Privacy-Aware and Context-Sensitive Access Control for Opportunistic Data Sharing

Privacy-Aware and Context-Sensitive Access Control for Opportunistic Data Sharing

Juan Luis Herrera*, Hsiao-Yuan Chen†, Javier Berrocal*, Juan M. Murillo* and Christine Julien†
*University of Extremadura, Spain. [jlherrerag, jberolm, juanmamu]@unex.es
†University of Texas, Austin, United States. [littlecircle0730, c.julien]@utexas.edu

***Abstract*—Opportunistic data sharing allows users to receive real-time, dynamic data directly from peers. These systems not only allow large-scale cooperative sensing but they also empower users to fully control what information is sensed, stored, and shared, enhancing an individual's control over their own potentially private data. While there exist context-aware frameworks that allow individual users to define when and what shared information peers can consume, these approaches have limited expressiveness and do not allow data owners to modulate the granularity of the information released depending on a particular peer or situation. In addition, these frameworks do not consider the consuming peers' privacy, i.e., how much information they have to provide to get access to some desired data. In this paper, 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 in order to precisely define who has access to what data when and at what level of detail. Our evaluation shows that PADEC is more expressive than other access control mechanisms and protects the provider's privacy up to 90% more.**

***Index Terms*—opportunistic networks, access control, privacy**

I. Introduction

During the last few years, there has been a massive deployment of mobile devices. Almost every developed country has at least 90% mobile phone penetration and 80% smartphone penetration [1]. Similar trends exist for IoT and other wearable devices, with a projected penetration of 25% in the US by 2022 [2]. These devices carry a large number of on-board sensors that provide their owners with a rich view of the surrounding context and support a wide array of application functionalities. In addition, as these devices are commonly mobile, they accompany the user, providing an interface between their owner and a wider networked community [3].

This astounding increase in *companion devices* has enabled new types of systems and applications, such as mobile crowd sensing [4] or opportunistic data sharing [5], which both 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 [6]. Opportunistic data sharing, which can be combined with mobile crowd sensing, relies on transient wireless network connections [7] to distribute and fulfill a data sharing task using information from the local networked devices. As interactions are pushed into the opportunistic space, individuals need mechanisms to control the release data according to their privacy needs. Privacy management mechanisms must empower users to manage who, when, and how their personal information can be consumed [8], i.e., each user should be able to decide which information is shared with whom, in how much detail, and under which contextual conditions.

Context-aware access controls constrain access to data or resources based on specific contextual conditions [9]. In contrast, basic access control mechanisms, such as Role-Based Access Control (RBAC) [10] or Dynamic Sharing and Privacy-aware Role-Based Access Control (DySP-RBAC) [11], allow data or resource providers to control access to information based on the identity of a potential consumer. These mechanisms are not sufficiently expressive to support applications with opportunistic device-to-device connections, in which identities of most peers are unknown. Other proposals, such as Attribute-Based Access Control (ABAC) [12] or the access control mechanism from [13], 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 having fine-grained control of the access to their data or resources. Furthermore, these systems are designed for centralized (cloud-based) data sharing in which a coordinator negotiates data sharing and access control. This is not the case of in our scenarios, which are often completely decentralized.

This paper presents a Privacy-Aware and Context-Sensitive Access Control mechanism (PADEC), designed for completely decentralized data sharing scenarios. PADEC empowers users by increasing the expressiveness of access rules and protecting the privacy of both providers and consumers. PADEC allows resource owners to define rich access rules based on different kinds of contextual information, as well as to attach filters to these rules in order to provide the data or resources at different levels of granularity. At the same time, PADEC protects the privacy of resource consumers by minimizing the amount of contextual information that they need to share within their access requests. To that end, we introduce the concept of *keyhole* that provides consumers information about the context information they must provide in order to gain access.

This paper is organized as follows. Section II presents a case study that serves as motivation. Section III describes related works on context-sensitive and privacy-aware access control, while Section IV elicits the threat model for our opportunistic data sharing scenarios. Section V incrementally introduces the contributions of PADEC, which is then evaluated over the case study in Section VI, providing comparisons with other access control mechanisms. Finally, Section VII concludes.

II. Case Study Application

To motivate the problem, we present a case study of a mobile crowd sensing application that relies on opportunistic data sharing in a smart city. In this scenario, tourists arriving in a smart city want to find the most popular Points Of Interest (POIs) of the city according to local residents: restaurants that the locals frequent, bars and cafes that are popular, parks that local residents enjoy, etc. To aid visitors, the local government has released an application that leverages users' sensors to store a history of the POIs they have visited and to share this information with other nearby users by relying on hyper-local wireless links. The government has encouraged local residents to use this application as a means to promote tourism. Such a scenario is not just a thought experiment; similar applications have already been envisioned [14]. From an application architecture perspective, there are two roles in the application; the local residents are *providers* of information in the smart city, and the tourists are *consumers* of that information. Concretely, a provider's device exposes one or more *endpoints* that a consumer can access using opportunistic device-to-device communication to retrieve the information.

When the local residents share information directly with nearby tourists, 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, so this information must be carefully protected. With a centralized approach, local residents would need to store their POI history to a centralized server, provided by a third party (e.g., the local government), that has access to the full POI history of every user. This means that users need to trust this third party, since they are transferring the ownership of their POI history. This is not the case with opportunistic data sharing.

Although privacy could be easily maintained by not allowing any user to access a resident's POI histories, this would render the application useless. Thus, while granting open access to everyone is undesirable because of the private nature of the POI history, so is consistently denying access. Users require access control mechanisms that grant access to individual users based on their context, as well as to allow data owners to share their private information at different levels of granularity, depending on that context. For instance, a local resident might not mind sharing a small portion of their history with nearby tourists. If this tourist is a friend of the resident, or part of their family, the resident is likely to be willing to share a much larger part of the history with them because the resident has some implicit trust in the consumer.

On the other side, tourists access the providers' endpoints by applying to the access control mechanism for authorization. To do so, they must provide their own contextual information. While the reduced amount of information released by the consumer makes it less critical than the data released by locals, the privacy of tourists should also be protected. Similar to how the data providers need a tool to select how much information they are willing to share based on the context, tourists need similar mechanisms to restrict their exposure. In this case study application, the visiting tourists will release their own context information for the purpose of getting access to providers' resources or data. In this process, the consumers can be guided by the policies defined by the data providers.

III. Background and Related Work

In the opportunistic data sharing approaches that motivate our work, the providers and consumers that are collaborating may be completely unknown to one another, we do not assume the presence of a central coordinator to help mediate privacy protection, and different parties may have dramatically different privacy sensitivities. In the remainder of this section, we examine existing work that addresses facets of the access control challenge for opportunistic data sharing applications.

Context-sensitive access control. During the last few years, different works focused on using personal context information to authenticate users. In [15], 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 [16], the users' movements, like gait, are leveraged to validate the user's identity. While the identification of users using their context information and attributes has proven useful for unlocking and adapting the behavior of personal devices, it requires an *a priori* model of the attributes or behavior of each user, which is unfeasible for large-scale applications, especially those that need to authenticate and grant access to unknown users depending on their context.

Decentralized access control. Different works propose access control frameworks to access shared data under different contexts. For instance, Penumbra [17] proposes an expressive and decentralized access control system that empowers users and that can be used in opportunistic scenarios in which there is no central mediator. Nevertheless, Penumbra only considers identity in its access control decisions, and omits other contextual information. In [18], the authors propose the use of a distributed ledger, called Tangle [19], to store the policies and access rights for resource-constrained IoT devices. This ensures the distributed auditability of the process. Although this mechanism is also privacy-aware, privacy is only preserved if access control decisions are made by honest participants, which is difficult to ensure in opportunistic scenarios.

Privacy-aware access control. Opportunistic and pervasive scenarios require decentralized and context-sensitive access control mechanisms in which individual users can also modulate the information to release depending on the consumer's context. Existing access control mechanisms have been defined for collaborative environments. The NIST standard RBAC [10] authenticates users based on identity by grouping them with roles. However, RBAC is not context-sensitive nor privacy-aware. DySP-RBAC [11] extends RBAC into a context-sensitive and privacy-aware mechanism. However, DySP-RBAC is implemented applying a pure server/cloud-based architectural style, in which all users are known in advance and users do not retain ownership of their own data. Moreover, context in DySP-RBAC is based on the relationships between users, and it therefore is fully identity-based.

ABAC [12] is a potential model for context-sensitive access control. However, it is still not a privacy-aware mechanism, as it does not allow users to set different levels of detail for the shared information. The work in [13] proposes an alternative mechanism for pervasive scenarios. Making use of semantic technologies, users can define access control policies by providing conditions over context. However, this mechanism also requires a centralized knowledge base that contains contextual 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.

While privacy-aware and context-sensitive access control mechanisms are addressed in the literature, the fact that none of these mechanisms are designed for opportunistic scenarios makes none of them suitable for our envisioned pervasive 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 special needs of opportunistic data sharing for empowering users to manage access to their own data.

IV. Threat Model

In our system model, a user, known as provider, voluntarily shares their data or resources with others, known as consumers, as long as both the provider and the 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 should also be able to filter or obfuscate the information it provides based on similar conditions over context. We assume a model of an attacker whose objective is to obtain information they are not granted access to by exploiting any weakness of the system. Attackers can have up to three roles: they can be consumers, and therefore they try to obtain information from providers; they can be providers, trying to obtain information from their contacts with consumers; or they can be third-party attackers, trying to obtain information from messages exchanged by other providers and consumers.

We build a threat model for decentralized access control for opportunistic data sharing incrementally, first considering general threats that are common to any access control mechanism and then moving to more concrete threats that are specific to our context-aware and opportunistic approach.

General threats:

- *Unauthorized access.* The most basic threat in which a consumer tries to obtain data when the provider desires to deny access.
- *Circumventing context constraints*, consumers get access to information because of their identity while it should be denied according to other contextual attributes.

Privacy threats:

- *Consumer over-exposure.* Providers obtain precise information from consumers using the context information they reveal to request access.
- *Provider over-exposure.* Consumers access information with a finer granularity or higher precision than the provider intended to grant access to.
- *Insider attack.* Consumers get finer-grained information than the provider's policies allow by correlating and aggregating the results of repeated allowed requests.

Third-party attacks:

- *Eavesdrop attack*. The attacker obtains the messages shared between providers and consumers, acquiring privileged information.
- *Replay attack*. The attacker obtains a legitimate message and replays it to get incorrectly granted access.

All of the above threats are directly addressed in our approach. There is another threat that is present in the current version of our approach, and it will be tackled in future work, the *insider multi-type correlation attack.* In this threat, a consumer could correlate different types of contextual information obtained legitimately to obtain other kinds of data not allowed. It is similar to an insider attack, but instead of correlating multiple queries to the same information type, the attacker correlates single queries to multiple information types.

V. The PADEC Conceptual Model

In this section, we present our approach to context-sensitive and privacy-aware access control for opportunistic data sharing. We do this incrementally, adding concepts to the model to address the threats elicited in the previous section. Throughout this section, we designate with underlines the key concepts that are the primary novel contributions of PADEC.

Devices in PADEC take on one of two roles: a data or resource *provider* and a data or resource *consumer*. A provider has some application-level data or service that a consumer desires to access; this data or service is provided through the exposure of an *API endpoint*For instance, in our case study, tourists can obtain the history of a local resident's device by calling an API endpoint that releases it.

A key tenet of PADEC is a strict *separation of concerns* between application-level functionality and PADEC's privacy and access control. From an application perspective, this means that the consumer should perceive that it directly calls the provider's application-level API endpoint; pragmatically, this request passes through PADEC components to determine access, but the application (on both sides) remains unaware of the details. Throughout this section, we use sequence diagrams to depict PADEC's details. The consumers are shown on the left of these diagrams, while the providers are shown to the right. Each side has two threads: the application threads (at the outside) and the access control mediating threads (in the center).

A. Step 0: Encrypted communication

To address the *eavesdrop attack* and the *replay attack*, we start with an assumption that all communications are protected by either symmetric or asymmetric cryptography (e.g., AES or RSA). We assume both devices have strong public and private

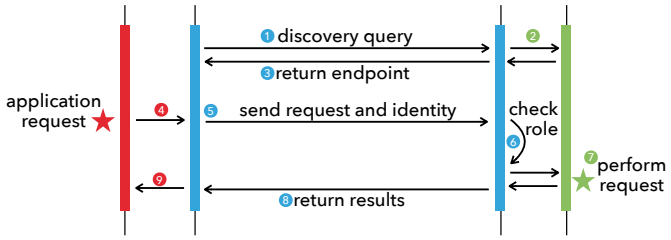

Fig. 1. *Step 1: Providing simple access control based on identity.* A consumer discovers an endpoint in the surroundings (1-3) and makes an application request (4), which is sent to the provider via the available network (e.g., an opportunistic device-to-device connection) (5). The provider checks the consumer's identity against the list of granted roles (6) and, if access is granted, the provider performs the request (7) and returns the results (8-9).

keys, or share a strong symmetric key. These keys are assumed to be shared through a secure, out-of-band channel.

Since messages are encrypted, eavesdropping is not a concern, since the attacker cannot examine the message content. The *replay attack* is also thwarted, since any answer the attacker might get by replaying messages will be as unreadable as any message they overhear via eavesdropping.

B. Step 1: Controlling access to API endpoints

The main, basic concept behind preventing *unauthorized access* is to differentiate users who are authorized to access an endpoint from those who are not. A simple approach is for each endpoint to have an allow list of user identities that are authorized to access the endpoint. Managing this mechanism on an endpoint-by-endpoint basis does not scale: managing multiple lists for different endpoints on an individual identity basis quickly explodes. Therefore, it can be desirable to create *groups*, and, instead of adding individual users to lists, users are added to groups and groups are added to allow lists. The reader might find this system familiar; if the term groups is replaced with *roles*, this is a simple version of RBAC [10]. This concept is shown in Fig. 1.

C. Step 2: Very basic context-aware access control

In many pervasive computing applications, it is necessary to support access control mechanisms that are sensitive to their context. In particular, it may be desirable to provide some level of access to complete strangers who have never before been encountered, as long as some condition on their context holds. Access lists are not sufficient because it is not possible to register an unknown person in a group. Furthermore, this type of access control needs to consider context attributes other than identity, such as the consumer's location or activity. A simple approach is to allow endpoint providers to specify more expressive rules that set conditions over context in order to grant access, e.g., *a consumer must be within 50 meters of the endpoint's position*. To support this approach, consumers must share their context information with the endpoint provider as part of their request, so that the rule can be checked, ensuring that the *circumventing context constraints* threat is mitigated. This system is also familiar, since the same concept is captured in ABAC [12]. This mechanism is depicted in Fig. 2.

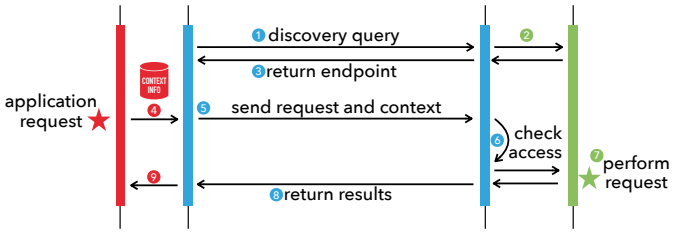

Fig. 2. *Step 2: Extending access control to consider the consumer's context.* The consumer provides its context information as part of the request (4-5), and the provider's process for determining access requires executing the more expressive rules associated with the endpoint (6).

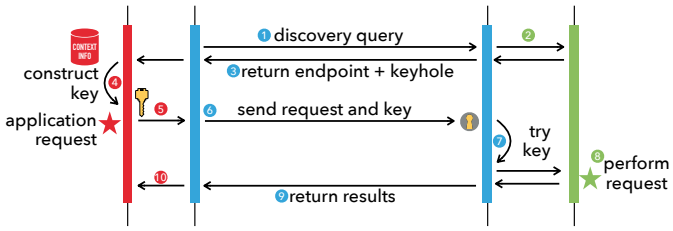

Fig. 3. *Step 3: Keys and keyholes.* In this step, PADEC adds *keyholes* (3), which define the context information required to access the endpoint, and *keys* (4-6), which a consumer application constructs to encapsulate the context information requested. The provider's keyhole defines a rule that validates the provided key (7) before granting access.

D. Step 3: Keys and keyholes

While the previous step enables a provider to consider context in access control, the approach has a few limitations. From a privacy standpoint, this requires the consumer to release *all* of context information that might be relevant without relying on any information about what might actually be needed or used. From a system efficiency standpoint, this is also an overhead in terms of communication cost. While this overhead may be negligible in a centralized system, where the context information may be stored and checked on a cloud server, it may prove prohibitive in opportunistic networks.

To reduce the amount of shared context information both for privacy and overhead reasons, the next step is to define a keyhole associated with an endpoint. The keyhole provides an access point for the API endpoint on the provider side and defines one or more contextual attributes that the consumer is required to provide to gain access. When the consumer discovers a nearby available endpoint (step (1) in Fig. 3), the consumer simultaneously discovers the associated keyhole. To access the endpoint, the consumer assembles a key for each attempted access via the keyhole; the key contains the consumer's current contextual values for the attributes requested by the keyhole. Key construction may also be constrained by the consumer's own local policies, which may limit or prevent the release of certain personal information.

On the provider's side, using only the name of the invoked endpoint and the data in the key, PADEC's access control mechanisms determine whether the access is granted. Practically, each keyhole is associated with a rule that is evaluated at run-time over the consumer's key. With this approach, only relevant contextual attributes have to be released, lowering the

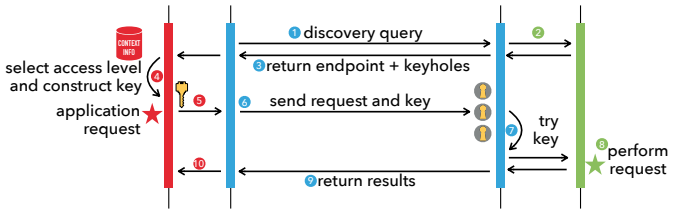

Fig. 4. *Step 4: Providing access options.* Discovery now includes multiple keyholes for each endpoint (3), and the consumer chooses one to target (4-6).

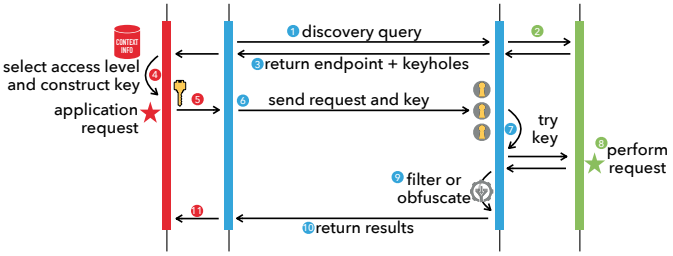

Fig. 5. *Step 5: Increasing providers' privacy flexibility.* After completion of the application-level API call, PADEC applies a filter associated with the selected keyhole (9) before returning the results to the consumer (10-11).

amount of revealed information and the size of keys that are communicated. This step aims to prevent the *consumer over-exposure* threat described previously. In addition, keys are not revealed to the application-level provider, but remain solely within PADEC, further protecting the consumer's privacy. On the other hand, keyholes impose an overhead, as consumers need to query for the keyhole of an endpoint before sending the key. These exchanges are depicted in Fig. 3.

E. Step 4: Providing access options

As a next iterative step, PADEC can allow a provider to set multiple rules (and hence, define multiple keyholes) to control access to a single endpoint. The goal is the corollary of the previous step, i.e., to prevent *provider over-exposure*. For instance, the rule *the consumer must be within 50 meters of my position* or *belong to the group "friends" to access this endpoint* can be decomposed into two, one that requires the consumer's location and the other that requires the consumer's identity. Before decomposition, the keyhole for the above rule would request *both* location and identity; after decomposition, the consumer sees two different keyholes and can choose which information to provide. In PADEC, we refer to these different keyholes as access levels; when a consumer discovers the endpoints, the provider also returns the keyholes of all the access levels, and the consumer can choose which one they want to attempt to access. This refined process is shown in Fig. 4. A consumer can choose the order in which it attempts to access the available keyholes, depending on the consumer's own sensitivity to private context information.

F. Step 5: Increasing providers' privacy flexibility

Up to this point, if a consumer gets access to an endpoint through any of its keyholes, the consumer gets access to all the information in that endpoint. The next layer places filters on top of access levels and their associated keyholes. In particular, a provider can attach a different filter to each access level exposed by an API endpoint; the filter can be used to abstract or obfuscate the results returned from the endpoint. Thus, rather than having to duplicate endpoints to provide different levels of access, the provider can use a single endpoint but post-process the results before returning them to the consumer. In this way, PADEC separates the application-level functionality of the provider endpoint from the concerns associated with privacy and access control, which are implemented within the keyholes and filters. This additional step is shown in Fig. 5.

In PADEC, a filter can be any technique used to limit the precision of the information returned by an endpoint. PADEC has basic built-in filters, but these are also user-definable by injecting small pieces of code or tailored parameters.

Finally, to assist consumers in selecting appropriate keyholes, the discovery information that is returned to the consumer (i.e., step (2) in Fig. 5) includes the achievable level of precision for each of the available keyholes. To avoid *insider attacks*, filters should be implemented in such a way that guarantees idempotence, i.e., repeated access grants to the endpoint should consistently provide access to the same information after obfuscation as long as the unobfuscated information does not change.

G. Step 6: Negotiating multi-level access

Finally, as we discussed above, there is a limitation in the design of PADEC's keyholes and access levels: a consumer cannot tell in advance which access levels will be granted and which will be denied, which may result in several back-and-forth interactions as the consumer tests different keyholes to attempt to gain the needed access. This has potential significant negative ramifications in terms of communication overhead, especially as the application scales in size.

To mitigate the impact of this limitation, PADEC includes a negotiation component. Rather than providing exactly a single matching key in each request, consumers in PADEC can provide a personalized key that the provider can try against multiple keyholes. Upon receiving the request, the provider will try to use the key in the access level with the highest precision. If access is not granted, the provider will automatically resort to the next access level, and so on. If no attempt succeeds, access is effectively denied.

H. A Final Note about encrypted communication

As we described at the beginning of this section, our iterative design of PADEC assumes that all communications are encrypted, to ensure protection against eavesdropping and replay attacks. These mechanisms rely on cryptographic keys exchanged among all pairs of devices out-of-band. However, to keep all interactions purely opportunistic, it is possible to embed the key exchange into the messages sent as part of PADEC by integrating a key exchange protocol similar to Diffie-Hellman [20] into the endpoint discovery process. Taking the protocol in Fig. 5 as the final PADEC model. The consumer can send its public key with its discovery query

in (1). As part of returning the endpoint and keyholes, the provider can create a new symmetric key and encrypt it with the consumer's public key before returning it. The consumer (and only the consumer) has the associated private key, which it can use to retrieve the symmetric key. This symmetric key can be used to encrypt the subsequent communications in the exchange (in particular exchanges (6) and (10) in Fig. 5).

VI. Evaluation

To evaluate PADEC, we first measure the trade-off between the *privacy level* achieved by the system and the successful accesses to endpoints in each of the steps explained in Sec. V. We then quantify the *overhead* of the system in each of the steps. Third, we explicitly compare the *expressiveness* of PADEC with that of alternative mechanisms, namely RBAC and ABAC. In fact, the first sets of metrics – privacy, successful accesses, and overhead – also allow comparison across PADEC, RBAC, and ABAC, since Step 1 in the PADEC protocol is equivalent to RBAC and Step 2 is equivalent to ABAC. Finally, we provide a discussion of PADEC's success in addressing the *threat model* described in Section IV.

Our primary mode of evaluation is through the implementation of PADEC as an application layer in the ONE [21] simulator. We incorporated a streetmap of New York City from OpenStreetMap [22], in which the network nodes are of two kinds: tourists visiting the city and local residents. From a PADEC architectural perspective, tourists are consumers of data, and their movements are constrained to the touristic zones of the map (i.e., central downtown squares). The nodes representing local residents are PADEC providers that roam through touristic and residential zones of the city. As our evaluation moves into evaluating our threat model, a subset of each group is designated as attackers. The simulated scenarios consider 100 local residents and 50 tourists, and each simulation runs for 24 hours of simulated time. In the threat model evaluation, we consider 90 honest locals, 40 honest tourists, 10 malicious locals and 10 malicious tourists. Every 50 seconds, if a tourist is not waiting for a response to a request, the tourist chooses a random connected local and starts a PADEC communication, using a simple broadcast communication across the opportunistic network. It is important to note that both tourists and locals may keep moving during these interactions, and thus, their messages may be routed through the opportunistic network instead of being delivered directly.

As data to support the application scenario, we construct POI histories for the providers using a set of FourSquare check-ins in New York City between April 2012 to February 2013 [23]. We generate a realistic POI history for each simulated resident using the anonymized user IDs in the dataset. Each provider has an average of 210 check-ins, and the full dataset contains 227,428 check-ins. Each consumer collects three pieces of context that can be shared with providers to gain access to their POI histories. Each type of context has a different category of sensitivity from the consumer's perspective. Higher categories are considered to be more

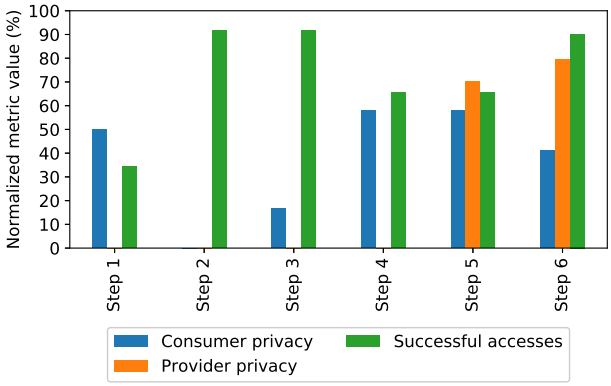

Fig. 6. Privacy vs. successful accesses trade-off in each PADEC step.

sensitive, i.e., to reveal more private information. 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). For our initial experiments, in Steps 1 through 3, we assume that all tourists are willing to reveal any context information to gain access to a provider's POI history endpoint. From Step 4 onwards, we assume that half of the tourists remain willing to reveal any information, while the other half will only reveal categories 1 and 2 (but not the category 3 identity information).

Each of the six steps is also associated with one or more rules defined by the provider to constrain access to the endpoint. For each of the six steps, we use the following rules:

Step 1 The provider grants access only to users whose identities place them in the *friend* group (14% of the tourists) or *family* group (6% of the tourists).

Step 2 The provider also grants access to *any* tourist within 500 meters of the provider.

Step 3 Same as Step 2.

Step 4 The rules are separated into two access levels (one for friends and family and a second for nearby strangers).

Step 5 Filters are associated the defined levels. The results for friends and family are not filtered, while those for nearby strangers only shows POIs visited at least three times between October and January.

Step 6 Same as Step 5.

Privacy vs. access. Our first evaluations consider the trade-offs that users navigate with respect to revealing their private information versus gaining or granting access. Fig. 6 shows these trade-offs for each step of PADEC. We present three metrics: consumer privacy, provider privacy, and number of successful accesses. Consumer privacy is based on 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. Provider privacy is based on the average precision of the information shared, so 0% means all of the raw POI information is shared (100% precision) and 100% means no POI information is shared (0% precision). Finally, successful accesses computes the percentage of the consumers' requests that were successfully answered.

Step 1 only considers identity, so the exposure of the con-

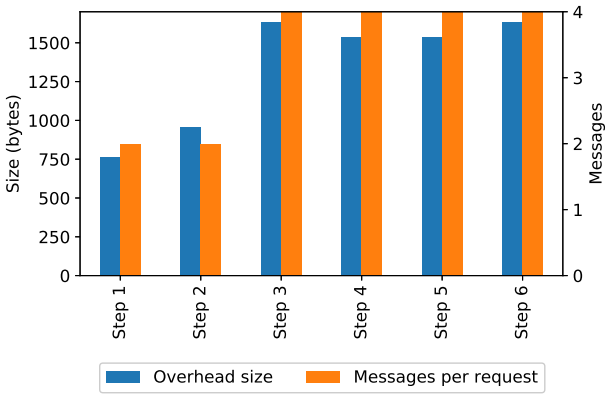

Fig. 7. Overhead in size and number of messages for each PADEC step.

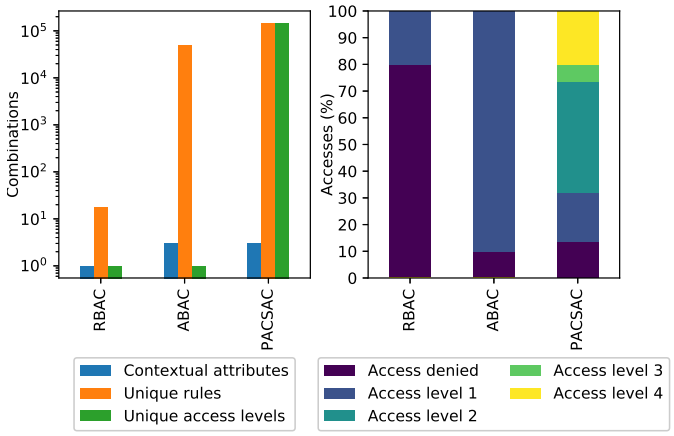

Fig. 8. Comparative expressiveness of RBAC, ABAC and PADEC.

sumers reflects the sensitivity of the context released, but also the fact that two context types are not shared. Because only family and friends are granted access, 65.57% of requests are denied. In Step 2, using other contextual attributes dramatically increases the percentage of fulfilled requests, but a consumer's privacy is completely exposed (i.e., all consumers share all of their context). In Step 3, the keyhole allows consumers to reveal slightly less information in their keys, increasing their privacy with no impact on successful accesses. The two access levels in Step 4 greatly increases the privacy of consumers, as they only have to reveal one contextual attribute; but consumers try the access level with the best precision given the context data they are willing to release. This means many try to access through the keyhole that requires identity, but they are not always friends and family, so 34.42% of the requests fail. Step 5 adds filters, which greatly improves the privacy of providers and has no impact on the other two metrics. Finally, the negotiation algorithm of Step 6 greatly increases successful accesses, getting up to 90.16% of the requests being satisfied, with a modest impact on consumer privacy.

Overall, while the increase of consumer privacy is not as large as the increase of provider privacy, there is an important feature of PADEC to consider: by design, the contextual information shared in the key can only be accessed by PADEC. The underlying applications are unable to access this information, so providers running malicious applications cannot collect data from the consumers. Compared to the information shared by providers, which can be read by underlying applications, consumer data is much more protected.

Overhead of PADEC. We consider two metrics for overhead: the number of messages sent during an exchange like the one shown in Fig. 5 and the total amount (in bytes) of data exchanged. Fig. 7 shows the overhead. While RBAC and ABAC (as Steps 1 and 2) have a low overhead, PADEC almost doubles this overhead in the later steps. PADEC has to share four messages (keyhole request, keyhole information, key, and response), whereas RBAC and ABAC only need two (authorization request and response). Although these two additional messages increase the overhead, the total overhead of PADEC in the context of this application scenario is 1.63 KB.This doubling of the communication overhead is the primary price an application pays to be able to leverage PADEC's capabilities. However, since the overhead of ABAC and RBAC is already very low considering current communication technologies, doubling the overhead is acceptable.

Comparative assessment of expressiveness. To evaluate the *expressiveness* of the relevant access control models, we compare the performance of ABAC and RBAC to a simulation in the same scenario featuring a PADEC endpoint with five access levels. Though PADEC allows for further access levels, these five levels are sufficient to demonstrate PADEC's expressiveness. We constrain ourselves to RBAC and ABAC because they are implementable in an opportunistic device-to-device network; DySP-RBAC, in contrast, requires a central registry with data that relates different users.

Our comparison is shown in Fig.8, which shows a theoretical comparison on the left and a practical demonstration on the right. From a theoretical perspective, we characterize the size of the rule space, i.e., how many different rules could be specified in each approach. We consider the use three context attributes as well as four operators and two combinational operators (and, or) to join rules. If we limit rules such that each combinational operator can be used only once (preventing infinitely long rules), we can compute the total possible number of attributes, rules, and access levels, as shown on the left of Fig. 8. Since ABAC and PADEC allow the use of contextual attributes other than identity, they are more expressive than RBAC. PADEC is, theoretically, the most expressive, nearly tripling the number of possible rules in comparison to ABAC. Furthermore, PADEC allows each rule to be associated with a filter, allowing multiple levels of privacy for each endpoint, while RBAC and ABAC can only set a single rule over an endpoint.

Threat model mitigation. Finally, we examine the degree to which PADEC addresses the threat model in Section IV. We used a scenario in which 10 consumers are attackers and 10 locals are attackers assigned each of the threats. Since it is not possible to access the key from an underlying application, *consumer over-exposure* attacks have been implemented in third-party nodes. In total, 37,109 attempts of various at-

tacks were performed for *circumventing context constraints*, *consumer over-exposure*, *eavesdropping* and *replay* attacks. *Unauthorized access* attacks were proven to fail in the previous simulations, since accesses were denied. As for *insider attacks*, filters for each access level are, by design, idempotent, which makes it impossible to correlate responses from the same access level, as responses do not contain different information from one another. A total of 0 attempts were successful in the simulation, and therefore, we conclude that PADEC is resistant against attacks from the proposed threat model.

We have validated PADEC against the case study in Section II. PADEC provides tools for consumers and providers to share data opportunistically. The results show that PADEC can be leveraged to increase privacy with a negligible overhead compared to alternative access control mechanisms.

VII. Conclusion and future work

As interest in systems that leverage sensors of companion devices (such as the interest in mobile crowd sensing) continues to grow and develop, users become more wary about privacy of their data. To empower them and maintain data ownership, opportunistic data sharing systems have evolved, but they require expressive, context-sensitive, and privacy-aware access control mechanisms. We developed PADEC to directly address these concerns, protecting the privacy of providers and consumers in opportunistic scenarios with a much higher expressiveness and minimal overhead compared with alternatives such as ABAC or RBAC. In the future, we expect to address the *insider multi-type attack*.

Acknowledgment

Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org. This work was supported by the projects RTI2018-094591-B-I00 (MCI/AEI/FEDER,UE), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR18112, IB18030), by the European Regional Development Fund, 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, and C. Brodeur, "Global mobile consumer trends, 2nd edition," 2020. [Online]. Available: 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," EMarketer, Tech. Rep., 2019. [Online]. Available: https://www.statista.com/statistics/793800/us-adult-wearable-penetration/

[3] J. Guillén, J. Miranda, J. Berrocal, J. García-Alonso, J. M. Murillo, and C. Canal, "People as a service: A mobile-centric model for providing collective sociological profiles," *IEEE Softw.*, vol. 31, no. 2, pp. 48–53, 2014.

[4] B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to Mobile Crowd Sensing," in *2014 IEEE PerCom Workshops*. IEEE Computer Society, 2014, pp. 593–598.

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

[6] J. Huang, L. Kong, H. N. Dai, W. Ding, L. Cheng, G. Chen, X. Jin, and P. Zeng, "Blockchain-Based Mobile Crowd Sensing in Industrial Systems," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6553–6563, oct 2020.

[7] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks," *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134–141, nov 2006.

[8] Y. Huang, A. Shema, and H. Xia, "A proposed genome of mobile and situated crowdsourcing and its design implications for encouraging contributions," *International Journal of Human-Computer Studies*, vol. 102, pp. 69 – 80, 2017, special Issue on Mobile and Situated Crowdsourcing.

[9] K. Olejnik, I. Dacosta, J. S. Machado, K. Huguenin, M. E. Khan, and J. Hubaux, "Smarper: Context-aware and automatic runtime-permissions for mobile devices," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 1058–1076.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Computer role-based access control models," pp. 38–47, feb 1996.

[11] A. K. Malik and S. Dustdar, "Enhanced sharing and privacy in distributed information sharing environments," in *2011 7th International Conference on Information Assurance and Security (IAS)*. IEEE, 2011, pp. 286–291.

[12] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, feb 2015.

[13] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila, "A semantic context-aware access control framework for secure collaborations in pervasive computing environments," in *Lecture Notes in Computer Science*, vol. 4273. Springer, Berlin, Heidelberg, 2006, pp. 473–486.

[14] Withlocals B.V., "Withlocals - personal tours & travel experiences." [Online]. Available: https://play.google.com/store/apps/details?id=com.withlocals.Withlocals

[15] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley, "CASA: Context-aware scalable authentication," in *Proceedings of the 9th Symposium on Usable Privacy and Security*. New York, New York, USA: ACM Press, 2013, p. 1.

[16] D. Schurmann, A. Brusch, S. Sigg, and L. Wolf, "BANDANA - Body area network device-to-device authentication using natural gAit," in *PerCom 2017*. Institute of Electrical and Electronics Engineers Inc., may 2017, pp. 190–196.

[17] M. L. Mazurek, Y. Liang, W. Melicher, M. Sleeper, L. Bauer, G. R. Ganger, N. Gupta, and M. K. Reiter, "Toward strong, usable access control for shared distributed data," in *12th USENIX Conference on File and Storage Technologies*, 2014, pp. 89–103.

[18] S. Shafeeq, M. Alam, and A. Khan, "Privacy aware decentralized access control system," *Future Generation Computer Systems*, vol. 101, pp. 420 – 433, 2019.

[19] S. Popov, O. Saa, and P. Finardi, "Equilibria in the tangle," *Computers & Industrial Engineering*, vol. 136, pp. 160–172, 2019.

[20] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably secure authenticated group diffie-hellman key exchange," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 3, pp. 10–es, 2007.

[21] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[22] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[23] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 129–142, 2015.