Distributed, Uncertainty-Aware Access Control for Pervasive Computing

Urs Hengartner and Ge Zhong
David R. Cheriton School of Computer Science
University of Waterloo
{uhengart, gzhong}@cs.uwaterloo.ca

Abstract

Access control to sensitive resources in pervasive computing needs to take uncertainty into account. Previous research has developed uncertainty-aware access-control models for environments that are managed by a centralized administrator. We demonstrate that environments managed in a distributed way require a more powerful model. Furthermore, we point out additional challenges that need to be considered when deploying uncertainty-aware access control, namely, identifying and authenticating both people and their intended actions, associating uncertainty with time, providing monotonicity, and defending against Sybil attacks. We present an access-control model that addresses these challenges and discuss a sample implementation.

1. Introduction

Access control to sensitive resources in pervasive computing needs to take uncertainty into account. For example, in biometrics-based access control, a person's observed trait, such as her face, is rarely completely identical to a trait stored by the access-control mechanism. Instead, the mechanism considers authentication a success if uncertainty about the two traits being from different people is small. In another example, a service that locates people might be uncertain about a person's current location. Here, the service should be able to include this uncertainty in a statement about the person's location, instead of issuing no statement at all. A location-based service exploiting this statement might be willing to accept a certain amount of uncertainty. Finally, a person could trust different services to a different degree to achieve a task, such as locating people. The person could have high confidence in a service that locates people's cellphones, but less confidence in a service that locates people's badges.

Traditionally, uncertainty-aware access control for pervasive computing has been studied in the context of centralized environments, where an administrator decides about the services that are used for authenticating and locating people and the amount of trust in the services' ability to fulfill these tasks [5, 9, 15]. The existing work does not support distributed environments, where individuals make these decisions. We focus on these environments in this paper. We also study some additional challenges that have not been addressed, namely, identifying and authenticating both people and their intended actions, associating uncertainty with time, providing monotonicity [3], and defending against Sybil attacks [6].

Our main contribution is to address these challenges by defining a formal, uncertainty-aware access-control model for pervasive computing environments that are managed in a distributed way. The model also supports the combination of statements made by different services to achieve consensus and ensures that corrupted services have only limited impact. In another contribution, we provide a sample implementation of the proposed access-control model.

We first discuss some challenges raised by uncertainty-aware access control in pervasive computing (Section 2). Next, we present a formal model that addresses these challenges (Section 3). Finally, we discuss the implementation of this formal model (Section 4).

2. Challenges

In this section, we discuss some challenges that arise when incorporating uncertainty into access control in pervasive computing and that have not been addresses in earlier work.

2.1. Distributed Environments

Often, there is some uncertainty about whether a service is able to fulfill its assigned tasks. We use the term *trust* to denote this kind of uncertainty. For example, a service that locates people by locating their badges might have only limited trust since people can leave their badges in their offices or exchange them. Different services will have different degrees of trust. For example, a service that locates people's cellphones will typically be more trusted than a service that locates people's badges since people are less likely to leave their cellphones in their offices or to exchange them. Trust in a service could also depend on the identity of the entity that provides the service.

Many pervasive computing environments (e.g., [4, 8, 14, 15]), sometimes implicitly, assume that there is a centralized administrator who configures the access-control mechanism and who decides about the amount of trust in individual services. Corresponding formal models can express the level of trust in a service, but not who decides about this level. The assumption is that the same level is appropriate for all people. This assumption holds for centralized, closed environments. For example, a company exploits a limited set of location services and assigns each a service a level of trust. In more open, distributed environments, like homes, universities, or shopping malls, these environment-wide decisions, covering all people, services, and devices in the same way, are not flexible enough. For example, in a university environment, students should be able to decide what services to use for gathering and providing location information and the level of trust in each service.

In summary, our access-control model should be flexible enough to also support environments where trust in services is defined in a distributed way.

2.2. Identification and Authentication

There are many scenarios in pervasive computing where considering only a person's context, such as her location, is not sufficient for access control and where user identification and authentication are required. For example, a pizza-delivery person should not have access to a projector in a meeting room just because she is in the meeting room. User identification and authentication should be seamless. For instance, people could carry personal devices with them that identify and authenticate their owner. Alternatively, a pervasive computing environment could identify and authenticate a person based on, for example, her face or her voice. Some of these technologies have uncertainty associated with them, which the access-control mechanism needs to take into account. For example, in the case of biometrics, the mechanism has to consider that a person's observed trait rarely matches a stored trait completely.

Identifying and authenticating a person is not sufficient, the access-control mechanism also needs to identify the action that the person wants to take. For example, when Alice turns on a projector, the access-control mechanism, on behalf of the projector, asks a videocamera (and maybe other sensors) observing the projector for the identity of the person who wants to use the projector. The camera notifies the mechanism of Alice's identity and of its degree of uncertainty about this identity and about the fact that Alice actually wants to use the projector (and not a different device that is nearby). Next, the mechanism evaluates the access policy of the projector and decides whether Alice should be granted access. Many pervasive computing projects (e.g., [4, 8, 14, 15]) support access policies, but do not formally express how an access-control mechanism derives that a particular person intends to take a particular action.

In summary, our access-control model should formally express uncertainty-aware identification and authentication of users and their intended actions.

2.3. Time

An important factor for uncertainty is time. Uncertainty tends to change over time, so when adding uncertainty to a statement, we have to define explicitly when this statement holds. There are multiple ways to achieve this goal. For example, a person's trust in a service changes over time, but probably only infrequently. Therefore, in our formal model, we keep the amount of trust in a service by a person constant, but we limit the lifetime of the statement expressing this trust. For services that locate or authenticate people, a service's uncertainty can change much faster, which can become a problem for context-sensitive services. Assume that an access-control mechanism makes a decision based on a person's location. Namely, the mechanism queries a location service for the person's location and subsequently makes a positive access decision. However, between issuing the query and making the decision, the person could have moved, so access might have been granted erroneously. There are ways to avoid this problem, though they are difficult to implement and not really necessary for pervasive computing [10] (e.g., people move only at a limited speed). An alternative is to have a service that issues a statement about a person's context also express the service's (anticipated) change in uncertainty about the context. For example, a location service states at what time it makes a statement about a person's location, its current uncertainty about this information, and how it expects this uncertainty to change in the near future (till the statement expires). For example, "At 8:00pm, the location service's uncertainty about Bob being in his office is at most 20%, this uncertainty increases linearly by 10% every minute, and the statement expires at 8:05pm".

In summary, our access-control model should be able to associate uncertainty with time.

2.4 Monotonicity

A desirable property of an uncertainty-aware access-control model is monotonicity. Monotonicity ensures that, given a set of statements that collectively grant access, a superset of this set will not deny access [3]. In terms of uncertainty, monotonicity guarantees that combining statements can only decrease uncertainty. For example, assume that there are two location services; one of them believes that Bob is in his office, the other one believes that Bob is at home. If a formal model allowed combination of such statements to a statement that Bob is in the office building, the summary uncertainty would have to be higher than the uncertainty that Bob is in his office. Therefore, if Bob tried to get access to a resource under the constraint that he is in the office building and if he had to collect statements about his current location, he would likely not present the statement about him being at home to the access-control mechanism and the mechanism might erroneously grant access. Non-monotonicity could also cause problems if the mechanism itself collected statements. For example, it is not clear how a service that is expected to make a statement, but is currently unreachable, should be dealt with in a non-monotonic model. Should we ignore it? Should we assume that it makes a negative statement? On the other hand, if the model was monotonic and ensured that a statement from this service could only decrease the summary uncertainty, it would be safe to ignore this service.

In summary, our formal access-control model should provide monotonicity.

2.5 Sybil Attacks

A Sybil attack [6] exploits monotonicity, as introduced in Section 2.4, to combine multiple statements, issued by the *same* service, but each under a different identity, to derive a summary statement with sufficiently low uncertainty for somebody to be granted access. For example, a location service issues a statement saying that Bob is in his office with at most 20% uncertainty. Bob could retrieve multiple such statements from the service, each issued under a different identity, till the summary uncertainty of the statements is below a threshold, and present them to the access-control mechanism. If the mechanism did not realize that the statements were issued by the same service, it would erroneously grant access to Bob.

In summary, our formal access-control model should defend against Sybil attacks.

3. Access-Control Model

In this section, we discuss an access-control model that addresses the challenges listed in the previous section. Our model is based on existing work [2], to which we added support for uncertainty. The existing model has the advantages that it is targeted at distributed environments (i.e., it formalizes who makes a statement), can be given a semantics in higher-order logic (i.e., the model is sound), and is similar to Lampson et al.'s well-known speaks-for logic [13].

3.1. Review of Existing Model

In the model, there is a $\mathbf{says}()$ statement, which identifies the entity making a statement. For example, a user, A, granting another user, B, access to a resource, U, is modelled as "A $\mathbf{says}$ $(\mathbf{delegate}(A, B, U))$". User B trying to access U is expressed as "B $\mathbf{says}$ $(\mathbf{goal}(U, N))$". (Nonce N avoids replay attacks.) Assuming that the access-control mechanism knows that A manages access rights for U, the mechanism tries to derive "A $\mathbf{says}$ $(\mathbf{goal}(U, N))$". Theorem DELEGATE-E in Table 1 supports this conclusion.

The model allows a statement to have a lifetime, which we exploit for tying uncertainty to time, as explained in Section 2.3. Here, we leave lifetime away for readability reasons. Entities are identified with their public key and issue statements in the form of digital certificates [1]. There is no need for a Public Key Infrastructure (PKI).

3.2. Uncertainty Awareness

Our access-control model should allow entities that issue a statement to express their uncertainty about the statement. A design choice is whether it should be possible to associate any statement with uncertainty or only a limited set. While there are logics that take the former approach [7], we choose the latter one. For some statements, the semantics of having a statement associated with uncertainty are ambiguous, which could lead to users being confused (and making wrong decisions). For instance, Alice either grants an access right to Bob or she does not, but there is no need for giving her the option to associate uncertainty with the access right itself. However, the access right can include a constraint that has uncertainty associated with it. For example, Alice grants Bob access under the constraint that he is in his office with uncertainty lower than 20%.

To address the challenges mentioned in Section 2, we add several statements and theorems to the access-control model. The statement "A $\mathbf{says}$ $(\mathbf{delegateIf}(I, V, P, A, B, U))$" denotes that A grants B access to U if the value of I is in set V with uncertainty lower than P. (We discuss possible implementations of uncertainty in Section 4.1.) For example, I could be "Alice's location" and V could contain her office and a meeting room. Theorem IF-E in Table 1 shows the application of the statement. The statement is combined with a statement saying that A thinks that there is consensus about I being in set W, expressed as "A $\mathbf{says}$ $(\mathbf{consensusIn}(I, W, Q))$", to get a regular $\mathbf{delegate}()$ statement, under the conditions that the uncertainty about the consensus statement, Q, is lower than P and that $W \subset V$ holds. We discuss the $\preceq$ condition in Section 4.1.

User A should be able to choose the services that can acknowledge satisfaction of a constraint in an access right of hers, such as the one shown above. A service is an entity that has a public key; it can be a sensor (e.g., a videocamera), a company (e.g., a cellphone company providing its clients' location), or another user. The statement "A **says** ($\mathbf{delegateIn}(A, S, I, V, P)$)" has A delegate the authority to decide whether I is in V to S, where A's trust in S is P. Theorem DELEGATE-IN-E in Table 1 shows the application of this statement. The statement "S **says** ($\mathbf{in}(I, W, S, Q)$)" denotes that S believes that the current value of I is in set W with uncertainty Q. We include S in the $\mathbf{in}()$ part of the statement to remember its origin; we will need this origin later for the consensus operation. The uncertainty about the derived statement, R, in Theorem DELEGATE-IN-E depends on P and Q. (See Section 4.1 for a discussion of the $\otimes$ condition). When issuing a $\mathbf{delegateIn}()$ statement to a service, A identifies which of maybe multiple identities of this service is authorized to issue $\mathbf{in}()$ statements by picking a particular public key of the service. This way, all the other public keys that the service might have are ignored by the access-control mechanism and Sybil attacks are avoided.

Multiple services can be authorized to issue $\mathbf{in}()$ statements. Theorem CONSENSUS-IN-I in Table 1 defines their consensus. (See Section 4.1 for a discussion of the $\oplus$ condition.) There can be more than two $\mathbf{in}()$ statements. The theorem might look counter-intuitive, since it supports consensus even if $V \neq W$. However, if A believes that $I \in V$, then A's uncertainty about I being in a set that includes V is at most as high. The theorem implicitly assumes that this superset is $V \cup W$ and computes the consensus of I being in this superset. This can be useful in scenarios where somebody is granted access if he is in a building, where two location services state that he is in different rooms of this building, and where the uncertainty about each individual statement is too high for the person to be granted access, whereas the uncertainty about the consensus statement is sufficiently low. Uncertainty R is guaranteed to be lower than P and Q (see Section 4.1), which gives monotonicity.

Statement "S **says** ($\mathbf{indirectGoal}(B, U, Q, N, S)$)" expresses that service S authenticates user B and his intention to access resource U with uncertainty Q. (N is a nonce.) Note that the statement authenticates both a user and the action that he wants to take. User A accepts this statement only if S is authorized to make such a statement, formalized in a $\mathbf{delegateAuth}()$ statement, and if B has access to U, as shown in Theorem DELEG-AUTH-E in Table 1.

A user and her action can be authenticated by multiple services, therefore, we need consensus, as shown by Theorem CONSENSUS-GOAL-I in Table 1. The theorem allows the derivation of "A **says** ($\mathbf{consensusGoal}(U, R, N)$)".

If A allows a service to authenticate users who want to access resource U, A should also define the maximum uncertainty that will still allow these users to be granted access, formally "A **says** ($\mathbf{confidence}(U, Q)$)". Theorem CONFIDENCE-E in Table 1 exploits this statement.

We also need a statement that expresses change in uncertainty in the near future, as discussed in Section 2.3. "S **says** ($\mathbf{linearIn}(I, V, S, T, P, Q, D)$)" denotes that S believes $I \in V$ at time T with uncertainty P and that this uncertainty increases linearly by Q in interval D. Theorem TIME-E in Table 1 shows how the current uncertainty is derived. A similar statement models change in uncertainty in terms of authenticating users and their actions.

3.3. Example

Let us discuss the application of our model in an example. Owner A of projector U grants user B access if he is in room 123, formally "A **says** ($\mathbf{delegateIf}(B.location, \{room \quad 123\}, P, A, B, U)$)". Owner A also issues a $\mathbf{delegateAuth}()$ statement that authorizes a camera next to the projector to authenticate users and their actions and a $\mathbf{delegateIn}()$ statement that authorizes a location service, S, to locate people. When B wants to access the projector, the location service issues an $\mathbf{in}()$ statement about B's location. Using the DELEGATE-IN-E theorem, the access-control mechanism ensures that the service has been authorized accordingly and applies the CONSENSUS-IN-I theorem to get the consensus statement "A **says**($\mathbf{consensusIn}(B, \{room \quad 123\}, R)$)". This statement is then combined with the access right issued by A, using theorem IF-E, to conclude "A **says**($\mathbf{delegate}(A, B, U)$)". The camera, C, authenticates the user and his action by issuing "C **says**($\mathbf{indirectGoal}(B, U, Q, N, C)$)". The access-control mechanism uses this statement, the camera's authorization from A, and the just derived $\mathbf{delegate}()$ statement to derive an $\mathbf{indirectGoal}()$ statement made by A, based on the DELEG-AUTH-E theorem, and uses consensus to get a $\mathbf{consensusGoal}()$ statement. Finally, if the uncertainty about this statement is lower than the uncertainty given by A in her $\mathbf{confidence}()$ statement, the access-control mechanism concludes "A **says**($\mathbf{goal}(U, N)$)" and grants access.

4. Implementation

In this section, we discuss an implementation of the access-control model presented in the previous section.

4.1. Subjective Logic

There are multiple options to express uncertainty in our access-control model. We can associate a statement with

Table 1. Theorems.

DELEGATE-E	$\dfrac{A\ \textbf{says}\ (\textbf{delegate}(A,B,U)) \quad B\ \textbf{says}\ (\textbf{goal}(U,N))}{A\ \textbf{says}\ (\textbf{goal}(U,N))}$
IF-E	$\dfrac{A\ \textbf{says}\ (\textbf{delegateIf}(I,V,P,A,B,U)) \quad A\ \textbf{says}\ (\textbf{consensusIn}(I,W,Q))}{A\ \textbf{says}\ (\textbf{delegate}(A,B,U))}\ W \subset V,\ Q \preceq P$
DELEGATE-IN-E	$\dfrac{A\ \textbf{says}\ (\textbf{delegateIn}(A,S,I,V,P)) \quad S\ \textbf{says}\ (\textbf{in}(I,W,S,Q))}{A\ \textbf{says}\ (\textbf{in}(I,W,S,R))}\ W \subset V,\ R = P \otimes Q$
CONSENSUS-IN-I	$\dfrac{A\ \textbf{says}\ (\textbf{in}(I,V,S,P)) \quad A\ \textbf{says}\ (\textbf{in}(I,W,T,Q))}{A\ \textbf{says}\ (\textbf{consensusIn}(I,X,R))}\ X = V \cup W,\ R = P \oplus Q,\ S \neq T$
DELEG-AUTH-E	$\dfrac{S\ \textbf{says}\ (\textbf{indirectGoal}(B,U,Q,N,S)) \quad A\ \textbf{says}\ (\textbf{delegateAuth}(A,S,B,P)) \quad A\ \textbf{says}\ (\textbf{delegate}(A,B,U))}{A\ \textbf{says}\ (\textbf{indirectGoal}(B,U,R,N,S))}\ R = P \otimes Q$
CONSENSUS-GOAL-I	$\dfrac{A\ \textbf{says}\ (\textbf{indirectGoal}(B,U,P,N,S)) \quad A\ \textbf{says}\ (\textbf{indirectGoal}(B,U,Q,N,T))}{A\ \textbf{says}\ (\textbf{consensusGoal}(U,R,N))}\ R = P \oplus Q,\ S \neq T$
CONFIDENCE-E	$\dfrac{A\ \textbf{says}\ (\textbf{consensusGoal}(U,P,N)) \quad A\ \textbf{says}\ (\textbf{confidence}(U,Q))}{A\ \textbf{says}\ (\textbf{goal}(U,N))}\ P \preceq Q$
TIME-E	$\dfrac{S\ \textbf{says}\ (\textbf{linearIn}(I,V,S,T,P,Q,D))}{S\ \textbf{says}\ (\textbf{in}(I,V,S,R))}\ R = linearIncrease(T,P,Q,D,currentTime)$

a probability, meaning that the entity making the statement believes in the correctness of the statement with at least the given probability. However, the one-dimensional probability, 0 to 1, would reduce the reliability of an access-control model that guarantees monotonicity. As explained in Section 2.4, monotonicity is a desirable feature. It would ensure that the probability of a summary statement is higher than the probabilities of the statements underlying the summary statement. For example, Ranganathan et al.'s framework [15] has this property. Unfortunately, this property implies that a false statement issued by a single corrupted sensor, which assigns high probability to the statement, will lead to a false summary statement that also has high probability, even if all uncompromised sensors give it low probability. This property contradicts the purpose of deploying multiple sensors to increase reliability.

To overcome this drawback, we use Subjective Logic [12] in our implementation. Subjective Logic distinguishes between the belief and disbelief in a statement (i.e., believing that the statement is true/false) and the ignorance about the statement (i.e., not knowing whether the statement is true or false) and associates the statement with a belief, disbelief and ignorance triplet of the form $[b, d, i]$, where $b + d + i = 1$ and $[b, d, i] \in [0, 1]^3$. Traditional probability, as discussed above, combines disbelief and ignorance in a single value. In the context of Subjective Logic, monotonicity implies that the ignorance about a summary statement is lower or equal than the ignorance about any of the underlying statements. However, the belief in the summary statement is not necessarily higher, which reduces the impact of corrupted sensors.

We use two existing operations of Subjective Logic, *recommendation* and *consensus*, for implementing the '$\otimes$' and '$\oplus$' conditions of our model, respectively. Assume two entities A and B where A has an opinion about B, and B has an opinion about a proposition p. A recommendation of these two opinions consists of combining A's opinion about B with B's opinion about p in order for A to get an opinion about p. The consensus rule for combining independent opinions consists of combining two or more independent opinions about the same proposition into a single opinion. The definition of the consensus operation [12] guarantees monotonicity.

We implement the $\preceq$ condition of our model as a slight modification of the ordering operation provided by Subjective Logic and define $P \preceq Q$ if and only if Subjective Logic orders P higher than Q and if $P.i \leq Q.i$. For the first condition, Subjective Logic provides ordering among triplets by selecting the one with the greatest $(b + i)/(b + d + 2i)$ ratio [12]. Note that this ordering does not necessarily give preference to triplets with large belief, instead it also takes ignorance (which could be transformed into greater beliefs) into account. The second condition in the definition of $P \preceq Q$ is required for monotonicity; if the ignorance of a consensus statement is not sufficiently low for being granted access, the ignorance of the underlying statements will not be sufficiently low, either. This property encourages people to provide more statements in order to get access. The more statements, the smaller the ignorance.

4.2. Access-Control Framework

We have implemented our access-control model in Prolog and are deploying this model in an existing access-control framework for pervasive computing [10]. The model is oblivious to how uncertainty is implemented. In our current implementation, we added several Prolog terms that implement uncertainty based on Subjective Logic. For example, we added a term "Order(P, Q)" that is true only when $P \preceq Q$, as defined in Section 4.1.

5. Related Work

Several papers [5, 9, 15] have introduced the idea of making services include uncertainty when authenticating a person. The amount of uncertainty can be determined by the service or by the administrator of an environment, maybe in combination. Our contribution is to recognize that authenticating a person is not sufficient, we also need to authenticate her intended action. In addition, the related work addresses a centralized system, where the administrator decides on an environment-wide base about the uncertainty of statements made by a service, whereas our model is more flexible and allows individuals to make these decisions.

In terms of time and uncertainty, Ganger [9] recognizes that authentication uncertainty increases over time, but does not formalize this observation in an access-control model. Our model allows a service to formally express how its uncertainty changes over time.

Indulska et al. [11] examine how to combine (potentially conflicting) location information from multiple sensors. Their algorithm does not guarantee monotonicity, as opposed to ours, which can result in problems, as explained in Section 2.4. Another benefit of our approach is that this combination is part of the access-control model, thus it avoids ambiguities.

Our access-control model requires that people express their uncertainty about statements made by a service and that services express their uncertainty when making a statement, but we do not discuss how this uncertainty is computed. Covington et al. [5] suggest to compute uncertainty based on behavior in the past and based on the closeness of observed information to stored information. Indulska et al. [11] also take past behavior into account. Both Covington et al. and Indulska et al. compute uncertainty for a particular moment in time and do not consider how changes in the (near) future can affect the access-control mechanism.

6. Conclusions and Future Work

We have presented an access-control model that allows different people to have different degrees of trust in services, that authenticates both people and their actions, that associates uncertainty with time, that provides monotonicity, and that defends against Sybil attacks.

We are currently deploying and evaluating our proposed model in an existing pervasive computing environment. We also need to prove soundness of our extensions to the existing access-control model. Finally, we have to validate whether the decisions mady by our uncertainty-aware access-control model correspond to users' expectations.

Acknowledgments

We thank the anonymous reviewers for their comments. This work is supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1] L. Bauer. *Access Control for the Web via Proof-Carrying Authorization*. PhD thesis, Princeton University, November 2003.

[2] L. Bauer, M. A. Schneider, and E. W. Felten. A General and Flexible Access-Control System for the Web. In *Proceedings of 11th Usenix Security Symposium*, pages 93–108, August 2002.

[3] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance Checking in the PolicyMaker Trust Management System. In *Proceedings of 2nd Conference on Financial Cryptography (FC '98)*, pages 251–265, February 1998.

[4] H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *Proceedings of First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004)*, August 2004.

[5] M. J. Covington, M. Ahamad, I. Essa, and H. Venkateswaran. Parameterized Authentication. In *Proceedings of 9th European Symposium on Research in Computer Security (ESORICS 2004)*, pages 276–292, September 2004.

[6] J. R. Douceur. The Sybil Attack. In *Proceedings of First International Workshop on Peer-to-Peer Systems*, March 2002.

[7] R. Fagin and J. Y. Halpern. Reasoning about Knowledge and Probability. *Journal of the ACM*, 41(2):340–367, 1994.

[8] F. Gandon and N. Sadeh. A Semantic eWallet to Reconcile Privacy and Context Awareness. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, October 2003.

[9] G. R. Ganger. Authentication Confidences. In *Proceedings of 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, page 169, Month 2001.

[10] U. Hengartner and P. Steenkiste. Avoiding Privacy Violations Caused by Context-Sensitive Services. *Elsevier Journal of Pervasive and Mobile Computing (PMC), PerCom 2006 special issue*, 2(4):427–452, November 2006.

[11] J. Indulska, T. McFadden, M. Kind, and K. Henricksen. Scalable Location Management for Context-Aware Systems. In *Proceedings of 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2003)*, pages 224–235, November 2003.

[12] A. Jøsang. Artificial Reasoning with Subjective Logic. In *Proceedings of Second Australian Workshop on Commonsense Reasoning*, December 1997.

[13] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in Distributed Systems: Theory and Practice. *ACM Transactions on Computer Systems*, 10(4):263–310, November 1992.

[14] K. Minami and D. Kotz. Secure Context-sensitive Authorization. In *Proceedings of 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, pages 257–268, March 2005.

[15] A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell. Reasoning about Uncertain Contexts in Pervasive Computing Environments. *IEEE Pervasive Computing*, 3(2):62–70, April-June 2004.