increased communication overhead and RAM. However, these values remain reasonably low to be tolerable on modern commercial drones.

7 CONCLUSIONS AND FUTURE WORK

In this work, we proposed a privacy-preserving multi-party access control solution for emerging Third-Party UAV Services, a.k.a., Drone-as-a-Service. As a distinctive advantage, our solution allows multiple DOs to provide their policies in a private form. These policies are evaluated in a privacy-preserving way by relying on the SFE paradigm, thus protecting sensitive information therein. We deployed a proof-of-concept of our solution on two testbeds, emulating the drone through a Raspberry PI 3 Model B+ device. We showed that our solution achieves privacy-preserving multi-party policies evaluation for system configurations typical of real-world use cases in a reasonable time. We also evaluated communication overhead, RAM occupancy, and energy consumption, showing the suitability of the proposed approach for Drone-as-a-Service scenarios. Overall, our work makes a step further in demonstrating the viability of privacy-preserving techniques on commercial drones. In the future, we plan to test our solution on real drones.

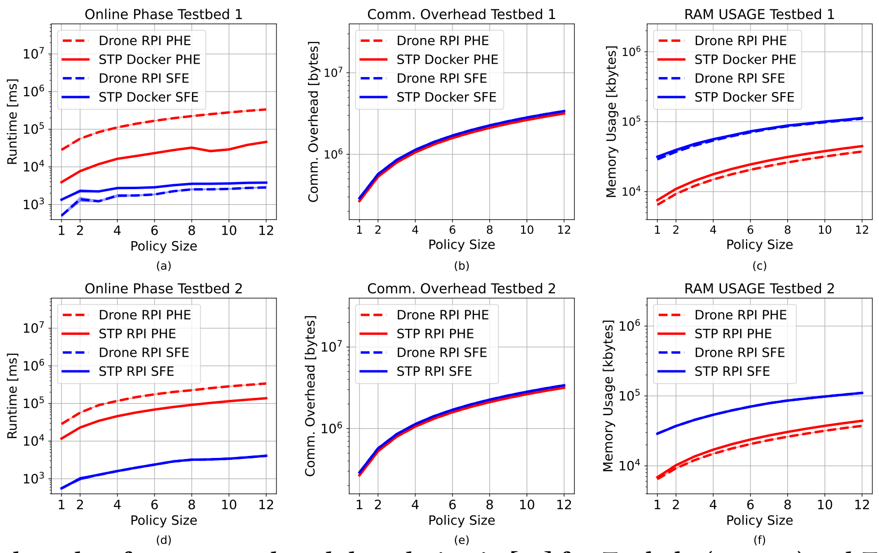

Figure 7: Experimental results of our approach and the solution in [25] for *Testbed 1* (top row) and *Testbed 2* (bottom row), varying the policy size.

ACKNOWLEDGMENTS

This work has been supported by the INTERSECT project, Grant No. NWA.1162.18.301, funded by Netherlands Organisation for Scientific Research (NWO). Any opinions, findings, conclusions, or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of NWO.

REFERENCES

[1] [n.d.]. *Become a Drone Pilot.* `https://www.faa.gov/uas/commercial_operators/become_a_drone_pilot/` (Accessed: 2023-03-01).

[2] [n.d.]. *Drone Services Market.* `https://www.fortunebusinessinsights.com/drone-services-market-102682` (Accessed: 2023-03-01).

[3] Antonio De Rubertis, et al. 2013. Performance evaluation of end-to-end security protocols in an Internet of Things. In *SoftCOM*. 1–6.

[4] Ludovic Apvrille, Tullio Tanzi, and Jean Dugelay. 2014. Autonomous drones for assisting rescue services within the context of natural disasters. In *GASS*. 1–4.

[5] Kais Belwafi, Ruba Alkadi, Sultan A. Alameri, Hussam Al Hamadi, and Abdulhadi Shoufan. 2022. Unmanned Aerial Vehicles' Remote Identification: A Tutorial and Survey. *IEEE Access* 10 (2022), 87577–87601.

[6] Corinne Bernstein. [n.d.]. *What is drone services (UAV services)?* `https://tinyurl.com/jtd8p5sr` (Accessed: 2023-03-01).

[7] Business Wire. 2021. *Global Drones as a Service Market - by Applications and Leading Industries.* `tinyurl.com/yw29enwt` (Accessed: 2023-03-01).

[8] Melissa Chase. 2007. Multi-authority Attribute Based Encryption. In *Theory of Cryptography*. Springer, Berlin, Heidelberg, 515–534.

[9] Melissa Chase and Sherman S.M. Chow. 2009. Improving Privacy and Security in Multi-Authority Attribute-Based Encryption. In *CCS '09*. 121–130.

[10] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation.

[11] DJI. 2021. DJI Mini 2 Specs. `https://www.dji.com/nl/mini-2/specs`. (Accessed: 2023-03-01).

[12] DroneDOJO. 2021. Raspberry Pi Drone | The Ultimate Project Drone. `https://dojofordrones.com/raspberry-pi-drone/`. (Accessed: 2023-03-01).

[13] Kai Fan, Huiyue Xu, Longxiang Gao, Hui Li, and Yintang Yang. 2019. Efficient and privacy preserving access control scheme for fog-enabled IoT. *Future Generation Computer Systems* 99 (2019), 134–142.

[14] Han Jin. 2021. Drones in construction 2022: Top Full Guide For You. `https://lucidcam.com/drones-in-construction/`

[15] Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. 2007. Attribute-Based Publishing with Hidden Credentials and Hidden Policies. In *NDSS*.

[16] Nikos Komninos and Aisha Junejo. 2015. Privacy Preserving Attribute Based Encryption for Multiple Cloud Collaborative Environment. In *UCC*. 595–600.

[17] Anis Koubâa, Adel Ammar, Mahmoud Alahdab, Anas Kanhouch, and Ahmad Taher Azar. 2020. Deepbrain: Experimental evaluation of cloud-based computation offloading and edge computing in the internet-of-drones for deep learning applications. *Sensors* 20, 18 (2020), 5240.

[18] Huimin Lu, Yujie Li, Shenglin Mu, Dong Wang, Hyoungseop Kim, and Seiichi Serikawa. 2017. Motor Anomaly Detection for Unmanned Aerial Vehicles Using Reinforcement Learning. *IEEE Internet of Things Journal* 5, 4 (2017), 2315–2322.

[19] Yang Ming and Tingting Zhang. 2018. Efficient Privacy-Preserving Access Control Scheme in Electronic Health Records System. *Sensors* 18 (10 2018), 3520.

[20] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. 2008. Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures. In *ACNS '08*. Springer, 111–129.

[21] Federica Paci, Anna Cinzia Squicciarini, and Nicola Zannone. 2018. Survey on Access Control for Community-Centered Collaborative Systems. *ACM Comput. Surv.* 51, 1 (2018), 6:1–6:38.

[22] Pere Molina, M. Eulàlia Paré, Ismael Colomina et al. 2012. Drones to the Rescue! Unmanned Aerial Search Missions Based on Thermal Imaging and Reliable Navigation. *InsideGNSS* 7 (2012), 36–47.

[23] David Perroud. 2021. Drones in construction and infrastructure. `https://tinyurl.com/mr4b6hmt` (Accessed: 2023-Mar-01).

[24] Raspberry Pi 3 Model B+. 2021. `https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/`. (Accessed: 2023-03-01).

[25] Mina Sheikhalishahi, Ischa Stork, and Nicola Zannone. 2021. Privacy-preserving policy evaluation in multi-party access control. *J. Comput. Secur.* 29 (2021), 613–650.

[26] Mina Sheikhalishahi, Gamze Tillem, Zekeriya Erkin, and Nicola Zannone. 2019. Privacy-Preserving Multi-Party Access Control. In *WPES*. ACM, 1–13.

[27] Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, and Nicola Zannone. 2018. Multi-party access control: requirements, state of the art and open challenges. In *SACMAT*. ACM, 49–49.

[28] Eva Wisse, Pietro Tedeschi, Savio Sciancalepore, and Roberto Di Pietro. 2023. A^{2}RID-Anonymous Direct Authentication and Remote Identification of Commercial Drones. *IEEE Internet of Things J.* (2023).

[29] Dan Woods. 2017. *10 Killer Use Cases: What Drones-as-a-Service Can Do For Your Business.* `https://tinyurl.com/5bhpm44x` (Accessed: 2023-03-01).

[30] Yang Xu, Quanrun Zeng, Guojun Wang, Cheng Zhang, Ju Ren, and Yaoxue Zhang. 2018. A Privacy-Preserving Attribute-Based Access Control Scheme. In *SpaCCS*.

[31] Kan Yang, Qi Han, Hui Li, Kan Zheng, Zhou Su, and Xuemin Shen. 2017. An Efficient and Fine-Grained Big Data Access Control Scheme With Privacy-Preserving Policy. *IEEE Internet of Things Journal* 4, 2 (2017), 563–571.

[32] Justin Yapp, Remzi Seker, and Radu Babiceanu. 2016. UAV as a service: Enabling on-demand access and on-the-fly re-tasking of multi-tenant UAVs using cloud services. In *DASC*. IEEE, 1–8.

[33] Yinghui Zhang, Xiaofeng Chen, Jin Li, Duncan S. Wong, and Hui Li. 2013. Anonymous Attribute-Based Encryption Supporting Efficient Decryption Test. In *ASIA CCS '13*. ACM, 511–516.