Context-Based Reasoning on Privacy in Internet of Things

Nadin Kökciyan
Department of Computer Engineering
Bogazici University, Istanbul, Turkey
nadin.kokciyan@boun.edu.tr

Pınar Yolum
Department of Computer Engineering
Bogazici University, Istanbul, Turkey
pinar.yolum@boun.edu.tr

Abstract

More and more, devices around us are being connected to each other in the realm of Internet of Things (IoT). Their communication and especially collaboration promises useful services to be provided to end users. However, the same communication channels pose important privacy concerns to be raised. It is not clear which information will be shared with whom, for which intents, under which conditions. Existing approaches to privacy advocate policies to be in place to regulate privacy. However, the scale and heterogeneity of the IoT entities make it infeasible to maintain policies among each and every entity in the system. Conversely, it is best if each entity can reason on the privacy using norms and context autonomously. Accordingly, this paper proposes an approach where each entity finds out which contexts it is in based on information it gathers from other entities in the system. The proposed approach uses argumentation to enable IoT entities to reason about their context and decide to reveal information based on it. We demonstrate the applicability of the approach over an IoT scenario.

1 Introduction

Preserving privacy online is being studied extensively. Internet of Things (IoT) is emerging as an area where privacy is crucial but difficult to preserve. Important characteristics of IoT that set it apart from other computational systems in terms of privacy are the following:

- Dynamic: In Web-based systems, users enter the system through well-defined means, such as logging in. With this, it is possible to establish a common grounds, e.g., through a privacy policy. However, IoT vision is based on entities being everywhere, without a central point of entry. A human that enters a room will not be aware of all the IoT entities, their capabilities, let alone their policies.
- Large: Privacy policies are generally conducted between a user and an entity. Even when creating a policy is possible, the scale of IoT makes it infeasible to realize and maintain. Each user would have to agree to a privacy policy from all entities. Every time a user walks into a new environment or a new device is being installed in an existing setting, the privacy policy has to be talked about [Weber, 2010].
- Heterogeneous: Many of the entities in an IoT will be from different vendors, installed for different reasons, have varying capabilities and possibly managed by different principals. Hence, when an IoT entity reports on a piece of information, the quality and accuracy of that information may not be sufficient. For example, a sensor can sense the existence of a human, whereas a 3D camera can report on the identity as well as the recent actions of the human. This requires information from different entities to be treated and trusted differently [Sicari *et al.*, 2015].

In order to preserve privacy under these properties, it is best if each entity can reason on the privacy of the information it is collecting, processing, or sharing, based on the current context. Nissenbaum's theory on *contextual integrity* [Nissenbaum, 2004] achieves this by categorizing information as sensitive or non-sensitive regarding the role and the social context of a user. In each context, some types of information are appropriate to share. For example, a medical expert should not ask the salary of a patient in the medical context. Hence, it is important to respect the *norms of appropriateness*, which dictate what information to reveal and disseminate in a particular context. We follow this intuition in this work with two main differences: (i) We enable an agent to be in more than one context at the same time with varying degrees of belief. (ii) We define context using relations rather than roles as IoT enables entities to take different parts in different organizations.

We represent each IoT entity as a software agent that can perceive, reason, act and communicate with other agents. Each agent in the system calculates the appropriateness of sharing information with others using *argumentation* [Fox *et al.*, 2007]. Argumentation enables an agent to reason on its existing rules, facts and assumptions. With each new information that is received from another agent in the system, the agent can come to a new conclusion as to which context is active and based on that decides to share or not share the information. The collected information can be wrong or contra-

dictory with each other. To make inference based on others' information, a trust model needs to be in place so that the agent can factor in the trustworthiness of the agents.

The rest of this paper is organized as follows: Section 2 develops our computational model and Section 3 explains the agent's autonomous decision making process. Section 4 describes our realization of the model using argumentation. Section 5 walks through the execution on the running scenarios and gives theoretical results. Section 6 discusses our work in relation to the literature.

2 Approach

Most of the times, people are not aware that they are surrounded with smart entities, which can collect their information. For example, there are surveillance cameras in many places where people walk by (or work in or live in). It is not realistic to assume that each entity will be aware of all the privacy expectations of each user separately and act accordingly. Instead, each entity needs to make privacy decisions *autonomously* based on information it can access. This will result in entities to make privacy decisions. The following example illustrates this:

Example 1. A surveillance camera at Alice's work place records a video 24/7. The footage is not shared with others except when there is an emergency. Alice's boss Bob would like to access Alice's footage taken on November 30, with the claim that Alice might be in trouble and that her recent footage might help solve the situation. The camera needs to make a decision autonomously. Should the camera share the footage?

On one hand, the camera has a policy not to share footage; on the other hand, if the footage would help Alice, it might be in Alice's interest that it was shared.

We represent each IoT entity as a software agent that observes its physical environment, communicates with other agents when necessary, and reasons on the information it gathers. We primarily focus on how each agent decides to share information autonomously taking into account the privacy aspects as necessary. To do this, we capture the idea of context and the norms associated with them. The information received from others enables an entity to reason on the context and then apply the norms associated with that context to decide whether to share a piece of information.

2.1 Context

Context can be defined as the necessary information to identify the situation of an entity correctly [Abowd *et al.*, 1999]. Various pieces of information such as time (e.g., Nov30), location (e.g., *atWork*), properties of the environment (e.g., *fire*) and such can be used to derive context. Possible contexts in the running example are *:work* context, the emergency context (*:em*) and the *:party* context. By definition of context, it is subjective; e.g., two entities in the same location and time can be in different contexts.

More interestingly, an agent can be in multiple contexts at the same time [Criado and Such, 2015]. Following the example, assume that Alice and Bob are at a party at work, talking about a recent report they wrote. Alice is at work context because of her location and is at party context because of the ongoing party. Because of the nature of IoT systems, we need to allow for multiple contexts to be active at the same time, possibly at varying degrees of belief. We achieve this by associating degree of beliefs to possible contexts an agent can be in. The degree of belief (`dob`) value is associated with each piece of information that an agent has. The `dob` value is a real value between zero and one. A high `dob` value of an information shows a high confidence in that information. For example, Alice can be at work context with a degree of 0.6 and at party context with a degree of 0.9.

In contextual integrity (CI), each context is associated with a set of roles [Nissenbaum, 2004]. For example, in the work context, it is possible to define roles such as boss, employee and so on. In a single organization, it is convenient to come up with a set of contexts and define roles in these contexts since users can play their roles within the organization. However, in IoT as multiple contexts exist, capturing the relations instead of roles are more realistic [Fong, 2011].

In each context, we define a set of relationships (as opposed to roles in CI). In Example 1, the following relationships are defined. In *:work* context, *isBossOf* is a relationship between an employee and a boss (e.g., *isBossOf*(*:bob*, *:alice*)). In *:party* context, *isFriendOf* relationship is defined between people. In *:em* context, *homeSensorOf* is a relationship to connect the home sensor to its owner (e.g., *homeSensorOf*(*:home*, *:alice*)), and *isResponsibleFor* relates a police officer to a person (e.g., *isResponsibleFor*(*:jack*, *:alice*)).

2.2 Norms

Similar to CI, in this work, agents behave according to a set of norms that are defined in contexts. To reason about their environments, agents only use the norms that are appropriate in specific contexts. The dynamic nature of IoT environments would require a large number of privacy policies between entities and users, which would not be scalable. Hence, agents decide how to disseminate information that they collect [Solove, 2006]. We define context formally in Definition 1.

Definition 1 (Context)**.** *Context is a two tuple $\langle \mathcal{N}, \mathcal{R} \rangle$, where $\mathcal{N}$ is a set of norms and $\mathcal{R}$ is a set of binary relationships. Each norm can be associated with a* `dob` *value between zero and one.*

In Table 1, the example norms of the camera agent are shown as Prolog rules. Each norm is denoted as N_i, where i is the norm id. The body of the rule consists of a conjunction of predicates from the language, and the head of the rule is a single predicate. Negations are allowed both in the body and the head of a rule. Each norm can be associated with a `dob` value as well (e.g., N_1). We refer to each entity as *:entity-name* (e.g., *:cam*). *inContext*(A, C, T) represents an agent A that is in context C at time T. *atWork*(A, T), *onleave*(A, T), *atHome*(A,T), *missing*(A, T) show that agent A can be at work, on leave, at home or missing at time T, respectively. Agent-independent predicates omit agent A (e.g., *fire*(T) or *workday*(T) means there is a fire or T is a workday, respectively.). These terms can be nested in *info*(X) to capture the information flow. The decision predicate is *share*(A, F, T), which takes values based on the context.

Table 1: Example Norms of *:cam* as Prolog Rules

N_1 : *inContext*(A, C_d, T) ⇐ 0.5.
N_2 : ~*share*(A, F, T) ⇐ *inContext*(A, C_d, T).
N_3 : *inContext*(A, *:work*, T) ⇐ *info*(*atWork*(A, T)).
N_4 : *inContext*(A, *:em*, T) ⇐ *info*(*missing*(A, T)).
N_5 : *inContext*(A, *:em*, T) ⇐ *info*(*atWork*(A, T)), *info*(*fire*(T)).
N_6 : ~*share*(A, F, T) ⇐ *inContext*(A, *:work*, T).
N_7 : *share*(A, F, T) ⇐ *inContext*(A, *:em*, T).
N_8 : *share*(A, F, T) ⇐ *inContext*(A, *:party*, T).
N_9 : *info*(*missing*(A, T)) ⇐ *info*(~*onleave*(A, T)), *info*(*workday*(T)), *info*(~*atHome*(A, T)).

We formally define the default context in Definition 2. In Example 1, the default context is $\langle \cup_{i=1}^{5} N_i \cup N_9, \{\} \rangle$. All other contexts are associated with norms, which have a decision predicate in the head of the rule (e.g., N_6 - N_8). For example, *:em* $= \langle \{N_7\}, \{isResponsibleFor, homeSensorOf\} \rangle$.

Definition 2 (Default Context)**.** *$C_d = \langle \mathcal{N}, \mathcal{R} \rangle$ is the default, unique context, where $\{N_1, N_2\} \subseteq \mathcal{N}$.*

Agents can be in multiple contexts at a time. For this, the agent should infer active contexts regarding what it already knows. We capture this in Definition 3.

Definition 3 (Active Context)**.** *If inContext(:a, :c, :t) is inferred by agent a, then :c is an active context for agent :a at time :t.*

An agent has a predefined set of contexts and starts execution in the default context (C_d) with a `dob` value of 0.5 (N_1). The default context is always active because N_1 has an empty body; i.e., the head predicate is always inferred by the agent. When `dob` value of any other context is above 0, then that context also becomes active. When an agent is in context C_d at time T, then a not share predicate is inferred (N_2). N_3 states that if an agent A is at work at time T, then this agent is in *:work* context at time T. A same context can be inferred in different situations; e.g., N_4 and N_5 are two different rules that can be used to conclude that an agent is in an emergency context. N_4 states that if an agent A is missing at time T, then this agent is in *:em* state at T. The presence of a fire when an agent is at work could mean that the agent is in emergency state as well (N_5). The norms N_6-N_8 specify when the camera agent could share or not share an agent's footage depending on the agent's context. *:cam* can share an agent's footage in an emergency state (N_7) or for the purpose of organizing a party for an agent (N_8). However, *:cam* would not share an agent's footage when it is in work context. N_9 is a rule to understand when an agent would be missing. If an agent A is not on leave at time T, which is a workday, and A is not at home at time T; then A could be missing at time T.

3 Decision Making

In order to reach a privacy decision, an agent can use its own knowledge base (KB) that consists of facts and norms, associated with a `dob` value, but it can also decide to consult other agents to collect more information. An agent should decide what information to collect, from whom, and how to process the received information. Example 2 illustrates how *:cam* consults others for information. In our model, each agent is equipped with a belief base (BB), which includes information that is collected from others. In BB, each piece of information is associated with a `dob` value as well.

Example 2. The camera decides to consult Alice's home device and it turns out that Alice was not at home that day, either. Next, the camera consults the police department and finds out that there is a missing report for Alice. Is this enough evidence to decide Alice is in an emergency?

We propose Algorithm 1 to show the decision making steps of the agent. An agent makes a request to the other agent to reveal some of its data via an initial question q_0. The agent uses its own knowledge base KB and its belief base BB to make a privacy decision, which is returned as the result.

Algorithm 1: DECIDETOSHARE (q_0)

Input: q_0, the initial question
Output: d, the final decision (share or not share)
Data: KB, the knowledge base of the agent $q_0.a_j$
Data: v, the view of the agent $q_0.a_j$

```
1  BB ← applyNorms(KB);
2  C ← getActiveContexts(BB);
3  foreach c in C do
4      qSet ← getQuestionsToAsk(KB, BB);
5      aSet ← getAgentsToAsk(c.R, v.A);
6      rSet ← getResponses(qSet, aSet);
7      foreach r in rSet do
8          BB ← eval(r, v);
9      BB ← applyNorms(KB);
10     C ← getActiveContexts(BB);
11 d ← getStrongestDecision(KB, BB);
12 return d;
```

Information Collection: When external entities ask the agent to reveal some data of a user, the agent initializes its BB by applying its norms in KB (line 1). The user can be in multiple contexts (Definition 3), it finds out the active contexts of the user by using getActiveContexts (line 2). In Example 1, *:cam* is in two active contexts C_d and *:work*. *:cam* infers C_d via N_1 and *:work* context via N_3 because *info*(*atWork*(*:alice*, Nov30)) exists in its KB.

The agent tries to apply the norms that come from active contexts. For this, it generates a list of predicates with the missing facts. In the running example, to prove that Alice is an emergency state, *:cam* should know whether Alice is missing but this information does not exist either in KB or in BB of *:cam*. Therefore, *missing*(*:alice*, *:Nov30*) is added to the list of predicates to be asked for. A question set ($qSet$) is prepared by the agent to collect information from other agents

according to the list of predicates (line 4). We give a formal definition of a question in Definition 4. For example, $q_1 = \langle$ *:cam*, *:bob*, *missing*(*:alice*, Nov30) $\rangle$ is a question that *:cam* asks *:bob* to learn whether Alice is missing.

Definition 4 (Question). *$q_n = \langle a_i, a_j, X \rangle$ denotes a question where a_i is the question owner agent, a_j is the asked agent and X is the requested information, which is a complex term from the language. n is the question id.*

Asking Agents: Once the agent knows the context information of the user, it can find the set of relationships that are defined within each active context ($c.\mathcal{R}$). The auxiliary function getAgentsToAsk is used to instantiate these relationships with the user in question and the agents that it already knows ($v.\mathcal{A}$), and the result is a set of agents $aSet$ (line 5). The agent contacts agents in this set to ask questions in $qSet$, and collects responses in $rSet$ (line 6). Definition 5 specifies what a response is. For example, $r_1 = \langle$ *:bob*, *:cam*, *missing*(*:alice*, Nov30), 0.9, $q_1 \rangle$ is a message that is sent by *:bob* to *:cam* as a response to q_1. According to *:cam*, Alice is missing with a dob value of 0.9.

Definition 5 (Response). *$r_n = \langle a_i, a_j, X, b, q \rangle$ denotes a response that is sent from agent a_i to agent a_j. X is the content of the response, which is a complex term from the language. b is the* dob *value of a_i for information X, and takes a real value between zero and one. The response is given for question q (Definition 4). n is the response id.*

Information Processing: An agent decides how to process the received information. An easy approach would be to add the received information directly to the belief base. However, the agent can come up with wrong conclusions if it keeps noisy data in its BB. The best way to take into account such data is to consider the trustworthiness of information sources [Wang and Singh, 2007]. Each agent has access to its own view and *getTrustValue* function that assigns trust values to agents it knows. This work does not consider the semantics of the function; however, in principle we expect the agent to have evolved such a function over interactions with other agents in the system.

Definition 6 (View). *$\langle \mathcal{C}, \mathcal{A}, \mathcal{F}, getTrustValue \rangle$ is a view, where $\mathcal{C}$ is a set of contexts such that $C_d \in \mathcal{C}$, $\mathcal{A}$ is a set of agents that the agent knows, and $\mathcal{F}$ is a set of facts. There is a function $getTrustValue$: $\mathcal{A} \rightarrow$ [0, 1] that assigns a trust value to every agent in $\mathcal{A}$.*

The world view of the agent is subjective as captured in Definition 6. In Example 1, *:cam*'s view is as follows: $\langle \{C_d$, *:work*, *:em*, *:party*$\}$, $\{$*:bob*, *:home*, *:jack*$\}$, $\{$*atWork*(*:alice*, Nov30) 0.8, ∼*onleave*(*:alice*, Nov30), *workday*(Nov30)$\}$, $\{$ *:bob* (0.3), *:home* (0.9), *:jack* (0.9)$\}\rangle$.

$$getDob(r, v) = v.getTrustValue(r.from) * r.b \quad (1)$$

The agent evaluates the received responses in $rSet$ according to the following steps. First, the agent computes its own dob value for the received information regarding the trust value of the asked agent, and the dob value associated with it ($r.b$). In Equation 1, we show how the message receiver agent ($r.to$) computes its own dob value for the response r by using *getDob* function. $getTrustValue(a)$ is a function that returns the trust value for an agent a in its view v. Second, the agent ($r.to$) uses $eval(r, v)$ function to evaluate a response r as shown in Equation 2 (line 8). All the received information is added to the agent's belief base via $upBB(r.X, y)$, where y is a low or high dob value chosen by the agent initially. The agent checks whether the computed dob value ($getDob(r, v)$) is greater than its threshold (TH), which is also set initially. In this case, the agent adds the received information to its belief base with the high dob value (H). Otherwise, the low dob value (L) is associated with the received information. The agent updates its belief base using applyNorms function, and retrieves the set of active contexts (line 10).

$$eval(r, v) = \begin{cases} upBB(r.X, H) & getDob(r, v) > TH \\ upBB(r.X, L) & otherwise \end{cases} \quad (2)$$

We will keep on trying the previous steps as long as there are active contexts in BB (lines 3-10). Finally, the agent decides to share or not to share the content in question ($q_0.X$). Different decisions can be made regarding different active contexts. The agent wants to find the strongest decision that can be inferred from its knowledge base and belief base. The strongest decision is the decision predicate with the highest dob value that is not falsified by any other information (line 11), DECIDETOSHARE returns this final decision. Note if two conflicting actions are inferred with the same dob value, then doing that action or not would both be a correct decision.

4 Argumentation

Argumentation is an approach where arguments with justifications are derived from a knowledge base, and strong arguments attack weak ones. The goal is to find a winning argument according to chosen semantics. In this paper, we use ASPIC [Fox *et al.*, 2007], which is a structured argumentation framework that is based on Dung's framework [Dung, 1995]. The agent is equipped with an ASPIC engine (i) to reason about its knowledge base and belief base, (ii) to compute dob values for the received information, and (iii) to find the strongest decision.

ASPIC: The knowledge base consists of: facts, strict rules and defeasible rules. Each strict rule is of the form σ_1,...,$\sigma_m \leftarrow \sigma_0$ ($m \geq 0$, $\sigma_i \in \mathcal{L}$), and each defeasible rule is of the form σ_1,...,$\sigma_m \Leftarrow \sigma_0$ ($m \geq 0$, $\sigma_i \in \mathcal{L}$) where $\mathcal{L}$ is a logical language and σ_i is a first-order predicate. In strict rules, the conclusion is always true if the premises hold whereas in defeasible rules, the conclusion may be true hence it can be refuted. Each fact and defeasible rule can be associated with a dob value. The default dob value of a defeasible rule is 0.9 unless specified explicitly (e.g., N_1). The strength of an argument is evaluated according to dob values. In ASPIC, an argument can be attacked on its uncertain premises, on its defeasible inferences (e.g., undercutting), or on the conclusions of its defeasible inferences. We use the weakest-link principle, where dob value of an argument is the minimum

Table 2: Trust Norms of an *:agent* as ASPIC Rules

C_1 : [*keep*(A, X)] *info*(X) $\Leftarrow$ *says*(A, X, B) H.
C_2 : *dob*(A, *info*(X), V_3) $\leftarrow$ *trust*(A, V_1), *says*(A, X, V_2), *mult*(V_1, V_2, V_3).
C_3 : *mult*(V_1, V_2, V_3) $\leftarrow$ *is*(V_3, $*$(V_1, V_2)).
C_4 : $\sim$*keep*(A, X) $\leftarrow$ *dob*(A, *info*(X), V_3), $<$(V_3, TH).
C_5 : *info*(X) $\Leftarrow$ $\sim$*keep*(A, X) L.

dob value over all of its sub-arguments. The norm head is assigned the lowest of dob values of the norm body predicates. For example, in N_9, $\sim$*atHome*(*:alice*, Nov30) is the weakest sub-argument with a computed dob value of 0.2. Hence, *missing*(*:alice*, Nov30) is inferred with this same dob value. In argumentation theory, it is possible to choose different semantics to compute extensions (i.e., sets of arguments). Then, the agent can use a credulous or skeptical inference for its decision making. In our work, the default behavior of the agent is not to share a specific content. Hence, finding an argument in at least one preferred extension is enough for the agent to share the content. Therefore, agents accept arguments under preferred credulous semantics.

In Table 2, we show the norms that an agent follows to compute a dob value for the received information. In ASPIC, we can define defeasible rules with schemes, which can contain any variable used in the rule itself. Hence, undercutters can be formulated per instance of a scheme. The scheme names are enclosed in square brackets at the beginning of a rule as shown in C_1. C_1 states that if an agent A gives information X with a dob value B, then there is new information X that should be considered with a belief of H by the receiver agent. The *keep* scheme consists of two variables: the agent A and the information X. When C_1 is applied, a scheme instance will be generated as well. C_2 is used to compute a dob value for the received information. V_1 is the trust value for the agent A, V_2 is the dob value of the agent A for the information X and V_3 is the multiplication of V_1 and V_2 (rule C_3). V_3 becomes the new dob value for the received information. The rules C_2 and C_3 are used for the dob value computation according to Equation 1. In C_4, the agent decides not to keep the received information in case the computed dob value is below its threshold (TH). Note that C_4 is an undercutter for C_1. If C_4 is applied, the agent keeps information with a low belief L (C_5). In other words, the agent keeps the information with a belief of H while other received information is kept with a belief of L in the agent's belief base. C_1 and C_5 are used to evaluate the received information as shown in Equation 2. Recall that H and L are high and low dob values specified by the agent. Here, L and H values are set to 0.2 and 0.9, respectively.

5 Evaluation

We first demonstrate how our approach handles our running examples and then show its theoretical results. Our implementation uses the ASPIC tool[1]. The norms in Tables 1-2 and the factual information of the agent are specified, and the decision to share is checked as a query (e.g., *share*(*:alice*, *:footage*, Nov30)).

5.1 Execution

In Figure 1, we show the interactions between four agents of Examples 1 and 2. For clarity, we only depict the relevant information being exchanged in questions and responses (Definitions 4 and 5). Each agent instance name is followed by the trust value of *:cam* for that agent. For example, *:cam* has a trust value of 0.3 for *:bob*. *:bob* requests Alice's footage taken in November 30. *:cam* evaluates this request according to its knowledge base and belief base. At this point, there are two active contexts: C_d and *:work*. *:cam* has a piece of information *atWork*(*:alice*, Nov30) with a dob value of 0.8. It applies the norms of active contexts. All missing predicates in KB are generated in form of questions (e.g., *missing*(*:alice*, Nov30)). There is only one active relationship in *:work* context since *isBossOf*(*:bob*, *:alice*) holds in KB. Hence, it asks *:bob* a set of questions and gets to know that Alice is missing with a dob value of 0.9. *:cam* evaluates this information according to its trust norms (Table 2). *:cam*'s threshold value is 0.7, the computed dob value for the received information is below threshold, hence *:cam* keeps this information with a low dob value of 0.2. Now, *:em* becomes an active context as well (N_4). In *:em* context, *:homeSensorOf* and *:isResponsibleFor* are the defined relationships. Therefore, *:cam* can ask questions to *:home* and *:jack*. *:home* reports that Alice is not at home with a dob value of 0.6. *:cam* has a trust value of 0.9 for *:home*. However the computed dob value for the received information is not enough, hence *:cam* keeps this information with a low confidence. It finds out that Alice is missing since *:cam* knows that *workday*(Nov30) and $\sim$*onleave*(*:alice*, Nov30) (N_9 and N_4). From *:jack*, it learns that Alice is missing with a dob value of 0.9. *:cam* has a high trust value for *:jack*, thus the computed dob value for this information is above threshold and keeps this information with a high dob value of 0.9. It finds out that Alice is in an emergency state with a high confidence this time (N_4). It cannot collect any more information from other agents. According to its ASPIC engine, the decision with the highest dob value (0.9) is *share*(*:alice*, *:footage*, Nov30), which is not attacked by any other information. Hence, the strongest decision is to share Alice's footage.

5.2 Results

Theorem 1 (Privacy Decision)**.** *Given a world view $\langle \mathcal{C}, \mathcal{A}, \mathcal{F}, getTrustValue \rangle$ of an agent a_j and a question $q_n = \langle a_i, a_j, X \rangle$ where X is an instance of a share predicate, a_j can always reach a decision to share or not to share the content specified in X using Algorithm 1.*

Proof Sketch. In order to come up with a privacy decision, the agent should infer *share* or $\sim$*share* instances in its knowledge base and belief base. The agent will ask agents in A for all the missing predicates in $C.\mathcal{N}$. In the worst case, all agents in A will be related to a_j with a relation in $C.\mathcal{R}$ and

[1] http://aspic.cossac.org

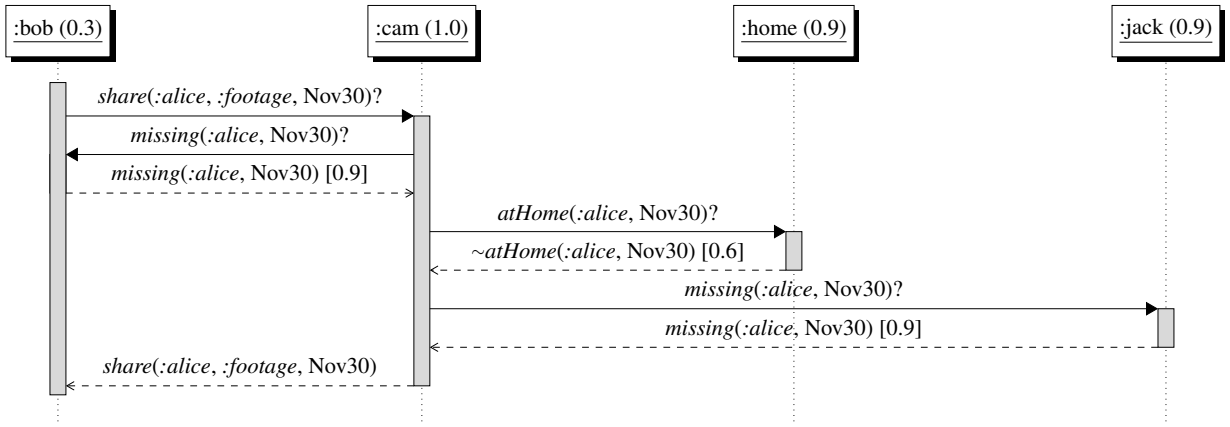

Figure 1: The sequence diagram showing the interactions between agents

will be contacted. Since C is finite, the algorithm will always reach line 10. Assume that $C' \subseteq \mathcal{C}$ is the set of active contexts in line 10. By Definitions 2, 3 and 6 we know that at least C_d is in C' and its norms will reach a decision.

Theorem 2 (Best Effort)**.** *Given a question $q_n = \langle a_i, a_j, X \rangle$ where X is an instance of a share to a_j, Algorithm 1 produces a sound privacy decision with respect to a given world view $\langle \mathcal{C}, \mathcal{A}, \mathcal{F}, getTrustValue \rangle$.*

Proof Sketch. Assume that the agent decides to share a content that it should not. This could happen for two reasons: (i) *There is a missing information in KB and BB of the agent*. The agent uses its internal knowledge, and it can also consult other agents. Algorithm 1 ensures all agents in A are contacted for all predicates in $C.\mathcal{N}$. With the given view, it is not possible to access any more information than that is accessed by agent a_j. Hence, with the given view, it is not possible to access more information. (ii) *KB and BB include complete information but the final decision is wrong*. In KB and BB of the agent, all information is associated with `dob` values. All attack relations between arguments are generated accordingly. The final decision is the strongest argument, which may be attacked by weaker arguments. The strongest argument is the correct decision to be inferred regarding the world view of the agent as guaranteed by the preferred credulous semantics of ASPIC. Hence, the final decision on the given information must be correct. Since neither of these reasons is possible, the privacy decision computation is sound.

6 Discussion

Privacy problems in IoT resemble the privacy problems in online social networks the most, because of the interaction and distribution of nodes. There have been various works to detect and avoid privacy violations in online social networks. Kökciyan and Yolum develop PRIGUARD, a system that models online social networks semantically and reasons on the correct processing and dissemination of information through policies [Kökciyan and Yolum, 2016]. Kafalı *et al.* develop Revani, where they provide patterns to revise normative privacy specifications and use model checking to verify privacy requirements [Kafalı *et al.*, 2016]. Kökciyan *et al.* propose an argumentation-based approach to preserve privacy collaboratively [Kökciyan *et al.*, 2017].

Other computational models for Contextual Integrity (CI) have been proposed before. Barth *et al.* propose a logical framework where a privacy policy is a set of distribution norms represented as temporal formulas [Barth *et al.*, 2006]. They show the expressiveness of their model by representing various privacy provisions such as HIPAA. In their work, users can be active in a single context that is an input to their model. However, in our work users can be involved in multiple contexts, which are inferred by the agent itself.

Krupa and Vercouter propose a CI-based framework to detect privacy violations in virtual communities [Krupa and Vercouter, 2012]. The information subject is allowed to specify privacy policies that should be respected at dissemination time. In our work, since an IoT entity cannot collect privacy policies of people individually and enforce them, an IoT entity uses its own knowledge base, which can be enriched by external information, to make a privacy decision.

Criado and Such propose a computational model where an agent can learn implicit contexts, relationships and appropriateness norms to prevent privacy violations to occur [Criado and Such, 2015]. Similar to our work, users can be involved in multiple contexts. Contrary to Criado and Such, in our work, agents use trust values to compute a `dob` value for the received information. Moreover, the context is not defined as a set of users; but through a set of norms and relations (see Definition 1).

Murukannaiah and Singh develop Platys, a framework targeted for place-aware applications [Murukannaiah and Singh, 2015]. They formalize the concept of place through location, activity and social circle. In our work, we enable agents to reason on privacy correctly based on a derived context. Accurate calculation of place could help understand context better and thus would complement our work.

For future work, we want to capture the relations between contexts so that a context can inherit norms and relations from a second context. This will help the system function even when the contexts are not specified fully. An important direction is to enable the system to learn the norms of the contexts over time so that the system can scale better.

References

[Abowd *et al.*, 1999] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999.

[Barth *et al.*, 2006] Adam Barth, Anupam Datta, John C Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *IEEE Symposium on Security and Privacy*, pages 184–198, 2006.

[Criado and Such, 2015] Natalia Criado and Jose M. Such. Implicit contextual integrity in online social networks. *Information Sciences: An International Journal*, 325:48–69, 2015.

[Dung, 1995] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[Fong, 2011] Philip W.L. Fong. Relationship-based access control: Protection model and policy language. In *Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 191–202, 2011.

[Fox *et al.*, 2007] J. Fox, D. Glasspool, D. Grecu, S. Modgil, M. South, and V. Patkar. Argumentation-based inference and decision making–a medical perspective. *IEEE Intelligent Systems*, 22(6):34–41, 2007.

[Kafalı *et al.*, 2016] Özgür Kafalı, Nirav Ajmeri, and Munindar P. Singh. Revani: Revising and verifying normative specifications for privacy. *IEEE Intelligent Systems*, 31(5):8–15, 2016.

[Kökciyan and Yolum, 2016] Nadin Kökciyan and Pınar Yolum. PriGuard: A semantic approach to detect privacy violations in online social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2724–2737, 2016.

[Kökciyan *et al.*, 2017] Nadin Kökciyan, Nefise Yaglikci, and Pınar Yolum. An argumentation approach for resolving privacy disputes in online social networks. *ACM Transactions on Internet Technology (TOIT)*, 2017. To appear.

[Krupa and Vercouter, 2012] Yann Krupa and Laurent Vercouter. Handling privacy as contextual integrity in decentralized virtual communities: The PrivaCIAS framework. *Web Intelligence and Agent Systems*, 10(1):105–116, 2012.

[Murukannaiah and Singh, 2015] Pradeep K Murukannaiah and Munindar P Singh. Platys: An active learning framework for place-aware application development and its evaluation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):19, 2015.

[Nissenbaum, 2004] Helen Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, 79:119, 2004.

[Sicari *et al.*, 2015] S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146 – 164, 2015.

[Solove, 2006] Daniel J Solove. A taxonomy of privacy. *University of Pennsylvania Law Review*, pages 477–564, 2006.

[Wang and Singh, 2007] Yonghong Wang and Munindar P. Singh. Formal trust model for multiagent systems. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 1551–1556, 2007.

[Weber, 2010] Rolf H Weber. Internet of Things: New security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30, 2010.