Privacy-Centric Modeling and Management of Context Information

Florian Dorfmeister, Sebastian Feld, Claudia Linnhoff-Popien
Mobile and Distributed Systems Group
Ludwig-Maximilians-Universität München
Munich, Germany
Email: {florian.dorfmeister, sebastian.feld, linnhoff}@ifi.lmu.de

Stephan A. W. Verclas
T-Systems International GmbH
Munich, Germany
Email: stephan.verclas@t-systems.com

***Abstract*—Context-aware computing has been an intensively researched topic for years already. Consequently, there exists a plethora of usage scenarios for context-aware applications as well as several approaches for the modeling and management of a user's context information, many of which focus on the efficient and scalable distribution of the latter. With the ongoing rise of smartphones as everyday mobile devices and their steadily increasing amount of sensing and communication capabilities, we finally find ourselves at the edge towards a widespread usage of these techniques. However, apart from technical issues such as how to reliably determine a user's current context, privacy still remains a crucial factor for these systems' acceptance rate. Therefore, inspired by earlier works on privacy in context-aware computing and the authors' beliefs in the necessity to put users in control, this paper presents a novel approach for modeling and managing a mobile user's context information in a user-centric and privacy-preserving way. To this end, this work's contribution is twofold: First, based on widely recognized requirements for privacy in context-aware applications, we propose a privacy-centric context model which allows for an intuitive and context-dependent definition of a user's privacy preferences, directly integrating privacy aspects into the context model itself. Second, we present a generic and flexible architecture for the management and distribution of context information in a privacy-preserving way fit for a multitude of different usage scenarios.**

***Keywords*—context-awareness; context modeling; privacy-by-design; context-dependent privacy policies; context obfuscation.**

I. INTRODUCTION

Both privacy-preserving computing of personal data as well as the automatic extraction of context information – possibly from a smartphone's sensor readings – are very active areas of research. One can think of many use cases for context-aware applications, such as proactive route planning services taking into consideration the current traffic volume and a user's appointment schedule, smart mechanisms automatically adjusting a phone's audio profile based on the user's current activity and occupation, or buddy finder apps alerting the user when sharing the current location with close friends of her. In addition, there are applications such as the SmartBEEs context aware business platform [1], which do not act based on a single user or a peer-to-peer basis, but leverage the combined knowledge of multiple users' current contexts and their surroundings' state, e.g., for business process optimization.

Quite a number of slightly varying interpretations of what context actually is can be found in literature. We base our understanding of context on the famous definition given by Abowd et al. [2], declaring context to be any information that can be used to describe the situation an entity resides in. Strictly following a user-centric approach, for this paper we assume a user's smartphone to be the primary source of information about her current context. How different kinds of context information can be acquired from sensor data is not within the scope of this work though. In order to offer high levels of service quality, algorithms for context recognition typically aim at maximizing the resolution, freshness and accuracy of their classification results. However, when talking about preserving a user's privacy, different – and sometimes even contradicting – objectives are to be pursued. For example, in many situations it might be perfectly sound to deliberately reduce the resolution of a piece of context information before sharing it with others in order not to reveal too much. The privacy issue gets additionally tightened given the fact that due to the popularity of smartphones, we are heading towards a full supply of small electronic devices with broadband internet access and extensive sensing capabilities. Enabling the acquisition of a user's context information with the help of her smartphone's sensors and eligible reasoning mechanisms in return also enables spying on this person. Thereby "one person's sensor is another person's spy", as [3] puts it.

Many approaches for modeling and managing context focus on the generation, processing and efficient distribution of context information as well as the realization of context-aware applications built thereon. Some works, however, argue that not only the usability and utility of context-aware applications are paramount for a wide acceptance, but also the establishment of appropriate privacy mechanisms which make the users feel comfortable within such ubiquitous computing environments. Concordantly, in this paper we propose a new method for giving the full control over the release and resolution of personal, private or characterizing context information to the data's owner. Our work's contribution is twofold: First, we define our *Context Representations* (CoRe) model, which presents a novel approach for modeling a user's context information in a privacy-centric manner. Second, we introduce the design of our *Layered Architecture for Privacy Assertions in Context-Aware Applications* (ALPACA) together with its privacy levels and describe the flexible interplay of the different components.

The remainder of this paper is structured as follows: In Section II, we will make our problem statement and present a comprehensive list of privacy requirements. We will then have a look at related work on privacy in context-aware applications in Section III. Section IV presents our privacy-centric CoRe model, which marks the base component for our logical and physical system architecture described in Section V. After discussing our results in Section VI, we conclude.

 ISBN: 978-1-61208-306-3

II. Problem statement and requirements

Given these preliminaries, with this work we aim at designing a generic solution for the modeling and management of a user's static and dynamic context information. Consequently, we consider it essential to primarily focus on putting the user in full control over the capturing, release and resolution of her personal data. As already stated before, we assume a user-centric, smartphone-based approach here. Hence, from a privacy-oriented point of view the less data is going to leave the user's mobile device, the better. Yet in order to enable a multitude of context-aware applications and services, there is usually a need for communicating one's current context information to other parties. For privacy reasons, however, we argue that there must not be any party but the user herself able to access or control her complete context information at any point in time, thereby ruling out any solutions based on a trusted third party approach. On the other hand, proactively providing a central component with some kind of "handpicked" context information seems nonetheless desirable in some situations, e.g., in order to allow for the efficient realization of multi-subject context-aware applications. A truly generic solution should hence be able to serve all kinds of context requesters while never sacrificing too much of a user's privacy.

Based on existing works on privacy in context-aware applications [4]–[7], we have identified and complemented a set of different techniques and requirements for realizing different aspects of privacy. Naturally, a comprehensive approach for context modeling and management should incorporate all of these mechanisms. In the following, we will name and briefly explain the most important of these requirements:

- A minimum set of **access control** operators such as $grant$ and $deny$ has to be available in order to be able to define different requesters' access rights. Additionally, users should be allowed to define their privacy preferences in a **context-dependent** way.
- **Variable granularity** can be used to reduce a piece of context information's resolution or accuracy. As an example, consider a user reading her e-mails. Different granularities of her currently modeled activity context might contain `read-e-mail`, `computer-work`, `office-work` and `working`.
- **Intentional ambiguity** and **plausible deniability** can be used in order to lower the confidence and validity of a piece of context information, respectively. Notice that these kinds of "white lies" present everyday actions in the offline world such as, e.g., not answering telephone calls in order to pretend not to be present.
- **Adjustable freshness** and **temporal resolution** are other means for intentionally reducing a piece of information's quality, e.g., regarding its age or capturing time. It can hence be used in a similar way as intentional ambiguity to obfuscate a user's context.
- We define **consistency** of a user's privacy preferences as another important requirement, stating that a context requester must not be able to retrieve ambiguous or contradicting pieces of context information.
- **Notifications** can optionally be sent to a user upon each request of her context information. This can hence be used as a social means able to prevent the intentional abuse of contextual information.
- **Symmetry** is especially important in peer-to-peer scenarios, stating that a certain party has to reveal just as much of its own information as it requests.
- Considering that completely denying a request for a piece of a user's context information might itself reveal much, we define **completeness** to be the principle of answering any request with a plausible response.
- Other useful concepts in context-aware applications are **anonymity**, **pseudonymity** and **k-anonymity**.

Additional requirements are security, scalability, extensibility and usability. In order to seamlessly protect a user's privacy, it also seems beneficial to closely band together the modeling and management of a user's context information. With this work's problem statement and privacy requirements being established, we will review related work in the next section.

III. Related Work

This section presents related work on privacy mechanisms for context-aware applications. Several different categories of approaches can be found in literature. Some of them rely on trusted third party (TTP) solutions for efficient context dissemination, whereas other systems adopt a peer-to-peer (P2P) based approach in order to avoid such central points of attack. On the other hand, there are rule languages for defining access control based on contextual information as well as different obfuscation techniques for adequately reducing the richness of a user's context information before their release.

A common architectural model for the realization of context-aware applications is the use of a TTP acting as some kind of middleware for the aggregation of its users' context information. It is necessary for all of the system's participants to fully trust this component. The CoPS architecture introduced in [7] implements such a central privacy service. While allowing for different granularities of context items, it does not permit the definition of context-dependent access rules. Another TTP-based approach called CPE [5] enables the definition of context-dependent privacy preferences, but lacks mechanisms for releasing information in different granularities. With a focus on context-dependent security policies, the CoBrA platform [8] deploys the Rei policy language [9] in order to enable the definition of access control rules depending on a user's current context. Beyond that there is much literature on different techniques for the obfuscation of contextual information. As an example, [6] presents another centralized approach focussing on context ownership and offering obfuscation mechanisms for several kinds of context information based on SensorML process chains, obfuscation ontologies and detailed taxonomies describing dynamic granularity levels. In contrast to these systems, purely P2P-based approaches such as [10] get along without any central component. As a major drawback, such architectures can hardly be efficiently deployed in applications depending on up-to-date context information of a whole group of users at the same time.

Hence, to the best of our knowledge we are not aware of any approach able to fully satisfy the requirements given in Section II and tackling the issue of preserving a user's privacy

 ISBN: 978-1-61208-306-3

in context-aware applications comparable to our framework's user-centric design. Therefore and based on these requirements, we will now introduce our privacy-centric CoRe model and the ALPACA system architecture for context modeling and management in Sections IV and V, respectively.

IV. Privacy-centric context modeling

Based on the given requirements, we will now present a novel approach for modeling a user's context information in a privacy-preserving way. As the name implies, our *Context Representions* model is designed to store several heterogeneous representations of one and the same kind of context information, each of them being intended for variably trustworthy groups of requesters. Subsequently, we introduce a user-friendly and flexible trigger mechanism that allows for context-dependent definitions of privacy preferences on behalf of the user. To this end, the trigger functionality has been devised to (optionally) depend on both the subject's, the requester's as well as the environments' current context. In Section IV-C we will join the latter with our model in order to describe how our approach is able to dynamically cope with inconsistent policy definitions at runtime.

A. Modeling context information using representations

In accordance with many existing approaches for context modeling (cf. [11] for a comprehensive survey), we adopt a tree-based view on a user's overall context information as depicted in Figure 1. In this hierarchic scheme, the root node aggregates all different categories of a user's context information. At the second level distinctions between the basic context categories are made, such as a user's current location or activity. However, this catalog is not fixed and can readily be extended to hold any additional kind of context information in case new types of sensors or inference mechanisms become available. Each of the tree's second-level nodes may be a parent to an arbitrary number of corresponding *Representations*, each reflecting the given category in a different way, e.g., concerning the respective item's resolution or – in case of a white lie – maybe even its validity. One should thus notice that in contrast to existing context models such as MUSIC [12], which use the term "representation" in order to label the data formats used for communication (such as XML, JSON, etc.), we define distinct *Representations* of a context information to differ from each other on a semantic level, independent from any encoding. As an example, consider the three different *Representations* of the user's current location in Figure 1: The representation on the left holds the exact GPS position fix of the user. In contrast, the one in the middle only states the user's location on a city level, while the third uses a non-geographic, symbolic location identifier that cannot be mapped to a geographic one – at least not without any further knowledge about the user. The idea behind providing multiple representations of the same category is that from a privacy-centric point of view, a user should be able to communicate different versions of her current context information to different parties. Which representation is to be released to whom might, e.g., be based on the trust level assigned to a requester and the current context itself.

Following an ontology-based modeling approach, our context model consists of the three base classes shown in Figure 2, namely $Context$, $Representation$ and $Audience$. The

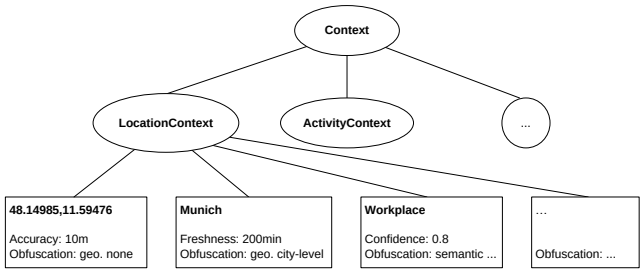

Fig. 1. An example context tree with its second-level *Context* nodes linking to an arbitrary number of *Representations*, each indicating its level of obfuscation based on the semantics of the underlying information.

first two of them can have subclasses such as *ActivityContext* and *ActivityRepresentation*, respectively. As already explained, a certain subclass of context may be described by multiple instances of *Representation* at the same time. Our model's basic structure was inspired by the ASC model by Strang et al. for enabling service interoperability based on a shared understanding of and transformation rules for different, yet logically equivalent scales [13]. Quite the contrary, however, our CoRe approach can be used to model different representations of the same kind of context information, which – according to the requirements stated in Section II – do not necessarily have to share the same (or at least similar) meanings at all. Especially, in our case there must of course not exist any transformation rules which allow for a simple conversion from a low-resolution representation to a high-resolution one.

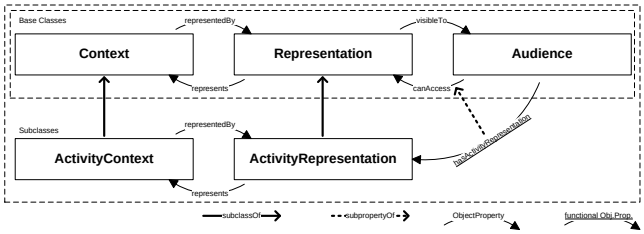

Fig. 2. The classes, subclasses and properties of the CoRe model. Each subclass of *Context* might have an arbitrary number of *Representations*.

As an enabler for the definition of privacy preferences based on our model, a representation instance can be assigned to a certain *Audience* using subproperties of the *canAccess* property. An audience can be any entity requesting the user's context information and might be defined both statically (e.g., based on group membership) and context-dependent, such as "entities near me". As we will see in Section IV-C, however, an entity must not be assigned to more than one representation of the same class of context information at the same time. We accomplish this by marking every subproperty of *canAccess* as being *functional*. In order to be able to automatically assess the resolution of a representation of a given class, each *Representation* stores additional information about its *obfuscation level*. A generic labeling approach for these levels seems unfeasible, considering the great differences in semantics that different classes of context information might possess. Hence, each subclass of *Representation* is expected to define its own obfuscation scales. Returning to the abstracted tree-based view of our model, the base class *Context* can be interpreted as the root node in the hierarchy. Subclasses thereof, such as

 ISBN: 978-1-61208-306-3

ActivityContext and *LocationContext* form the second-level nodes, which might link to an arbitrary number of appropriate subclass instances of *Representation*.

B. Defining context-dependent privacy preferences

In addition to the model's base concepts just described, we have designed a trigger mechanism which allows for a simple, yet flexible and effective definition of context-dependent privacy preferences on behalf of the user. With our system following a conservative, whitelist-based approach (cf. Section V-A) in order to protect a user's context information from accidental leakage, explicitly defining a release trigger is the only possibility for a user to share any piece of information with others. The corresponding classes and properties can be seen in Figure 3. Broadly speaking, a *Trigger* can be used for setting up preferences defining how the system should respond to any requests for context information. A trigger fires each time the set of *Conditions* associated with that instance matches the currently known overall situation, possibly taking into account the current states of both the user's, the requester's and the surroundings' contexts. As an *Effect*, one or more *Representation* instances of the requested user become accessible for the given *Audience*. From [7] we have adapted the idea that accessing a piece of context information by a certain requester might have side effects such as notifying the user, as we agree on that being a proper means suitable for possibly containing data abuse. If desired, a user can hence also specify which *SideEffect* should be activated when a certain kind of *Representation* is requested by a certain audience. Additionally, a user is able to define the obfuscation level, accuracy, confidence and freshness properties the now-accessible representation has to fulfill for this audience under the circumstances described in the corresponding conditions.

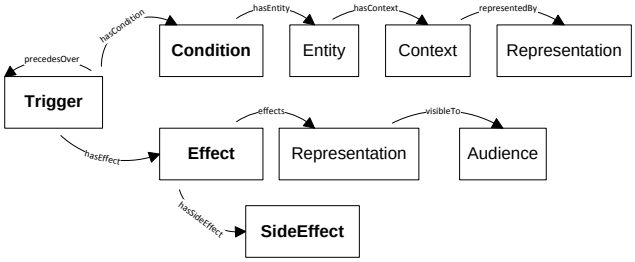

Fig. 3. The structure of the *Trigger* mechanism enabling users to define context-dependent privacy preferences. Each *Effect* might also have *SideEffects* describing reactions to requests for context information.

C. Dealing with inconsistent privacy preferences

We will now describe how our context modeling approach can be used in conjunction with the trigger mechanism in order to detect any inconsistencies arising from ambiguous or conflicting privacy preferences. For this work, we define a set of privacy preferences to be inconsistent if it enables any requesting entity to access more than one representation of the same class of context information at the same time, as this situation is clearly prone to harm a user's integrity. The issue becomes evident when considering two contradicting representations, e.g., a user's true and fake location information. If one requester is granted access to both representations, the result is not only ambiguous, but also likely to negatively influence the user's respectability due to being caught lying. Logically, such situations might occur when a requester is belonging to more than one (possibly dynamically defined) *Audience*. Hence, there has to be a way for detecting and solving such preference conflicts. In order to prohibit such situations from occuring, we use the underlying ontology's built-in reasoning capabilites to dynamically check the model's state for consistency. By defining all subproperties of *canAccess* to be *functional*, we assure that each entity is allowed to see at most one representation of the same class. This check is performed each time a trigger fires. In order to solve conflicting situations, the *precedesOver* property can be set. In a practical implementation, the first time an inconsistent state is detected, a user might be notified about the conflict and be able to choose which of the triggers involved should precede over the other.

This section presented a novel approach for privacy-centric context modeling as well as a trigger mechanism for a context-dependent definition of a user's privacy preferences. In the next section we will describe the privacy layers and components of the ALPACA architecture built upon our model.

V. Privacy Layers and System Architecture

We will now introduce our *Layered Architecture for Privacy Assertions in Context-Aware Applications* and its four different privacy layers. These layers resemble what we believe to be a good compromise about a privacy-aware user's sensation of different levels of information accessibility and reduction of complexity. Afterwards, we will introduce our architecture's central component, the *Privacy Manager* and describe an example setup and communication flow.

A. Different layers for different audiences

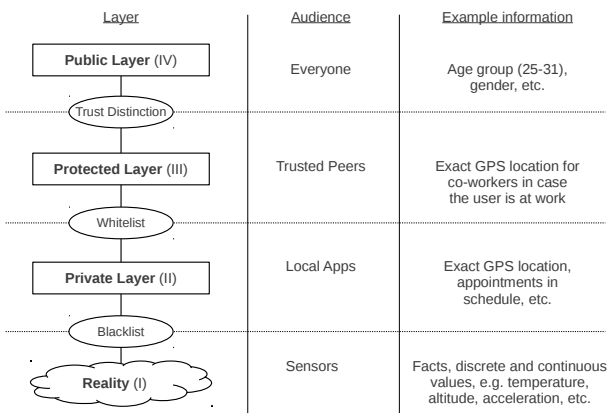

Fig. 4. The four different privacy layers defined in ALPACA and their most likely audiences, as well as some example context items for each layer.

As shown in Figure 4, ALPACA defines four logically different layers which can be mapped to a user's privacy levels, as well as some kind of privacy gateways between these layers. At the bottom layer we put *reality*, possibly containing more information than any kinds of sensors and reasoning mechanisms will ever be able to capture. Although one does not have to implement anything on this layer, we still have to define it as this layer is what is aimed to be reflected in any context model. However, even today a user might

 ISBN: 978-1-61208-306-3

feel uncomfortable knowing that each of her smartphone's sensors is recording data all the time. Hence, a user can set up a context-dependent blacklist for defining which sources of context information should be turned off under certain conditions.

The next layer is the *private layer*, holding all the information a user wants to have available for herself, i.e., context-aware services and applications running exclusively on her mobile device. Consider for example locally run apps which adapt their appearance and behaviour according to the user's current context. In order for such services to be responsive and proactive, this layer enables access to the most fresh and sophisticated context representations. Naturally, these high-resolution representations are probably not intended for everyone else as well. Thus, the trigger mechanism described in Section IV-B is used as a context-dependent whitelist for the release of certain representations to the upper layers.

Context information which pass this whitelist enter the *protected layer* and might hence be available for some other entities, too, such as trusted services and peers. For example, a user might be reluctant to share her current whereabouts with everyone, but maybe with some of her friends in her spare time or with her employee during working hours. Naturally, the number and composition of context representations available on this layer will hence change dynamically based on the user's privacy preferences and current context.

Additionally, a user might be willing to share some kind of information about herself with anyone, meaning that these information are available on the *public layer*. This might, e.g., be true for information that are somehow obvious anyway, such as personal profile data containing the user's gender or age group. However, a user might still define notifications to be displayed when these kinds of information are being requested. That said, notice that our system's user is of course not forced to abide by these layers in the way we just described, but rather can individually choose which level of visibility fits her own situation by the use of appropriate release triggers.

B. Components of our hybrid system architecture

We will now briefly describe the flexible architecture of the ALPACA system, as well as the *Privacy Manager* as its core component responsible for managing access to all of the user's context information. Figure 5 gives a component-oriented view of our system. It is up to the privacy manager instance running on the user's mobile device to enforce compliance with the privacy preferences set up by the user at any time. Therefore, it is the only component which has full access to all representations available in the context model in order to be able to fulfill the user's blacklist and whitelist preferences. The privacy manager decides on the release of context information on a per-request basis, thereby realizing some kind of lazy rule evaluation. This is necessary given the context-dependent nature of the user's preferences: As explained in Section IV-B, the release of a certain context representation might well depend on the requester's current context. Hence, in such cases, the requester has to communicate his own context to the requested user's privacy manager, which will then decide on whether or not to release the requested information.

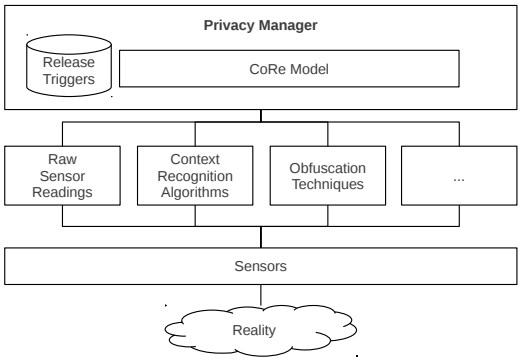

Fig. 5. The client-side components of the ALPACA framework: The *Privacy Manager* acts as an exclusive interface to access a user's context information for both local, peer-to-peer and third-party applications.

In accordance to the definitions of the different privacy layers, those representations belonging to the private layer must inalienably be managed on the mobile device itself. However, in order to enable the efficient and scalable implementation of applications based on, e.g., the current contexts of several users, there might also exist less trustworthy, yet possibly distributed instances of the privacy manager and the user's context model on the protected layer. If so, the privacy manager residing on the user's device will commit any updates in the set of currently released context representations to the external instances of the model. Naturally, not all representations currently released by the user's whitelist will be sent to the latters, but only those assigned to the respective application identified by the corresponding release triggers' *Audience* definitions. The ALPACA architecture can hence be considered flexible in the sense that it can be used both in a purely P2P-based fashion as well as in classic client/server-based applications without the need for a true TTP.

C. Communication flow in ALPACA

Having introduced the privacy layers and software components of ALPACA in the previous sections, we will now describe the basic communication flow in an example setup of our system. For the sake of clarity, however, we will only refer to the P2P-based case with one internal instance of the privacy manager here.

As illustrated in Figure 5, we assume a number of different sensors, algorithms for context recognition as well as obfuscation techniques to be available on the user's smartphone. These are responsible for capturing a user's real context and transforming it into instances of the *Representation* class for our context model, also providing the necessary meta information such as, e.g., *freshness* and *obfuscation level*. Every time one of the available sources of context information produces fresh data, the model will be updated. Additionally, the context-dependent blacklist defining which sensors and inference mechanisms to turn off under certain conditions will be re-checked. Now different audiences might request some of the user's context information from the privacy manager. Hence, it will first check which audiences the requester belongs to. In case the requesting entity is a local service that the user wants to have full access to her high-resolution context information, the privacy manager will return all matching

 ISBN: 978-1-61208-306-3

representations from the private layer. Otherwise, the privacy manager will take the requester's context information contained in the request in order to re-interpret the user's whitelist based on these information, thereby possibly firing some of the release triggers. Eventually, given that no inconsisteny is detected, the privacy manager will return the appropriate representation to the requester. In order for our system to be able to answer every incoming request even in case there is no such representation, some kind of highly obfuscated standard representation will be returned (cf. Section II).

VI. Discussion

We will now dicuss the pros and cons of our privacy-centric approach for modeling and managing a user's context. Strictly following a user-centric point of view, all decisions in the design process have been taken in order to offer a maximum level of control and privacy concerning the release of a user's context information. As a result, our approach is able to fulfill the different privacy requirements presented in Section II, which were collected from existing works on privacy in context-aware applications: In order to be as generic as possible, we decoupled our model from the generation of context information, thereby making it independent from existing sensors, as well as reasoning mechanisms and obfuscation techniques. The latters are able to realize the concepts of variable granularity, plausible deniability and the like. In practical implementations, new sensors and inference algorithms can simply register themselves at the privacy manager, which will inform the user about the additional context sources and obfuscation levels being available for the definition of her privacy preferences. Also, our framework will always give an unambiguous answer – according to the user's context-dependent preferences – to any incoming request, thanks to the automatic detection and remedy of inconsistent privacy preferences based on the underlying ontology's internal reasoning capabilities and modeled precedence rules, respectively. However, for a secure implementation our system must be run on some kind of trusted computing platform able to prevent the privacy manager from being circumvented by malware.

The ALPACA architecture exhibits considerable flexibility with regard to the fact that it can be used both exclusively on a user's mobile device as well as in P2P and client/server applications. Making usage of distributed instances of the context model and the privacy manager running on a server in the Internet can also help against network layer attacks trying to locate or identify a user by her current network location. However, there is a natural trade-off between a user's privacy and the amount of context information stored on distributed servers likely to be under someone else's administration. Finally, although having designed our trigger mechanism to be intuitive and user-friendly, it is not clear whether a majority of users will be likely to adopt such a restrictive whitelist-based system, facing peer pressure and their own reluctance towards manually configuring release triggers.

VII. Conclusion

This paper presented a novel approach for modeling a user's context-information in a privacy-centric way. We introduced our ontology-based CoRe model, which can be used for managing multiple, semantically different representations of the same class of context information fit for differently trustworthy groups of context requesters. In order to enable context-dependent definitions of a user's privacy preferences, a flexible whitelist-based trigger mechanism has been created. Eventually, we presented the design of our ALPACA system architecture and discussed our approach.

At the time of writing, we are currently working on a prototype implementation of our system allowing us to conduct a user study for evaluating the feasibility of our trigger mechanism. As for our future work, we aim at finding additional mechanisms capable of ensuring context consistency over several consecutive requests by a single entity. Furthermore, we want to analyze how to also protect the requesting entities' contexts, as well as how to handle possible deadlock situations resulting from mutually exclusive privacy preferences. Finally, we will try to find new obfuscation techniques for different types of context information and integrate them into ALPACA.

References

[1] F. Dorfmeister, M. Maier, M. Schönfeld, and S. A. W. Verclas, "Smartbees: Enabling smart business environments based on location information and sensor networks," in 9. GI/KuVS-Fachgespräch "Ortsbezogene Anwendungen und Dienste", 2012, pp. 23–37.

[2] G. D. Abowd et al., "Towards a better understanding of context and context-awareness," in Handheld and ubiquitous computing. Springer, 1999, pp. 304–307.

[3] M. Ackerman, T. Darrell, and D. J. Weitzner, "Privacy in context," Human–Computer Interaction, vol. 16, no. 2-4, pp. 167–176, 2001.

[4] W. Bokhove, B. Hulsebosch, B. Van Schoonhoven, M. Sappelli, and K. Wouters, "User privacy in applications for well-being and well-working," in AMBIENT 2012, The Second International Conference on Ambient Computing, Applications, Services and Technologies, 2012, pp. 53–59.

[5] M. Blount et al., "Privacy engine for context-aware enterprise application services," in Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on, vol. 2. IEEE, 2008, pp. 94–100.

[6] R. Wishart, K. Henricksen, and J. Indulska, "Context privacy and obfuscation supported by dynamic context source discovery and processing in a context management system," in Ubiquitous Intelligence and Computing. Springer, 2007, pp. 929–940.

[7] V. Sacramento, M. Endler, and F. N. Nascimento, "A privacy service for context-aware mobile computing," in Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on. IEEE, 2005, pp. 182–193.

[8] H. Chen, T. Finin, and A. Joshi, "An intelligent broker for context-aware systems," in Adjunct proceedings of Ubicomp, vol. 3, 2003, pp. 183–184.

[9] L. Kagal, T. Finin, and A. Joshi, "A policy language for a pervasive computing environment," in Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on. IEEE, 2003, pp. 63–74.

[10] W. Apolinarski, M. Handte, D. Le Phuoc, and P. J. Marrón, "A peer-based approach to privacy-preserving context management," in Proceedings of the 7th international and interdisciplinary conference on Modeling and using context, ser. CONTEXT'11, 2011, pp. 18–25.

[11] C. Bettini et al., "A survey of context modelling and reasoning techniques," Pervasive and Mobile Computing, vol. 6, no. 2, pp. 161–180, 2010.

[12] R. Reichle et al., "A comprehensive context modeling framework for pervasive computing systems," in Proceedings of the 8th IFIP WG 6.1 international conference on Distributed applications and interoperable systems, ser. DAIS'08, 2008, pp. 281–295.

[13] T. Strang, C. Linnhoff-Popien, and K. Frank, "Cool: A context ontology language to enable contextual interoperability," in Distributed applications and interoperable systems. Springer, 2003, pp. 236–247.

 ISBN: 978-1-61208-306-3