Context-Aware Privacy-Preserving Access Control for Mobile Computing

Juan Luis Herrera[a], Hsiao-Yuan Chen[b], Javier Berrocal[a], Juan M. Murillo[a], Christine Julien[b]

[a] *University of Extremadura, Spain*
[b] *University of Texas, Austin, United States*

Abstract

This work has been published: https://doi.org/10.1016/j.pmcj.2022.101725
In mobile and pervasive computing applications, opportunistic connections allow co-located devices to exchange data directly. Keeping data sharing *local* enables large-scale cooperative applications and empowers individual users to control what and how information is shared. Supporting such applications requires runtime frameworks that allow them to manage the who, what, when, and how of access to resources. Existing frameworks have limited expressiveness and do not allow data owners to modulate the granularity of information released.In addition, these frameworks focus exclusively on security and privacy concerns of data *providers* and do not consider the privacy of data *consumers*. We present PADEC, a context-sensitive, privacy-aware framework that allows users to define rich access control rules over their resources and to attach levels of granularity to each rule. We provide a formal definition of PADEC and an implementation based on private function evaluation. Our evaluation shows that PADEC is more expressive than other mechanisms, protecting privacy of both consumers and providers.

Keywords: privacy, device-to-device, mobile computing, access control

1. Introduction

During the last several years, there has been a massive increase in the deployment of mobile devices [1]. Similar trends exist for Internet of Things (IoT) and wearable devices [2]. These devices carry a large number of on-board sensors that provide a rich view of the surrounding context and support a wide array of applications. In addition, these devices commonly

accompany the user, providing an interface between their owner and a wider networked community [3].

This astounding increase in *companion devices* enable new systems and applications that leverage the myriad devices to execute large-scale sensing tasks. Mobile crowd sensing recognizes that a user's sensing needs can often be fulfilled by others nearby [4]. Opportunistic data sharing [5], which can be combined with mobile crowd sensing, relies on transient wireless connections [6] to distribute and fulfill a data sharing task using information from local networked devices. Both domains rely on communities formed based on the contact nature of human behavior [7]. As interactions are pushed into the opportunistic space, users need mechanisms to control the release of data according to their privacy needs. In decentralized and opportunistic scenarios, trust among peers and the security of the information shared are crucial [8]. Nevertheless, it is also of paramount importance to implement privacy management mechanisms so that users can manage who, when, and how their personal information is consumed [9].

As a running example to use throughout the paper, we describe an opportunistic data sharing application for the city of New York. This application leverages users' sensors to provide functionalities to both tourists and residents. Concretely, one functionality of the application is to store points of interest (POIs) identified by residents and then offer this information to tourists or other residents via a microservice. Obviously, information about an individual's visits to particular POIs may be sensitive, and residents should be able to decide with whom their POI information (e.g., date, purpose, people they visited with, review, etc.) can be shared and under which conditions. For instance, some residents may share this information only with other users that belong to similar social groups, or with tourists that have a specific context (e.g., are physically nearby).

Context-aware access controls constrain access to data based on contextual conditions [10]. In contrast, basic access control, such as Role-Based Access Control (RBAC) [11] or Dynamic Sharing and Privacy-aware Role-Based Access Control (DySP-RBAC) [12], allow providers to control access to information based on the identity of a potential consumer. These mechanisms are not sufficiently expressive to support opportunistic applications in which the identities of most peers are not known *a priori*. In addition, these existing techniques do not allow providers to define dynamic and contextual conditions. For instance, residents in our example would not be able to limit access to POI information to tourists who are nearby. Attribute-Based Access Control (ABAC) [13] and the access control mechanism from [14] do allow users to define rules based on contextual conditions beyond identity.

However, these mechanisms cannot modulate the precision of information released to each peer as a function of the shared context, which prevents providers from exerting fine-grained control over the access. For example, these techniques would allow tourists to access to the complete list of POIs if the stated contextual conditions are met, but they do not allow residents to release only a partial set of the POIs depending on the contextual situation of each tourist. Furthermore, these systems are designed for centralized (cloud-based) data sharing in which a coordinator mediates access. This is not the case in our scenarios, which are often completely decentralized and rely on the provider and consumer coordinating directly.

Finally, attribute-based or context aware techniques usually require consumers to share their whole context to evaluate if they can access the offered services. In a cloud-supported system, a trusted third-party has an omniscient view of this information. However, in opportunistic environments, the exchange of context to support access control is peer-to-peer. This could lead to an increase in the exposure of consumersIn our running example, tourists would have to release their social groups, noise level, location, etc. to get access to POI information of a provider, regardless of the granularity of the information they desire.

We present a context-aware, privacy-preserving access control mechanism designed for completely decentralized data sharing (PADEC, Privacy-Aware DEvice Communication). PADEC empowers users by increasing the expressiveness of access rules and protecting the privacy of both providers and consumers. PADEC allows data and resource owners to define rich access rules based on widely varying contextual information. Providers attach filters to rules to provide access at different levels of granularity based on a consumer's provided context. At the same time, PADEC protects consumers' privacy by minimizing the amount of contextual information they need to share in their access requests. This paper's novel contributions are:

- We provide a formal specification of PADEC to allow for exact replication, implementation, and verification of PADEC.
- We use this formalization to show how to employ PADEC.
- We provide details about PADEC's implementation using private function evaluation [15, 16] to increase privacy and security.
- We apply PADEC in a large case study.
- We evaluate PADEC using a threat model and assess how PADEC mitigates each threat while also reducing consumers' and providers' exposure.
- We evaluate the probability of type I and type II errors that exist in PADEC's access control evaluation.

- We compare PADEC with state-of-the-art approaches for access control.
- We perform an user survey to evaluate the usefulness of PADEC's expressiveness for users, and the effects of using such user-created content in the proposed case study.

Section 2 introduces related context-sensitive and privacy-aware access control mechanisms. Section 3 describes the threat model for PADEC. Each of the threats is dealt with in the PADEC model, as detailed in Section 4. Additional implementation details are offered in Section 5, and a case study and user survey are presented and evaluated in Section 6. Finally, Section 7 concludes the paper with some final remarks.

2. Related Work

During the last few years, different context-sensitive, decentralized or privacy-aware access control mechanisms have been developed.

Context-sensitive access control. A variety of recent efforts use personal context information to authenticate users. In [17], context information such as nearby familiar devices or locations is used to apply an access control policy and adapt the authentication method of mobile phones. In [18], users' movements, like gait, are leveraged to validate identities. While identifying users using context attributes has proven useful for unlocking and adapting the behavior of personal devices, it requires an *a priori* model of each user, which is not feasible for large-scale applications, especially those that need to grant access to unknown users depending on their context.

Decentralized access control. Other existing frameworks allow access to shared data under different contexts. For instance, Penumbra [19] proposes a decentralized system that empowers users and that can be used in opportunistic scenarios without a central mediator. However, Penumbra only considers identity and omits other more expressive context information. In [20], the authors use a distributed ledger, called Tangle [21], to store policies and access rights for resource-constrained IoT devices. This ensures distributed auditability. Although this mechanism is also privacy-aware, privacy is only preserved if access decisions are made by honest participants, which is difficult to ensure in opportunistic scenarios.

Privacy-aware access control. Opportunistic and pervasive computing require decentralized and context-sensitive access control in which individual users can modulate the specificity of information to release depending on a consumer's context. Existing mechanisms have been defined for collaborative environments. The NIST standard RBAC [11] authenticates users

based on identity by grouping them with roles. However, RBAC is not context-sensitive nor privacy-aware. DySP-RBAC [12] extends RBAC into a context-sensitive and privacy-aware mechanism. However, DySP-RBAC is implemented by applying a cloud-based architectural style, in which all users are known in advance and do not retain ownership of their data. Moreover, context is based on users' relationships and is therefore fully identity-based.

ABAC [13] is potentially applicable for context-sensitive access control. However, it is not privacy-aware, as it does not allow data providers to set different levels of detail for information shared with different consumers. The work in [14] proposes an alternative for pervasive computing. Making use of adaptive technologies, users can define access control policies by providing conditions over context. However, this mechanism also requires a centralized knowledge base with context information from all users, which takes data ownership away from users and is unsuitable for opportunistic scenarios. Furthermore, it does not allow users to control access at different levels of granularity.

From these comparisons, while privacy-aware and context-sensitive access control exists, none of these mechanisms are designed for opportunistic scenarios making them unsuitable for pervasive computing environments. Efforts in privacy-aware access control are not often coupled with context-sensitivity and *vice versa*, and those that address both do not consider the needs of opportunistic data sharing.

3. Threat Model

We next define a threat model that we use to present the remainder of our contribution. In our system model, a user, known as the *provider*, voluntarily shares data or resources with *consumers*, as long as both provider and consumer meet certain conditions set by the provider. These conditions may constrain the provider's own context, the consumer's context, or a combination of the two. The provider can also filter or obfuscate information it provides based on the context. We assume an attacker whose objective is to obtain unauthorized information by exploiting any weakness of the system. Attackers can have up to three roles: they can masquerade as legitimate consumer and try to obtain information from providers; they can masquerade as legitimate providers, trying to obtain information from consumers; or they can be third-party attackers, trying to obtain information from messages exchanged by providers and consumers.

We build our threat model for decentralized access control for opportunistic data sharing incrementally, first considering common general threats

General Threats		
UA	*Unauthorized Access*	attacker poses as a consumer to try to obtain some data that the provider desires to deny access to.
CC	*Circumventing context constraints*	attacker poses as a consumer and attempts to use identity to access data when that access would be denied based on other context.
Individuals' Privacy Threats		
CE	*Consumer over-exposure*	an attacker using a modified PADEC poses as a provider to obtain information from a consumer using context in a way that is incongruent with the consumer's intent.
PE	*Provider over-exposure*	an attacker posing as a consumer obtains information with a finer granularity or higher precision than a provider desires to grant access to
IA	*Insider attack*	an attacker posing as a consumer obtains finer-grained information than a provider allows by correlating and aggregating results of repeated allowed requests.
Threats from Third Party Attackers		
EA	*Eavesdrop attack*	a third-party attacker obtains private messages shared between providers and consumers.
RA	*Replay attack*	a third party attacker obtains a legitimate message and replays it to incorrectly obtain access.

Table 1: Threats identified and addressed in PADEC.

and then moving to threats specific to a context-aware and opportunistic approach. We assign each threat a label to use throughout the paper (Table 1). We start with two general threats: *Unauthorized access* (UA) and *Circumventing context constraints*. Because we aim to create a privacy-aware access control model, we also identify three threats to individuals' privacy: *Consumer over-exposure* (CE), *Provider over-exposure* (PE), and *Insider attack* (IA). Finally, we also identify two third-party threats: *Eavesdrop attack* (EA) and *Replay attack* (RA).

We assume that an anti-tampering mechanism such as remote attestation [22] exists, and thus, contextual information cannot be forged. All of the remaining threats are directly addressed in our approach, as described in the following section.

4. The PADEC Model

In this section, we present the model of our context-aware, privacy-sensitive access control mechanism, PADEC. Section 4.1 presents the general architecture of PADEC and its key concepts, while Section 4.2 presents the

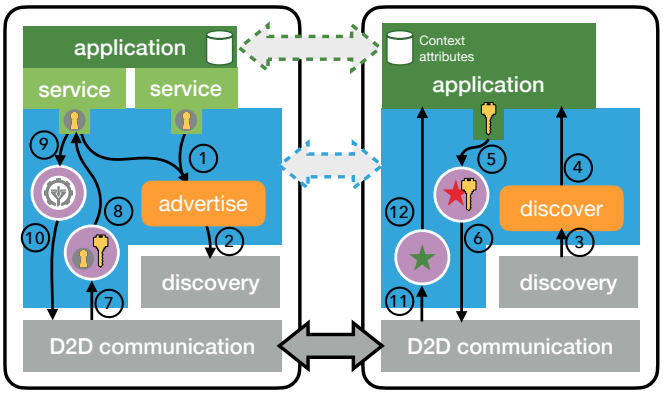

Figure 1: A PADEC interaction is initiated when a service provider (on the left) advertises a service (1). The advertised service is shared with nearby devices via an underlying discovery mechanism (2-3) and becomes available to nearby consumers (4). The provider's advertisement contains the keyholes for the service, which a consumer can use to construct a key (5) that is places in an application request (6) and sent to the provider. The provider checks the key against the service's keyholes (7) and invokes the service (8). A filtered or obfuscated response (10) is returned to the consumer device (11) and delivered to the application (12). As the horizontal arrows indicate, abstractly, the provider and consumer applications communicate with one another without considering the underlying mechanics. The semantics of the application and the context required for access control are shared at the application level. In contrast, PADEC (the blue box in the figure) holds the semantics of keys and keyholes and the nature of requests and filtering, while the D2D communication is concerned with networking details.

formal model. Finally, Section 4.3 presents how PADEC can be applied in a mobile computing case study.

4.1. Architecture and key concepts

From an architectural point of view, PADEC is designed to be a layer that can be placed *on top* of the endpoints exposed by an application to control their access. As such, one of the keys is the separation of concerns: the application endpoints should not concern about how access is controlled, and, in turn, PADEC should not concern about the specifics of the application such as the data semantics. Therefore, the concerns of PADEC are to offer users an expressive manner to decide which information should be shared with who, under which conditions to do so, as well as to enhance privacy. The architecture of PADEC is depicted in Figure 1, and the key concepts of PADEC are as follows:

1. **Lock**: locks are the elements of PADEC that protect endpoints. Each lock protects a single endpoint, and each endpoint can only have one

lock. A lock is a collection of *access levels*, sorted in non-decreasing order of information degradation.

2. **Access level**: access levels encapsulate a given granularity for the information provided by an endpoint, as well as the circumstances under which it can be accessed. Each access level contains a *filter* and a *rule*.
3. **Filter**: a filter is an abstraction of a mechanism to degrade or alter the granularity or the level of detail of the information returned from an endpoint. Built-in filters exist in PADEC, and they can be tailored by users by setting parameters. New filters can be added by injecting small pieces of code. Each filter comprises the filtering technique, the user-tailored parameters and the expected degradation of the information. Filters effectively address *PE* threats, and filtering techniques must be idempotent in order to avoid *IA* threats.
4. **Rule**: a rule in PADEC represents the conditions under which access is granted to a given access level. Rules can be defined over one or more of the contextual *attributes* of the provider, the consumer, or combinations of attributes (e.g., relative distance from the consumer to the provider), and define values for each attribute where access is granted. Rules tackle *UA* threats by controlling access to information, and allow attributes other than identity to be used to control access, effectively thwarting *CC* threats.
5. **Attribute**: an attribute is a type of contextual information, such as location, interests or social groups. The context of a user at a given point in time comprises the values of all of their attributes at that point in time.
6. **Keyhole**: a keyhole is the set of contextual attributes that a rule requires from the consumer in order to be evaluated. Keyholes are the elements through which consumers perceive the rules associated to access levels. Producers advertise locks as collections of keyholes, each associated with the expected information degradation. Keyholes enhance the efficiency of PADEC and decrease *CE* threats by minimizing the amount of exposed information.
7. **Key**: a key is the set of values for the attributes defined in one or more keyholes. When consumers discover a lock, they select the access levels they want to try accessing, and build a key that fits their keyholes. On the provider side, keys are used to evaluate the rules of the lock's access levels. Keys are never transmitted to the application, and stay solely within PADEC to ensure their privacy.

There is an additional feature that can be optionally enabled in PADEC,

but has no effects over the architecture: private function evaluation. Private function evaluation (PFE) [15] allows PADEC's key evaluation to be performed in a completely private manner: consumers provide an encrypted version of their key that can only be used a single time, while providers can evaluate whether their rule holds without decrypting the key or revealing their rule logic. The use of PFE enables PADEC to completely mitigate the *CE* threat. Keys do not change conceptually, although their communication is performed in a different manner due to PFE's characteristic of single-use inputs. The details of these communications are detailed in Section 5.2.

Finally, with regards to the device-to-device communication, PADEC ensumers that all device-to-device communications are encrypted, so that only the provider and consumer are able to interact, avoiding both third-party attacks in Table 1. To encrypt communications, before starting a PADEC interaction, the provider and consumer follow an ephemeral Diffie-Hellman key-agreement protocol [23], in a manner similar to SSL/TLS. The consumer generates an ad-hoc public key and shares it with the provider, including some of the parameters used to generate the public key. This key is *public* because it is shared with the other party. A new key is generated for each PADEC exchange, used exclusively for key agreement within the communication, and does not serve an identification purpose [23]. The provider, using the settings received from the consumer, generates its own public key, which is shared with the consumer. Finally, based on the shared public keys, both consumer and provider generate a secret. This secret is guaranteed to be the same for both parties and is used as a symmetric encryption key to protect the rest of the communications. Since messages are encrypted, eavesdropping is not a concern, since the attacker cannot examine message content. *RA* threats are also thwarted, since any answer the attacker might get by replaying messages will be as unreadable as any message they overhear.

Each data exchange is supported by a continuously executing endpoint discovery process [24]. The workflow that is used to communicate back and forth in a PADEC interaction is depicted by Figure 2. The ephemeral Diffie-Hellman protocol can be piggybacked onto this discovery process: the consumer can send its public key with its discovery query (step 1 in Figure 2), and, as part of returning the endpoint, the provider can share their public key. The agreed-upon symmetric key can be used to encrypt subsequent communications.

4.2. The PADEC Formal Model

The formal model of PADEC provides a formal description of how access is controlled and concrete details on how each of the architectural elements

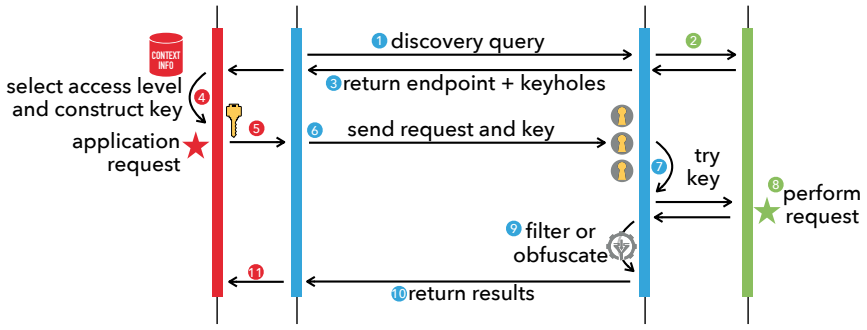

Figure 2: PADEC Sequence Diagram: A consumer (on the left side) discovers an endpoint and its access levels in the surroundings (1-3). The application selects an access level via a *keyhole* (4) and makes an application request (5). The request is sent to the provider (on the right side).The provider checks the consumer's key against the set of available keyholes (7) and, if access is granted, the provider performs the request (8) and returns the results (10-11), after applying a filter (9).

intertwine, making it easier to re-implement or replicate.

Users and endpoints. In this formal model, the base components are users ($\mathcal{U}$), who participate in a PADEC interaction. Users can be humans, devices, or applications and each user can act as a provider, a consumer, or both. A provider exposes a set of endpoints ($\mathcal{E}$) belonging to application programming interfaces (APIs) that consumers can discover and access to leverage some data or service. Formally, an endpoint $e \in \mathcal{E}$ is a function that maps an input to an output: $e : I \to O$, and a given provider's endpoints can be referenced as $\mathcal{E}^u = \{e\}$ for provider $u \in \mathcal{U}$. An endpoint is identified by its *interface*, and therefore the same endpoint may be provided by multiple providers.

Attributes. In PADEC, attributes ($\mathcal{A}$) describe the contextual state of users. Each user is associated with a set of attributes that describes their situation; a particular user's attributes are defined by:

$$\mathit{attributes} : \mathcal{U} \to \mathcal{P}(\mathcal{A}) \tag{1}$$

where $\mathcal{P}(\mathcal{A})$ is the power set of $\mathcal{A}$. For a given user u, $\mathit{attributes}(u) \subseteq \mathcal{A}$.

Each attribute has a primitive type, which constrains what a legal value is (e.g., locations are pairs of longitude and latitude). A particular application may provide additional constraints on the value. For instance, locations of points of interest may be constrained to be within a particular city's boundaries, and ratings, while integers, must be between 0 and 5. For the formal model, we refer generically to the universe of possible values as

$\mathcal{V}$, with the understanding that, for a given attribute a, the set of possible values is a subset of $\mathcal{V}$.

Context state. The *context* is defined as a function that maps each user $u \in \mathcal{U}$ to a set of values for each attribute:

$$context : \mathcal{U} \nrightarrow \mathcal{A} \times \mathcal{V} \tag{2}$$

The *context* function is a *partial function*; the attributes that are relevant for a particular user may be only a subset of the full attribute set $\mathcal{A}$. It can be more convenient to reference the context state of a user $u \in \mathcal{U}$ as a mapping of the user's contextual attributes to concrete values:

$$context(u) : attributes(u) \rightarrow \mathcal{P}(\mathcal{V}) \tag{3}$$

Rules. PADEC defines *rules* ($r \in \mathcal{R}$) that allow providers to restrict access to resources based on a consumer's context. A rule comprises one or more *clauses* that constrain the values of context attributes. PADEC defines set operators for discrete attributes and comparison operators for attributes whose values are ordered or partially ordered. In total, PADEC provides six operators($\mathcal{OP}$): $\in$, $=$, $\neq$, $>$, $<$, and $<>$ (which tests whether a value is within a specified range). To combine clauses over multiple attributes, PADEC supports combinations using $\wedge$, $\vee$, and $\neg$.

To fully specify rules and clauses, we use a context-free grammar [25], $G = (N, \Sigma, P, S)$. The sentence symbol of G is a rule, i.e., $S = r$. Because a PADEC rule is composed of clauses that combine attributes, operators, and values; the clause is the only nonterminal symbol, i.e., $N = \{c\}$. The terminal symbols are $\Sigma = \{a, op, v, \vee, \wedge, \neg\}$. These symbols connect directly to the PADEC formal model: $a \in \mathcal{A}$, $op \in \mathcal{OP}$, $v \in \mathcal{V}$ and $r \in \mathcal{R}$. The grammar captures the creation of rules through clauses, attributes, values, and operators via its production rules:

$$\begin{array}{lll} r \rightarrow r \vee c, & r \rightarrow r \wedge c, & r \rightarrow \neg r, \\ r \rightarrow c, & c \rightarrow a\ op\ v, & c \rightarrow v\ op\ a \end{array}$$

Although the final two production rules are similar, they allow for asymmetric operators (e.g., $\in$). Moreover, the first two production rules allow one to combine multiple clauses, thus also allowing for combined rules.

At a higher level, a PADEC rule can be viewed as a function, defined by its clauses, that maps a context state or a subset of a context state to a Boolean value:

$$rule : \mathcal{R} \times \mathcal{C} \rightarrow \{0, 1\} \tag{4}$$

Generically, we refer to the universe of rules as $\mathcal{R}$.

Keyhole. Consumers need to know what context information they need to provide to access an endpoint. The *keyhole* captures the set of contextual attributes required by the rule, limiting the state the consumer needs to provide. Given a rule (r) that combines clauses, each of which references an attribute of $\mathcal{A}$, the *keyhole* for rule r is simply the union of the set of attributes referenced by all of the clauses:

$$keyhole(r) = \{a : a \in r\} \tag{5}$$

Given that $\mathcal{R}$ is the set of rules in a PADEC system, we define $\mathcal{H}$ to be the complete set of possible keyholes:

$$\mathcal{H} = \{keyhole(r) \forall r \in \mathcal{R}\} \tag{6}$$

Key. A key is a set of pairs of attributes and values, $\mathcal{K} \subseteq \mathcal{A} \times \mathcal{V}$, obtained from the context state of the consumer and used to evaluate a rule. Given the set of required attributes for a keyhole, h, a consumer can define a *key* that contains only attributes listed in h. This key is a *restriction* of the user's complete context state function:

$$keyFromKeyhole(u, h) = context(u) \mid_h \tag{7}$$

It is important to note that each key is a subset of the complete context state of the user. Thus, with a slight abuse of notation, the *rule* function can operate over a key, given that it is a subset of a complete context state.

Filter. A PADEC filter is any technique used to degrade the information returned by an endpoint. Formally, a filter is a function whose input is the result returned by an endpoint (i.e., an output, O) and whose output is a degraded (or *filtered*) version of the endpoint's output. Filters must be idempotent: for a given input to the filter, its output should always be the same. Moreover, the information degradation can be expressed as a real number:

$$f : O \rightarrow O \times \mathbb{R} \tag{8}$$

Each filter comprises a method used to degrade the information, as well as a set of parameters tailored by the provider, that specify how much information is degraded or which information should be obfuscated. Furthermore, each filter also reports the expected degradation of the filtered output:

$$degradation(f) = \mathbb{R}, f \in \mathcal{F} \tag{9}$$

where $\mathcal{F}$ is the set of all filters. This *degradation* value provides a means for the provider and consumer to communicate about the quality of the result in a datatype- and application-specific way, as the degradation metric is tied to the datatype the filter is able to be used on (e.g., the euclidean distance may be a degradation metric for a filter that works on coordinates, but not for one that works on strings).

Access level. Formally, an access level combines a rule and a filter:

$$\mathcal{AL} \subseteq \mathcal{R} \times \mathcal{F} \tag{10}$$

Each access level also has a public interface. That is, for $al \in \mathcal{AL}$, if r_{al} is the access level's rule and f_{al} is the access level's filter, then the public interface for al is:

$$al_{pub} = (keyhole(r_{al}), degradation(f_{al})) \tag{11}$$

Through this public interface, consumers can select a desired access level based on the combination of the context information they are required to provide (i.e., the keyhole) *and* the quality of information they can achieve in return (i.e., the level of information degradation).

Lock. A lock in PADEC is the set of all access levels that a provider defines over an endpoint. Conceptually, locks are the access control entities in PADEC: endpoints are directly protected by user-defined locks, which can contain as many access levels as the provider desires. Formally, the existence of locks is represented as the definition of PADEC's access control function, relating each endpoint to a lock (i.e., multiple access levels):

$$\mathcal{AC} : \mathcal{E} \times \mathcal{U} \nrightarrow \mathcal{AL} \tag{12}$$

Finally, we consider that consumers can construct keys for multiple endpoints. To do so, we define $\mathcal{AL}_{e_u t}$ as the subset of $\mathcal{AC}(e_u)$ a given consumer t is interested in. The consumer can use multiple criteria, based on both degradation and keyhole information, to select some access levels: those with low enough degradation to enable their use case, those that do not contain attributes they are unwilling to share, or even all of them. Based on this, they can build a single key designed to fit all the keyholes in the public interfaces of the access levels in $\mathcal{AL}_{e_u t}$ they are interested in at once:

$$key(t, \mathcal{AL}_{e_u t}) = \bigcup_{al \in \mathcal{AL}_{e_u t}} keyFromKeyhole(t, keyhole(r_{al})) \tag{13}$$

To evaluate a key, the provider must test the key in their access levels in their non-decreasing order by their filter's degradation. Formally, to test a

key in a given access level, the provider evaluates the *rule* function associated with the access level. If the function evaluates to 1, access is granted, the provider evaluates the filter function f associated to the access level, and returns the filtered output to the consumer, which finishes the evaluation. However, if the rule evaluates to 0, access at that level is denied, and the next access level should be tested. If no further access levels remain to be tested, access is effectively denied.

4.3. Case study: Mobile Crowd Sensing

To better understand the concepts of PADEC, we present a case study based on a mobile crowd sensing (MCS) application for a smart city. The MCS application leverages users' sensors and computing power to, first, promote social running among locals, and, second, help tourists identify Points of Interest (POIs) in the city. Such a scenario is not just a thought experiment; similar applications have already been envisioned [26]. Concretely, residents can share some of their contextual information, such as the routes they use for running in the city, the time they take to run through it or their presence in one of them, with fellow residents. This information is exposed from the users' smartphone by three endpoints: *getRoutes*, *getBestTimes*, and *getPresence*. On the other hand, to promote tourism, it exposes from residents the POIs they frequent, along with their rating with two endpoints: (*nearByPOIs*), and (*getRatings*) respectively. Moreover, PADEC also manages other kinds of contextual information for the consumers of these endpoints, such as their social groups, location or interests.

When users share personal or sensitive information such as their presence in a given route , privacy is crucial due to the personal nature of the data being handled. A malicious party could potentially use the history of a user to track their habits, learn their routine, etc., so this information must be carefully protected. The privacy exposure of the above detailed endpoints can be modulated with PADEC by each user. In this sense, residents can be willing to share their presence with their friends or family, but not with unknown users. Therefore, as Figure 3 shows, a resident can define two access levels: one providing the exact location in the route and another one where it is only shared during the daytime (i.e., it has a degradation of 0.5). These two access levels have a lock associated detailing two rules: the first one indicates that the exact location can only be consumed by family members, and the second one defines that the degraded one can be consumed by runners in nearby routes. Likewise, users can set up a lock for each of their endpoints, not detailed in the the figure to improve its legibility.

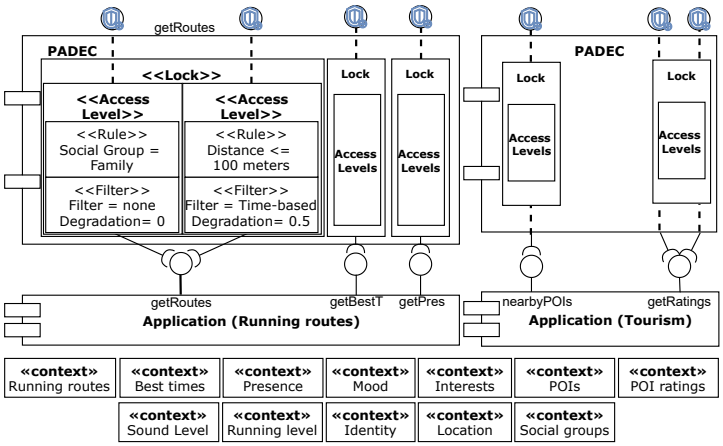

Figure 3: Diagram detailing how PADEC is used in the case study.

5. Implementing PADEC

Based on the PADEC model from Section 4, we next present PADEC's implementation details. So far, we distinguished between two PADEC implementations: PADEC, which directly implements the formal model with no additional details, and *PFE PADEC*, which makes use of Private Function Evaluation (PFE) techniques to evaluate the rules.[1]

5.1. PADEC implementation

The PADEC implementation uses Java to implement the detailed formal model. Java is used for two main reasons: first, Java is a highly portable language that can be used as a native language in billions of devices. Thus, Java allows PADEC to run in companion devices natively. Second, Java's reflection capabilities, and its concept of interfaces and its ease of integration with other languages fit PADEC's separation of concerns requirements.

Starting from the building blocks, PADEC defines endpoints through the *Endpoint* interface: a generic interface with a single method that takes generic parameters and returns a specific data type. While PADEC will call this method, there are no limitations to its implementation (e.g., as a wrapper to an already existing application endpoint). Attributes are implemented as holders for a certain data type that inherit from the *Attribute* class. Therefore, each attribute is free to have its own validity checks, data type, and implementation. As stated above, attributes can be combined

[1]This implementation is available at `https://bitbucket.org/spilab/padec_theone`

using operators (which are elements of $\mathcal{OP}$ and the logical connectors). All of the operators are implemented using interfaces. These operators generate booleans from two given objects (i.e., comparing attributes to values), or from other booleans (i.e., joining together the results of different clauses).

Rules with single clause are implemented by storing the attribute, values and operator. Rules with multiple clauses are created by storing two existing rules and the connecting operator that joins them. Both single and multi-clause rules inherit from the *Rule* class, this implementation allows us to joining any number of clauses, allowing more expressive and concrete rules. In addition, Rule provides a utility method that returns all the attributes required by the rule. This method is specially important for getting the *keyhole* for the rule. This *keyhole* with the names of the attributes is used by PADEC for creating the *key* that may open the keyhole.

Finally, an *AccessLevel* instance combines an endpoint, the rule that must hold in order to grant access, and a filter. Filters can define the data type or types they can operate with, as well as their degradation as a real numbers.Each filter can be parameterized through a map of strings to objects, which affect the degradation value it returns.

PADEC has been implemented to be easily integrated with other tools, such as wizards that allow for the creation of locks in a user-friendly manner.

5.2. PFE PADEC

PFE allows entities to share information that are unable to understand, but still able to perform operations over it. PFE PADEC uses this technique for sharing and validating the keys, instead of encrypting them with a key. The PFE PADEC implementation is designed so three constraints hold: the consumer cannot know the logic behind the provider's access control rules, the provider cannot know the value of the consumer's attributes, and the validation results must be identical to those of PADEC.

To perform PFE, this implementation makes use of ABY [16], which, despite being intended as a secure two-party computation framework, serves the purpose of a PFE framework. ABY was chosen because of its relatively low overhead, and its availability as open source software.ABY is fully implemented in C++, and was integrated in PFE PADEC using the Java Native Interface (JNI). ABY's functions are built as *circuits*. By using ABY, consumer and provider are able to compute a common circuit, each with private inputs to it, though the circuit is public to both parties. To maintain the logic secret, PFE PADEC uses circuits that perform every available operation over the data, and have their output connected to a multiplexer. The multiplexer is the output of the circuit, and its output is chosen by a

provider's private input. Thus, the consumer cannot know which operations will be outputted, and hence, cannot know the logic behind the rule.

The main difference between PFE and non-PFE PADEC are keys. In non-PFE PADEC, keys contain the values of the required consumer's attributes, exactly mimicking the formal model. While PFE PADEC still maintains the key concept and design from the formal model, the transmission and evaluation of keys is performed differently. In non-PFE PADEC, the key is transmitted by the consumer as a single message, containing the name of each attribute and the value for said attribute. A key is received by the provider, who knows all of its content and can reuse it on multiple access levels. In PFE PADEC, however, the key is transmitted as a series of messages. First, the consumer sends a synchronization message telling the provider which access levels they want to test, in which order, with which attributes, and in which order each of the attributes will be provided. After this message is sent, a set of JNI calls are performed. Consumer and provider create the circuit on their side, provide their private inputs (in the case of the consumer, they are sent encrypted to the provider), and execute the circuit. The resulting output is also constrained so only the provider can obtain its value.

6. Evaluation

In this section, we present an evaluation of PADEC over the case study presented in Section 4.3, including both a simulation of the case study and the results of a conducted survey. The evaluation's objective is twofold: on the one hand, it is aimed at comparing PADEC with other access control mechanisms: RBAC, ABAC and DySP-RBAC, along with comparing PADEC and PFE PADEC. Furthermore, it is also aimed at evaluating the usefulness of PADEC's expressiveness for users, as well as effects of leveraging user-made locks over some of the metrics.

The results of the evaluation are divided into two kinds of results: experimental evaluation (presented in Section 6.2), which comprises results obtained from the simulations using a baseline to compare multiple access control mechanisms, and survey evaluation (presented in Section 6.3), which includes results obtained from the conducted survey and the simulations that use it as a baseline. Before presenting the results, Section 6.1 contains the setup of simulator and the case study used to evaluate PADEC.

6.1. Scenario setup

PADEC allows device-to-device interactions within opportunistic networks. Therefore, PADEC is evaluated within a simulated opportunistic network in order to validate its applicability to opportunistic scenarios. Keep in mind that PADEC is a security and privacy framework at the application level of the communication, and is evaluated as such. Thus, the objective of the evaluation is not to evaluate or test the performance of the opportunistic network, or the opportunistic routing algorithms. The simulated opportunistic scenario has been implemented using the tools from TheONE simulator [27]. Within the simulator, the map of New York City is implemented, as obtained from OpenStreetMap [28], for users (i.e., nodes) to roam. Users are separated into two kinds: tourists and residents. Both kinds of users roam around the NYC map using shortest path map-based movement: they select a point in the map at random, find the shortest path to said point through the streets of NYC and move there. Once the point is reached, another point is selected randomly. Tourists roam around points from the touristic zones of the city, whereas residents roam around both touristic and residential zones. For each of the scenarios, 100 tourists and 50 residents are simulated. Furthermore, to ensure comparability among the results, the same RNG seeds are used in all the evaluation scenarios. Each of the considered nodes communicates with others using TheONE's SimpleBroadcastInterface, able to transmit data in a 50 meters range, and routes information using the default EpidemicRouter included in TheONE. The simulation lasts for 86000 seconds, which is approximately 24 hours, and each consumer attempts to make a new PADEC service discovery request every 50 seconds, unless the consumer is waiting for the response of a previous request. This opportunistic network environment is used to perform the comparative analyses included in the evaluation, ensuring a fair scenario for all four access control mechanisms. The contact times of the nodes last for an average of 73 seconds, with a standard deviation of 116.32 seconds. Moreover, 90% of the contacts last less than 100 seconds. Focusing on this 90% of the contacts, they follow a normal distribution, with a mean contact time of 44.4 seconds and a standard deviation of 12.8 seconds. Intercontact times do not follow a known distribution, and last on average 12029.56 seconds with a standard deviation of 12519.39 seconds.

At the application level, the case study described in Section 4.3 has been implemented in the TheONE simulator. The evaluation is performed by measuring statistics of the *nearByPOIs* endpoint, which yield POIs obtained from the New York City POI dataset from [29]. Each of the simulated users is linked with an anonymized user ID in the dataset, and only releases POIs

belonging to said user ID. Despite the fact that the case study presents a higher number of endpoints, we decide to measure *nearByPOIs* only. This decision is motivated by a key tenet of PADEC: separation of concerns. Endpoints belong to the application logic side, and hence, PADEC is not dependant on the endpoints or vice-versa.

In these simulations, each type of context has a different category of sensitivity from the consumer's perspective. Higher categories are considered to be more sensitive. The context types we use are: (1) *identity*, which each consumer perceives as the most sensitive attribute, which we refer to as category 3; (2) *location* (category 2); and (3) *sound level* (category 1). We define consumer privacy as the average sum of the categories of the attributes shared, so that 0% means all attributes are shared and 100% means no attributes are shared. Higher values for this metric therefore indicate a higher degree of consumer privacy.

6.2. Experimental evaluation

The first objective of this evaluation is to evaluate the trade-off between the privacy PADEC offers to providers and the data quality being released, so that providers are not overly exposed when they release data, while consumers are able to get enough data quality to work with the released information. Moreover, this trade-off is also evaluated on the consumer side by comparing consumer privacy with the amount of consumers that could get their requests fulfilled within a time frame, minimizing consumers' exposure surface while making sure they are still able to get the information they need. The next objectives of the evaluation are to compare how well users are able to *hide in the crowd* using PADEC w.r.t. other mechanisms. We also intend to evaluate the number of false positives and negatives in PADEC generated due to the computation and communication overhead compared to alternative mechanisms.

Our first result is related to the number of successfully executed PADEC protocols. Due to the nature of the opportunistic network, any given two nodes may not be permanently connected to each other, as the range of their wireless interfaces is limited, and a path from the sender to the receiver may not exist. Therefore, two nodes may start a PADEC interaction, but disconnect from each other before said interaction has finished. In those cases, the PADEC protocol would not finish successfully, and some of the messages would never be sent or received. As a result, in the simulations, a total of 216318 PADEC service discovery requests are initiated, and 216268 of those requests were successfully executed. Thus, PADEC presents a 99.976% of success rate within the simulation, and consequently, we consider the packet

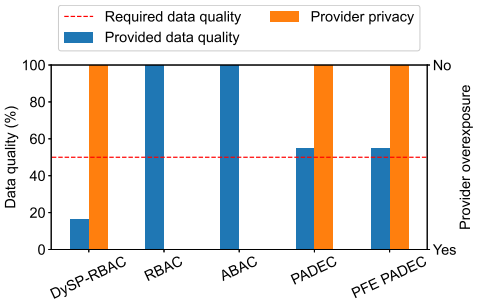

Figure 4: Data quality provided and provider privacy in PADEC

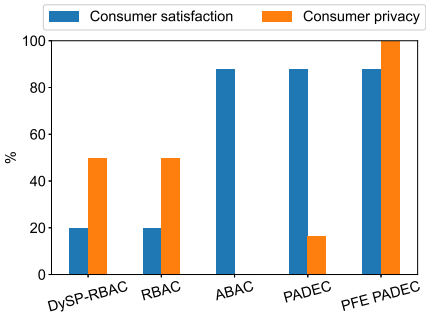

Figure 5: Consumer satisfaction and consumer privacy in PADEC.

loss rate from the application's perspective low enough to provide meaningful results in application evaluation.

Figure 4 compares the data quality offered by providers with the privacy PADEC offers to them in terms of overexposure. To be able to better understand the implications of data quality, Figure 4 depicts the data quality required by consumers to retrieve a good enough list of POIs (red dashed line), which is approximately 50%. In RBAC and ABAC, filters are not implemented, which implies that data is never degraded and, thus, the data quality is always 100%. On the other hand, this means that providers always share 100% of their data, hence being overexposed with no privacy. The implementation of filters in PADEC, as well as DySP-RBAC's restrictions on granularity, allow providers to customize the data quality they reveal, thus eliminating overexposure. This comes at the cost of reducing the average data quality of consumers. In the case of DySP-RBAC, this quality is under the required threshold, which implies that the average consumer obtains a very low-quality list of POIs. This problem is addressed by PADEC, thanks to the fact that consumers are automatically granted access at the highest access level they are allowed to. The raise in data quality does not mean that the data quality improves per-se in PADEC w.r.t. DySP-RBAC, it is a sign that more users are allowed access to higher access levels with higher data quality. Since the PFE integration does not affect data quality or provider privacy, its results are the same as PADEC.

Figure 5 shows the results comparing consumer privacy with consumer satisfaction. Consumer satisfaction is measured as the amount of consumers that got at least a satisfied request within a timeframe, in this case, 3 hours. It is important to note that the timeframe for satisfaction is completely application-dependent. Furthermore, if a consumer never meets the conditions set on the rules, they will never have their requests satisfied. Thus, our

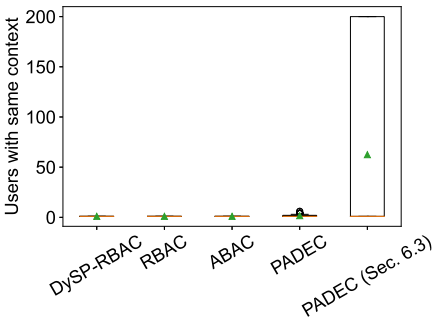

Figure 6: Consumer indistinguishability in PADEC.

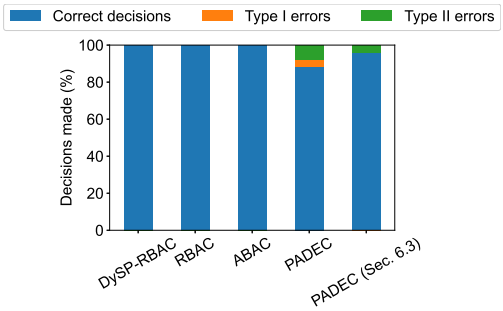

Figure 7: Type I and type II errors in PADEC.

baseline is not to have 100% consumer satisfaction, but rather to maximize consumer satisfaction along with consumer privacy, balancing the trade-off between them. With DySP-RBAC and RBAC, while consumers retain a high privacy due to only sharing their identity, a very low number of consumers, namely, those that are explicitly allowed access, can be satisfied. Satisfaction highly rises with ABAC due to a complete context check, with the cost of consumers having their context completely exposed, leading to no consumer privacy. With PADEC, consumers reveal more information than with DySP-RBAC or RBAC, decreasing their privacy, although releasing less information than ABAC. On the other hand, their satisfaction increases to ABAC levels, as their access attempts are always automatically adjusted to the according access levels. Finally, the introduction of PFE maintains satisfaction while guaranteeing 100% consumer privacy, as consumers do not have to reveal any of their contextual attributes any longer.

Along with the previous analysis, the consumer's privacy can also be seen as a metric of how easy they are to tell apart from other users. This definition of privacy, inspired by the concept behind differential privacy [30], is reflected as the *indistinguishability* of a key: given a certain key, its indistinguishability is the number of users who would be able to provide the same key. For instance, if a consumer provided exclusively their sound level, all the users with a similar sound level would also be able to provide an indistinguishable key. Thus, if there are multiple users with a similar sound level, the key would be highly indistinguishable. It is not possible to use techniques for differential privacy within PADEC, as differential privacy requires for all individuals to be contained in a dataset, while PADEC reveals a single individual's info (i.e., a single data point) at a time. However, an analysis of indistinguishability can be performed in a simulation in which it is possible to obtain the context of all the users. The results of the indistin-

guishability analysis are provided in Figure 6. In it, we can see DySP-RBAC, RBAC and ABAC have completely distinguishable keys, mainly due to the fact that they carry identity information. Only the consumer with a given identity is able to provide it in the key, and thus, the indistinguishability of keys with identity is very low. Nonetheless, PADEC provides more indistinguishable keys. In average, PADEC keys have an indistinguishability of 1.65 (1.10%), with an interquartile range of 1 (0.67%) to 2 (1.33%). This means that, normally, in our simulation, there is at least another user that is indistinguishable from the consumer. Moreover, we find some cases in which 3 (2%), 4 (2.67%) and even 6 (4%) users have been indistinguishable based on their key information. Thus, we conclude PADEC allows consumers to *hide in the crowd* better than DySP-RBAC, RBAC or ABAC.

An interesting analysis of the implications of the overhead is shown in Figure 7, which depicts the type I (false positive) and type II (false negative) errors for the each access control mechanism. It is important to note that these errors are not due to mistakes in rule evaluation, they are exclusively due to contextual changes over time. In some cases, by the time the rule is evaluated, the consumer's context has changed and, with it, the evaluation of the rule may change. For instance, it is possible that a consumer sends a key with a location that is close enough to the provider to get access, but, by the time the key is received and evaluated, the consumer's location has changed, and may not be close enough to get access. These are the kinds of errors reported in this analysis. We can see DySP-RBAC, RBAC and ABAC have no errors of this kind. Nonetheless, this phenomenon appears on PADEC in 12% of the interactions. In a 9%, the errors are of the false negative kind, i.e., if the consumer had sent their key by the time it was evaluated, access would have been granted rather than denied. In only 3% of the cases, an initially granted access could have been denied if the key was sent afterwards. These errors appear in PADEC due to its temporal and size overhead, as the PADEC handshake takes extra time to be sent and processed.

6.3. User survey

To evaluate PADEC from the user perspective, we have conducted a survey using the Qualtrics service, in which users were asked about their interest in the MCS case study applications, as well as to select and define their own, custom PADEC locks. The source of the survey is publicly avail-

able[2]. This survey is aimed at four main objectives: first, to determine if users are interested in context-aware applications. Second, to find out, for one of these applications, which kinds of information users would be willing to share with different groups of people (family, friends, or strangers) based in different contextual conditions. Third, out of different pre-made rules defined by users, how many of them could be expressed using different access control mechanisms. Finally, we allow users to define their own rules to protect their information, and evaluate which access control mechanisms would be required to implement said rules. Concretely, we define five types of contextual information: the POIs visited by the user, the ratings for said POIs, the running routes frequented by the user, their best times when they run through the routes, and their presence in a given running route.

On the other hand, we have also performed additional analyses based on the survey: as the surveyees were able to select some pre-made locks or define their own custom locks, they have been implemented in PADEC and used in an additional simulation.Furthermore, both residents and tourists in the survey-based simulations use the state-of-the-art SWIM movement model [31]. In these simulations, contacts last for 2.59 seconds on average, while the standard deviation is of 48.96 seconds. In the case of SWIM, 99.5% of the contacts are below 10 seconds, with an average of 1.44 seconds and a standard deviation 0.84 seconds. Intercontact times do not follow a known distribution, and last an average of 14533.24 seconds with a standard deviation of 14122.40 seconds. In these survey-based simulations, different users apply different rules over each concrete endpoint. Thus, to ensure the fidelity of the results in this scenario, the survey results are measured among all five endpoints, rather than a single one. It is important to note that the survey-based simulations should not be compared to the other access control mechanisms, as they use different baselines for their configuration.

Therefore, there are six evaluation objectives for the survey: using the results from the survey-based simulations, we analyze the indistinguishability of PADEC users, as well as the type I and II errors. On the other hand, using the survey results themselves, we analyze the interest of users in the proposed applications, the groups of people they would share the information with, and the access control mechanisms that need to be used to express both user-selected and user-made locks.

First, we analyze the indistinguishability of PADEC using the locks from the survey, as depicted in the rightmost box plot in Figure 6. In this case,

[2] `http://tiny.cc/padec_survey_source`

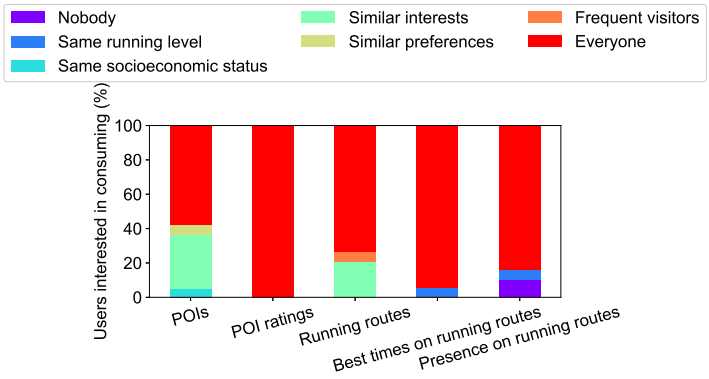

Figure 8: Interest of users on consuming the data of context-aware applications.

Figure 9: Groups that users would share different types of personal information with.

the rules are very diverse, as the interquartile range from 1 to 200 shows. On average, the keys provided have an indistinguishability of 62.42, which exhibits how PADEC consumers can *hide in the crowd* in such a situation. Continuing with the errors, as shown in Figure 7. Using the baseline for other simulations, in 12% of the interactions, the decision taken by PADEC would have changed due to the overhead. If the user-provided locks are used, however, PADEC exhibits no false positives and a 4% of false negatives, and thus, it can be expected to take a correct decision in 96% of the interactions. In general, the survey-based simulations show that PADEC provides good results when users are given freedom to create their own locks, rather than sticking to a given baseline.

The third analysis, as depicted in Figure 8, is aimed at assessing whether

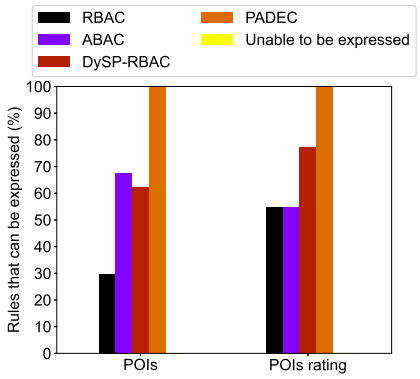

Figure 10: Percentage of pre-made rules selected by users that could be expressed with each access control mechanism.

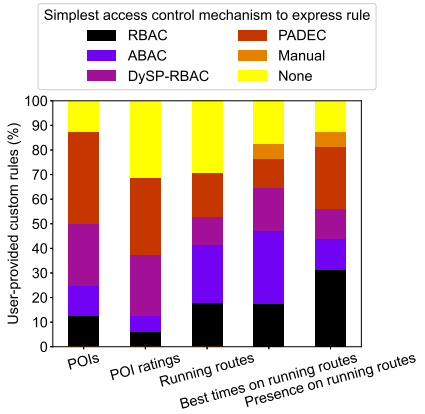

Figure 11: Simplest access control mechanism that can express the custom rules made by users.

the users are interested, as consumers, in knowing this information. In general, the vast majority of users are interested in obtaining this information: in the case of POI rating, all users answered this information is useful. Between 95 and 73% of the users also wish to obtain running route information unconditionally, while in the case of POIs other conditions, such as the socioeconomic status, interest or preferences of the producer, are also considered. The main highlight of this analysis, nonetheless, is that all users are interested in the first four types of information, and 89.5% of them are also interested on knowing the presence information.

Continuing with the analysis, Figure 9 shows with whom the users want to share their route-related information with. In the case of running routes, users would often share them with their friends (79%), runners in the same zone (63%), their family (58%), or runners with the same skill level (42%), while some would also share them with their neighbours (32%). While closed social groups, such as family, friends or neighbours, can be expressed easily through classic access control mechanisms such as RBAC, some popular choices include strangers which meet given criteria. Thus, users have a need for more expressive access control mechanisms, such as PADEC. A similar trend can be seen in the rest of information types: most users find acceptable to share their personal information with their closest circles, such as family (68%, 79%) or friends (79%, 73%), but sharing it with strangers that are in the same zone (63%, 21%) or physically close in the moment they consume the information (52%, 37%) are also popular choices.

The next analysis assesses the ability of the analyzed access control mech-

anisms to express different pre-made rules selected by users. Concretely, for POIs and their rating, users were presented with a variety of pre-made PADEC rules to protect their information, and they had to rate how likely they would be to use each of the rules between 0 and 5. The results shown in Figure 10 depict the access control models needed to express the rules that were rated as 3 or more by users. We find that no rule was unable to be expressed using our analyzed mechanisms, and moreover, PADEC is able to express every proposed rule. Furthermore, users were often interested on filtering the information offered to different types of consumers, and hence, DySP-RBAC is able to express 62% of the POI-protecting rules and 77% of the rating-protecting rules. In the case of POIs, users highly rated rules that allowed strangers to access their information under certain circumstances, and thus, ABAC can express 68% of the proposed rules, in contrast with RBAC, which can only express a 30% of them. In the case of ratings, these rules were often tied to filters, and therefore unable to be expressed with ABAC, which is only able to express the same rules as RBAC (55%).

Finally, the analysis depicted in Figure 11 shows the simplest access control mechanism to express the custom rules that users created and provided in the survey. Out of the four analyzed mechanisms, we consider RBAC to be the simplest, followed by ABAC, DySP-RBAC, and finally PADEC. Moreover, to ensure the fairness of this comparison, we also add two access control models that are even simpler than RBAC: manual, in which the user interactively controls access by manually allowing or denying each request, and none, in which no access control is required because the access to the information is always granted. Starting with the rules that require no access control, in the cases where users find the information more sensitive, such as POIs or presence, only 12.5% of them chose to share the information freely. On the other hand, users find reasonable to share less sensitive information with everyone, such as POI ratings or running routes, with approximately 30% of the rules allowing for free information sharing. The case of manual choice, however, only exists in best times and route presence, where 5.8% and 6.25% of the rules, respectively, require for such access control mechanism. RBAC and ABAC have similar shares, between 6.25 and 30%. Both mechanisms are the least popular for protecting ratings, while RBAC is the most popular for protecting presence and ABAC for protecting running routes and best times. Rules that require DySP-RBAC are slightly more popular in general, oscillating between 12 and 25% and being a more common choice than RBAC and ABAC for protecting POIs and ratings. Finally, PADEC is the most popular for protecting POIs (38%) and ratings (31%). On average, PADEC is the most popular access

control mechanism required to express user-made rules (25%), followed by none (21%), DySP-RBAC (18%), RBAC and ABAC (17% each) and finally manual choice (2%). Furthermore, PADEC is able to express all the rules from simpler access control mechanisms, and thus, it would be able to express 98% of the user-made rules, only being unable to express those that require manual choices.

7. Conclusions

As the penetration rate of companion devices continues raising, their applications in data sharing and mobile crowd sensing also increase. With an userbase increasingly worried about their privacy, allowing users to maintain data ownership and pushing the interactions into the opportunistic space is a reasonable solution. However, users lack mechanisms to constrain the access to their exposed endpoints and to control the privacy of the released information in a decentralized system. Furthermore, the nature of opportunistic interactions calls for allowing users to express arbitrary constraints on context for allowing or denying access to protected information. In this paper, we present how PADEC fills this gap by allowing users to express their arbitrary, context-sensitive access control rules, as well as to link them with data degradation methods. In the future, we expect to integrate privacy-preserving datamining techniques into PADEC's filters. Finally, we will also further analyze the complexity of creating PADEC rules for users, as well as to assist providers by giving them tools to assess the privacy of a certain rule.

Acknowledgment

Map data copyrighted OpenStreetMap contributors and available from `https://www.openstreetmap.org`. This work was supported by the project PID2021-124054OB-C31 (MCI/AEI/FEDER,UE), by the European Regional Development Fund, by the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR21133), by the Valhondo Calaff institution, and by the National Science Foundation (CNS-1703497). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

References

[1] C. Wigginton, M. Curran, C. Brodeur, Global mobile consumer trends, 2nd edition (2020).
URL https://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology-media-telecommunications/us-global-mobile-consumer-survey-second-edition.pdf

[2] EMarketer, US adult wearable penetration rate 2016-2022, Tech. rep., EMarketer (2019).
URL https://www.statista.com/statistics/793800/us-adult-wearable-penetration/

[3] J. Guillén, et al., People as a service: A mobile-centric model for providing collective sociological profiles, IEEE Software 31 (2) (2014) 48–53. doi:10.1109/MS.2013.140.

[4] J. Huang, et al., Blockchain-Based Mobile Crowd Sensing in Industrial Systems, IEEE Transactions on Industrial Informatics 16 (10) (2020) 6553–6563. doi:10.1109/TII.2019.2963728.

[5] M. Abouaroek, K. Ahmad, Foundations of opportunistic networks, Opportunistic Networks: Mobility Models, Protocols, Security, and Privacy (2018).

[6] L. Pelusi, A. Passarella, M. Conti, Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks, IEEE Communications Magazine 44 (11) (2006) 134–141. doi:10.1109/MCOM.2006.248176.

[7] B. Guo, et al., Opportunistic iot: Exploring the harmonious interaction between human and the internet of things, Journal of Network and Computer Applications 36 (6) (2013) 1531 – 1539. doi:https://doi.org/10.1016/j.jnca.2012.12.028.

[8] D. Wu, et al., Security-oriented opportunistic data forwarding in mobile social networks, Future Generation Computer Systems 87 (2018) 803–815.

[9] Y. Huang, A. Shema, H. Xia, A proposed genome of mobile and situated crowdsourcing and its design implications for encouraging contributions, International Journal of Human-Computer Studies 102 (2017) 69 – 80. doi:https://doi.org/10.1016/j.ijhcs.2016.08.004.

[10] K. Olejnik, et al., Smarper: Context-aware and automatic runtime-permissions for mobile devices, in: 2017 IEEE Symposium on Security and Privacy, 2017, pp. 1058–1076.

[11] H.-C. Chen, Collaboration iot-based rbac with trust evaluation algorithm model for massive iot integrated application, Mobile Networks and Applications 24 (3) (2019) 839–852.

[12] A. K. Malik, et al., A comparison of collaborative access control models, Environment 8 (3) (2017).

[13] D. Brossard, G. Gebel, M. Berg, A systematic approach to implementing abac, in: Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control, 2017, pp. 53–59.

[14] H. Ouechtati, N. B. Azzouna, L. B. Said, Towards a self-adaptive access control middleware for the internet of things, in: 2018 International Conference on Information Networking, 2018, pp. 545–550.

[15] P. Mohassel, S. Sadeghian, How to hide circuits in mpc an efficient framework for private function evaluation, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2013, pp. 557–574.

[16] D. Demmler, T. Schneider, M. Zohner, Aby-a framework for efficient mixed-protocol secure two-party computation., in: NDSS, 2015.

[17] E. Hayashi, et al., CASA: Context-aware scalable authentication, in: Proceedings of the 9th Symposium on Usable Privacy and Security, 2013. doi:10.1145/2501604.2501607.

[18] D. Schurmann, A. Brusch, S. Sigg, L. Wolf, BANDANA - Body area network device-to-device authentication using natural gAit, in: PerCom, 2017, pp. 190–196. arXiv:1612.03472, doi:10.1109/PERCOM.2017.7917865.

[19] M. L. Mazurek, et al., Toward strong, usable access control for shared distributed data, in: 12th USENIX Conference on File and Storage Technologies, 2014, pp. 89–103.

[20] S. Shafeeq, M. Alam, A. Khan, Privacy aware decentralized access control system, Future Generation Computer Systems 101 (2019) 420 – 433. doi:https://doi.org/10.1016/j.future.2019.06.025.

[21] S. Popov, O. Saa, P. Finardi, Equilibria in the tangle, Computers & Industrial Engineering 136 (2019) 160–172.

[22] L. Jain, V. Jayesh, Security Analysis of Remote Attestation, CS259 Project Report (Ml) (2011).

[23] E. Bresson, O. Chevassut, D. Pointcheval, Provably secure authenticated group diffie-hellman key exchange, ACM Transactions on Information and System Security 10 (3) (2007) 10–es.

[24] A. McMahon, Discovery of delay-tolerant networking endpoint elements, Ph.D. thesis, Trinity College Dublin (2013).

[25] N. Chomsky, Three models for the description of language, IRE Transactions on Information Theory 2 (3) (1956) 113–124. doi:10.1109/TIT.1956.1056813.

[26] Withlocals B.V., Withlocals - personal tours & travel experiences. URL `https://play.google.com/store/apps/details?id=com.withlocals.Withlocals`

[27] A. Keränen, J. Ott, T. Kärkkäinen, The ONE Simulator for DTN Protocol Evaluation, in: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, 2009.

[28] OpenStreetMap contributors, Planet dump retrieved from https://planet.osm.org , `https://www.openstreetmap.org` (2017).

[29] D. Yang, D. Zhang, V. W. Zheng, Z. Yu, Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs, IEEE Transactions on Systems, Man, and Cybernetics: Systems 45 (1) (2015) 129–142. doi:10.1109/TSMC.2014.2327053.

[30] C. Dwork, Differential privacy, in: International Colloquium on Automata, Languages, and Programming, Springer, 2006, pp. 1–12.

[31] S. Kosta, A. Mei, J. Stefa, Small world in motion (swim): Modeling communities in ad-hoc mobile networking, in: 2010 IEEE Communications Society SECON, 2010, pp. 1–9.