Theme Article: Digital Trust: Trust Management in Cyberspace

TURP: Managing Trust for Regulating Privacy in Internet of Things

Nadin Kökciyan
University of Edinburgh

Pınar Yolum
Utrecht University

***Abstract*—Internet of Things (IoT) applications, such as smart home or ambient-assisted living systems, promise useful services to end users. Most of these services rely heavily on sharing and aggregating information among devices, many times raising privacy concerns. Contrary to traditional systems, where privacy of each user is managed through well-defined policies, the scale, dynamism, and heterogeneity of the IoT systems make it impossible to specify privacy policies for all possible situations. Alternatively, this article argues that handling of privacy has to be reasoned by the IoT devices, depending on the norms, context, as well as the trust among entities. We present a technique, where an IoT device collects information from others, evaluates the trustworthiness of the information sources to decide the suitability of sharing information with others. We demonstrate the applicability of the technique over an IoT pilot study.**

Internet of Things (IoT) is emerging as an area where privacy is crucial but pose different challenges than of Web systems.[1] In Web systems, often users first log in using their credentials, get authenticated, and then their privacy settings are in effect. The privacy settings are frequently understood and exercised as *informed consent* as in General Data Privacy Regulation. Put simply, a website user gives consent as to which of her personal data will be collected and shared. The main assumption behind this is that the user herself is capable of thinking through every possible occasion, under different contexts and knows specifically the effect of sharing a piece of information with certain others. With IoT systems, this is rarely the case. A human walking on the street will not be aware of the cameras around her, whether

Digital Object Identifier 10.1109/MIC.2020.3020006
Date of publication 27 August 2020; date of current version 20 November 2020.

they are recording at the time, whom the footage is being shared with, and what possible effects this can bring.[2] This makes it both impractical and ineffective to regulate privacy using privacy policies that specify informed consent.

A different formulation of privacy is *contextual integrity*,[3] which understands privacy in a social context, by defining norms that govern appropriate *information flows*. Depending on the context of the user, some information is appropriate to share. A typical example is that in a medical context, it would be appropriate to share a patient's information with medical staff even if explicit consents were not given for each medical staff. Contextual integrity as a theory is powerful to be applied even in situations where context as well as individual players changes frequently, as it advocates reasoning on contexts rather than putting users responsible for specifying the correct behavior for each possible case.

To clearly determine the context in an IoT application, it is usually necessary to factor in information from different entities, who would know different aspects of context, such as location, participants, or activity. With each new information that is received from other IoT devices, an IoT device can interpret the context better and make a sharing decision accordingly.

TRUST FOR PRIVACY

Many of IoT devices will have varying capabilities and be possibly managed by different principles. Hence, when an IoT device reports on a piece of information, the quality and accuracy of that information may vary significantly. For example, a particular sensor might not detect objects after the night comes down, whereas a home security camera can even perform face recognition. This requires information from different IoT devices as well as humans to be treated and trusted differently.[1]

Running Example: Is Alice Missing?

Alice works for a firm. When she comes and leaves work, she uses her ID card to log her hours. On November 30th, Alice's boss cannot reach her. Alice's phone is on her desk in the office, but Alice is not there. Her boss does not have the credentials to see Alice's emergency contacts so her boss requests access to see them. Alice's phone can gather information from other devices, such as the motion sensor at Alice's home and the CCTV camera near her home. Under normal circumstances, Alice's phone should not reveal the emergency contacts, but if Alice is missing, it might be to Alice's benefit that the information is indeed revealed. How can the phone decide on the context autonomously and make a decision to reveal information about Alice's emergency contacts?

In order to make this decision, first the necessary physical and network IoT layers should be in place to facilitate communication among entities; e.g., the phone should be able to gather information from other IoT devices that it sees fit. However, it could easily be the case that these IoT devices themselves are not confident about the information they provide. Either they are not willing to share information, or that the phone knows from previous experience that certain information sources are not trustworthy. For example, the motion sensor at home might itself be confident with the measurements it provides; however, the phone might not trust the motion sensor as much, since the sensor may have limited access to the area.

This calls for a trust management system for decision making in IoT systems such that a device that receives information from others will assess the trustworthiness of the information it receives. In order to ensure that the gathered information is aggregated correctly, the trust management system should receive feedback when appropriate from end users so that assessed trustworthiness of entities can be updated appropriately.

Accessing Trust in IoT

Modeling and managing trust has been studied widely especially in the context of e-commerce in multiagent systems to find trustworthy service providers to carry transactions with. Each consumer agent maintains a trust value for some of the service providers in the system. When a consumer is in need of a service, it selects a service provider by considering its trust and possible others' trust in the service provider. After service provisioning, the consumer agent updates its trust in the service provider based on the quality and outcome of the service.[4]

While the aforementioned procedure provides intuitive points to design a trust management system for IoT systems, building and maintaining trust in IoT applications exhibit properties that are not apparent in e-commerce systems.[5] First, while in service dealings, there are many service providers to choose from in IoT applications, there could be a single device, which you are obliged to use. Second, e-commerce service providers provide services mostly in the same context. Thus, after frequent transactions with a service provider, it is possible to have an accurate estimation of trustworthiness of the provider. However, IoT entities may collect information from previously unknown entities and make dynamic aggregations on data that were not present before. Hence, judging the trustworthiness of an entity is difficult. Finally, in e-commerce, after one service provider provides the service, its trust value is updated based on its performance. Whereas in IoT applications, a decision is reached based on information collected from many devices. Hence, it is not easy to identify whose trustworthiness to update and to what extent.

TURP

We start with the model of Kökciyan and Yolum,[6] where IoT entities and users of the system are represented as software agents. Each agent has information about its environment as well as some of the other agents around it. Agents can request and provide information to each other; however, it is possible that an agent might decline to provide information. When an agent provides information to another agent, it associates a confidence value, *c-value* (the higher the value is, better the confidence is). Furthermore, each agent maintains a trust value for another agent, both in the range of 0–100. When an agent needs to make a privacy decision, it uses its own knowledge as well as information collected from other agents, factoring in the confidence as well as the trust value to decide on the trustworthiness of the information.

Reasoning on Context

TURP uses a disjunctive logic programming language, *Disjunctive Datalog*,[7] extended with true negation to represent and reason on information collected from other agents. This language supports nonmonotonic reasoning; hence, an agent can reason with conflicting information in its knowledge base (KB), which consists of facts as well as rules. This is important because information collected from devices might easily conflict.

Rules Rules can serve different purposes such as inferring more information (e.g., context), or representing contextual norms that define the appropriateness of sharing some information. A rule consists of a head and a body. The body of the rule is a conjunction of predicates from a logical language, and the head of the rule can be a disjunction of predicates. All the predicates include a *c*-value V to show the confidence of a piece of information. Negations are allowed both in the body and the head of a rule (e.g., *-share_details* predicate).

In Table 1, the example rules of the *phone* agent are shown as Disjunctive Datalog rules. Our example contains five agents (*phone*, *sensor*, *cctv*, *boss,* and *punch*) and two contexts: *work* and emergency (*em*). All the predicates in the language are in italic text, and each rule is denoted as R_i. When a user is missing, this may imply that the user is in emergency context (R_5). R_3 represents a contextual norm that states if a user is in emergency context, it is appropriate for the reasoning agent (i.e., *phone*) to share details of the user. The default context C_d dictates that the user's details should not be shared with a *c*-value of 50, a value that can be set by the agent (R_0 and R_1). The *work* context stops the agent from revealing the user's details (R_4 and R_2). When a rule body includes more than one predicate; the *c*-value of the inferred information, predicate in the head of the rule, is the minimum *c*-value among all the predicates. Hence, the agent adopts a cautious stance while making an inference. R_6 is an example of this; if an agent is not on leave or not home on a work day, that agent may be missing with a *c*-value equal to the minimum of other predicates' *c*-values.

The reasoning agent computes a *c*-value for each information that it receives from other agents (R_7). For this, it multiplies the trust value of the other agent, TR, by the *c*-value associated with the received information, CV, and normalizes it to 100, which gives the final *c*-value of the received information.

Table 1. Example rules of *phone* as Disjunctive Datalog rules.

R_0 : *in_context*(A, C_d, T, 50) ⇐ *agent*(A), *time*(T).
R_1 : *-share_details*(A, T, V) ⇐ *in_context*(A, C_d, T, V).
R_2 : *-share_details*(A, T, V) ⇐ *in_context*(A, *work*, T, V).
R_3 : *share_details*(A, T, V) ⇐ *in_context*(A, *em*, T, V).
R_4 : *in_context*(A2, *work*, T, V) ⇐ *keep*(A1, *info*(*at_work*(A2, T)), V).
R_5 : *in_context*(A2, *em*, T, V) ⇐ *keep*(A1, *info*(*missing*(A2, T)), V).
R_6 : *keep*(*phone*,*info*(*missing*(A, T)), V) ⇐ *keep*(_, *info*(*not_on_leave* (A, T)), V1), *keep*(_, *info*(*workday*(T)), V2), *keep*(_, *info*(*not_at_home* (A, T)), V3), *minimum*(V1, V2, V3, V).
R_7 : *keep*(A, *info*(X), V2) ⇐ *trust*(A, TR), *says*(A, X, CV), V1 = TR*CV, /(V1,100,V2).

Context Computation Using the information that it has collected and by applying suitable rules, a reasoning agent can find out different contexts that an agent can be in. Because of the way the reasoning and the rules are set up, it can easily be the case that the agent might consider an agent to be in more than one context. This is a desirable property as a situation can belong to multiple contexts. Note that because the information is recorded with *c*-values, the agent will also compute a *c*-value with each inferred context. We call each possible interpretation for a context a *model*. Each model has one decision predicate (*share* or *-share*), which represents a privacy decision of the model with a certain *c*-value, which follows from the contextual norms (e.g., R_1, R_2, and R_3). The agent computes all possible models, finds the model with the highest *c*-value, and performs the decision associated with the context of the model. We categorize the models as *share* and *-share* models, based on their outcome. Figure 1 depicts all models for the running example, where agents with different trust values provide information with various *c*-values.

Trust Update

The trust values of the agents are updated after the user gives a dichotomous feedback as to whether the decision was correct or not. In many models, the sharing decision does not depend on a single-agent but multiple agents; these agents are the *relevant agents* to make a sharing decision in that model. Notice that this aspect of trust update is an important difference compared to the trust updates in e-commerce setting, where after a service engagement, a negative feedback is only related to the actual service provider that delivered the service. Here, however the decision is taken based on information from multiple entities and, thus, all relevant agents have to be considered.

If based on the feedback, the share decision is deemed wrong, then the trust values of the entities should be updated in such a way that the user's details would not be revealed again if the same decision was being taken now. There are two consequences of this. First, the contexts that yielded the *share* decision at a higher *c*-value than the *-share* decision were not correctly identified. This, in return, means that the entities that provided information that led to this decision should have been trusted less. Second, at least one of the contexts that would lead to a *-share* decision should have received a higher *c*-value so that it would have been identified as the winning context.

To make the proper updates, first, the agent identifies the *share* model and *-share* model with the highest *c*-value. Next, it computes an average *c*-value of these two models, the θ value. θ will be a threshold to find relevant agents to update (decrease or increase) their current trust values; t_a denotes the trust value of the agent a. Trust value updates guarantee a *-share* model to infer the winning context while updating trust values of the relevant agents.

Decreasing Trust The trust update will be applied to the relevant agents involved in a *share* model with a *c*-value greater than θ (i.e., $\mathrm{CV}(m) > \theta$). The reasoning agent will set the trust value of agent a in model m, (t_a^m), as the maximal trust value that would change the sharing decision, considering the *c*-value provided by the agent a while sharing information (CV_a^m). Equation (1) shows this calculation, where λ is the *penalty factor* that is a value between 0 and 1. If the reasoning agent chooses a lower λ value, an agent that provides misleading information is trusted less. This equation ensures that the agent gets a trust value to compute a *share* decision model with a *c*-value lower than θ. Note that an agent can be involved in more than one model in this step. The minimum trust value of an agent is computed

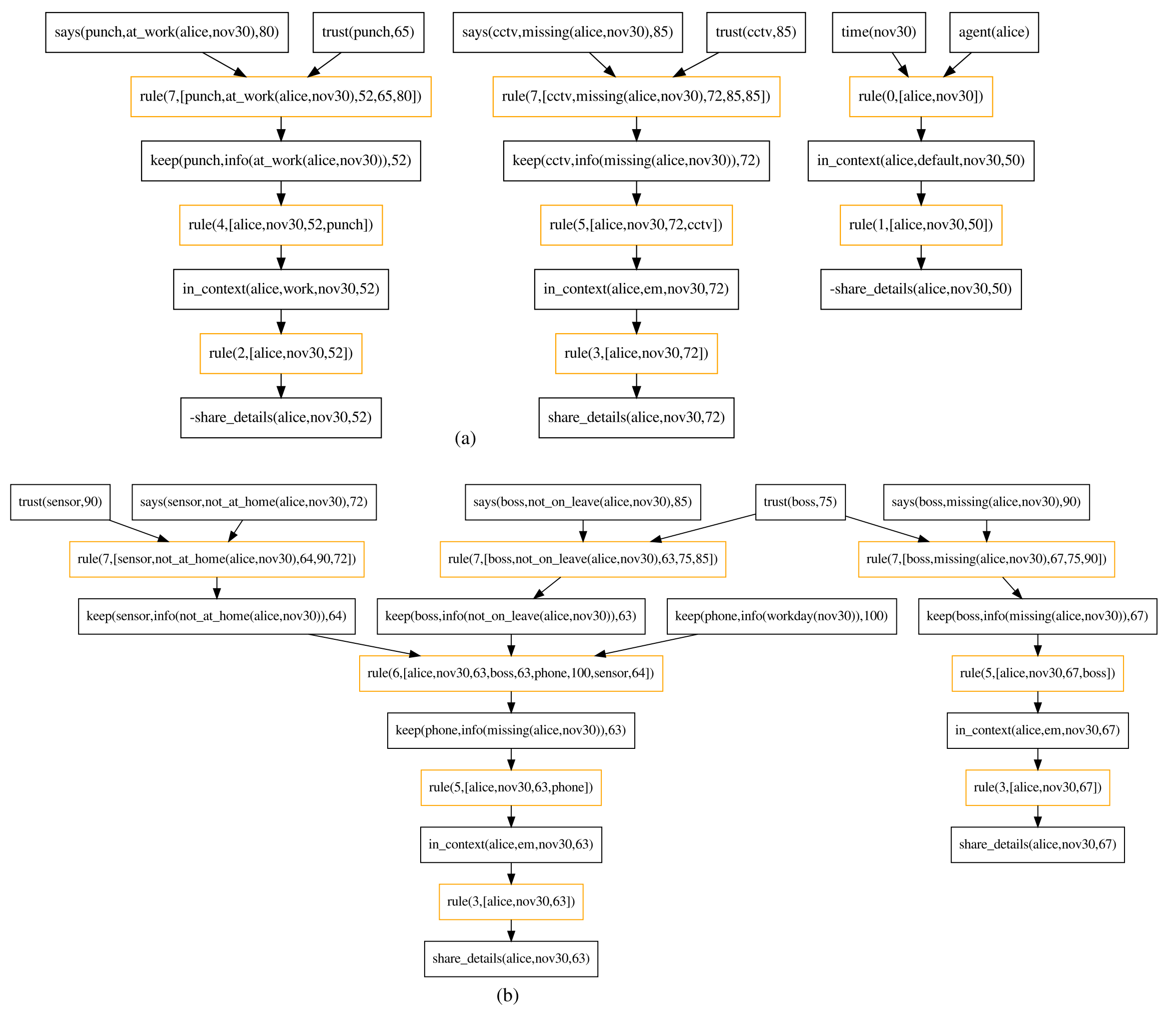
says(punch,at_work(alice,nov30),80)
trust(punch,65)
rule(7,[punch,at_work(alice,nov30),52,65,80])
keep(punch,info(at_work(alice,nov30)),52)
rule(4,[alice,nov30,52,punch])
in_context(alice,work,nov30,52)
rule(2,[alice,nov30,52])
-share_details(alice,nov30,52)
says(cctv,missing(alice,nov30),85)
trust(cctv,85)
rule(7,[cctv,missing(alice,nov30),72,85,85])
keep(cctv,info(missing(alice,nov30)),72)
rule(5,[alice,nov30,72,cctv])
in_context(alice,em,nov30,72)
rule(3,[alice,nov30,72])
share_details(alice,nov30,72)
time(nov30)
agent(alice)
rule(0,[alice,nov30])
in_context(alice,default,nov30,50)
rule(1,[alice,nov30,50])
-share_details(alice,nov30,50)
(a)
trust(sensor,90)
says(sensor,not_at_home(alice,nov30),72)
rule(7,[sensor,not_at_home(alice,nov30),64,90,72])
keep(sensor,info(not_at_home(alice,nov30)),64)
says(boss,not_on_leave(alice,nov30),85)
trust(boss,75)
says(boss,missing(alice,nov30),90)
rule(7,[boss,not_on_leave(alice,nov30),63,75,85])
rule(7,[boss,missing(alice,nov30),67,75,90])
keep(boss,info(not_on_leave(alice,nov30)),63)
keep(phone,info(workday(nov30)),100)
keep(boss,info(missing(alice,nov30)),67)
rule(6,[alice,nov30,63,boss,63,phone,100,sensor,64])
rule(5,[alice,nov30,67,boss])
keep(phone,info(missing(alice,nov30)),63)
in_context(alice,em,nov30,67)
rule(5,[alice,nov30,63,phone])
rule(3,[alice,nov30,67])
in_context(alice,em,nov30,63)
share_details(alice,nov30,67)
rule(3,[alice,nov30,63])
share_details(alice,nov30,63)
(b)

Figure 1. All the models generated by the *phone* agent. The rules are depicted as orange boxes, whereas all other predicates are depicted as black boxes. An arrow from a black box to an orange box shows inputs to a rule, and an arrow from an orange box to a black box means that the rule was applied to conclude the target predicate. (a) Best -*share*, the best *share,* and the default -*share* models. (b) Remaining *share* models.

according to (2), where $[\text{dec}]t_a$ denotes the trust value of an agent in the decreasing step

$$t_a^m = \lambda \times \max_{0 \le T \le 100} \left(T < \frac{\theta \times 100}{\text{CV}_a^m} \right), \text{CV}(m) > \theta \tag{1}$$

$$[\text{dec}]t_a = \min(\{t_a^m | \text{CV}(m) > \theta\}) \tag{2}$$

Increasing Trust It is necessary to increase the trust values of agents who provided some information to support a -*share* decision. In other words, if these agents had been trusted more, the right sharing decision would have been reached. Since trust should be difficult to build up, the reasoning agent will only minimally increase trust values of agents who are involved in the best -*share* models. Equation (3) computes nshare value, which is the highest *c*-value in -*share* models ($\mathcal{M}^-$). Equation (4) ensures that the agent obtains the minimal trust value to compute a -*share* decision model with a *c*-value greater than θ. Note that multiple best -*share* models can exist. Equation (5) ensures that the minimum trust value is assigned to an agent, which is denoted as $[\text{inc}]t_a$.

$$\text{nshare} = \max(\{\text{CV}(m) | m \in \mathcal{M}^-\}) \tag{3}$$

$$t_a^m = \min_{0 \leq T \leq 100}\left(T > \frac{\theta \times 100}{\mathrm{CV}_a^m}\right), \mathrm{CV}(m) = \mathrm{nshare} \tag{4}$$

$$[\mathrm{inc}]t_a = \min(\{t_a^m | \mathrm{CV}(m) = \mathrm{nshare}\}) \tag{5}$$

The final trust values of the agents will be assigned after the decreasing and increasing steps, as shown in (6). If the agent was not involved in the given model, its trust stays the same

$$t_a = \begin{cases} [\mathrm{inc}]t_a, & \text{if } [\mathrm{inc}]t_a \text{ exists} \\ [\mathrm{dec}]t_a, & \text{if } [\mathrm{inc}]t_a \text{ does not exist} \\ & \text{and } [\mathrm{dec}]t_a \text{ exists} \\ t_a, & \text{otherwise} \end{cases}. \tag{6}$$

EVALUATION

We implemented TURP[†] to run on user agents in an IoT setting using DLV reasoner[8] and conducted a pilot study. Our implementation is a graph-based approach using Python, where we apply depth-first search algorithm to identify the relevant agents in the computed models.

To Share or Not to Share?

We first revisit the running example. When *boss* requests access to see Alice's emergency contacts, it provides information about Alice being missing (i.e., *missing*(*alice*, nov30)). *phone* infers the *em* context by applying R_5, and it contacts *sensor* and *cctv* to collect more information about Alice. Similarly, *work* is an inferred context (R_4) since *punch* provides information about Alice being at work (i.e., *at_work*(*alice*, nov30)).

Once *phone* finishes collecting information from other agents, it computes all models according to its KB [see Figure 1]. Figure 1(a) depicts the best *-share* and *share* models with *c*-values 52 and 72, respectively. In the best *share* model, *cctv*, which has a trust value of 85, is the only agent providing information about Alice being missing with a *c*-value of 85. This information is kept in KB with a *c*-value of 72 (R_7). The *em* context is inferred (R_5), which is used to conclude Alice's details should be shared (R_3). Since this is the model with the highest *c*-value, *phone* decides to share emergency contacts of Alice.

Alice considers this to be the wrong decision and, thus, gives negative feedback. Now, *phone* will update the trust values of the relevant agents. First, it gets all the *share* models with a *c*-value above θ that is 62, the average of *c*-values of the best *share* and *-share* models. For each such model, it finds all the relevant agents and updates trust values according to (1), with λ set to 0.7. $m(c)$, $m(bs)$, and $m(b)$ are the *share* models to be considered ($m(x)$ denotes a model where *x* is the set of relevant agents, only initial letters are used). In $m(bs)$, *boss* and *sensor* are two agents providing information. According to (1), $t_{\mathrm{sensor}}^{m(bs)}$ and $t_{\mathrm{boss}}^{m(bs)}$ are computed as 60 and 50, respectively. Similarly, $t_{\mathrm{cctv}}^{m(c)}$ and $t_{\mathrm{boss}}^{m(b)}$ become 50 and 48, respectively. *boss* is the only agent that appears in two models: $m(bs)$ and $m(b)$ [see Figure 1(b)]. The minimum trust value is the one computed in $m(b)$, the trust value of *boss* is updated as 48.

The *-share* model with the highest c-value (52) is $m(p)$. Since the only agent providing some information is *punch*, it will be rewarded according to (4), yielding 78 for $t_{\mathrm{punch}}^{m(p)}$. *punch* is not involved in any other *share* model; therefore, its trust value is set to 78.

Running the same scenario with the updated trust values yields a *c*-value of 43 for the best *share* model and 62 for the best *-share* model [see Figure 2(b)]. Hence, *phone* would decide not to share Alice's details.

Pilot Study

To conduct a pilot study, we use an existing dataset collected from users, containing interactions with sensors and users' privacy expectations in different IoT scenarios.[2] Each scenario in the dataset contains features such as data collected by a particular device, the purpose, and the location. Each user is given 14 scenarios, where in each scenario the user interacts with a single device and labels it with *Allow* or *Deny*.

In order to capture the rules governing these scenarios, we ran the *a priori* association rule learning algorithm[9] on scenarios labeled by 775 users, with a minimum support value of 0.1 and a minimum confidence value of 0.6. We then selected 20 rules with a high confidence value that contain either *Allow* or *Deny* labels in the

[†][Online]. Available: https://git.ecdf.ed.ac.uk/nkokciya/turp

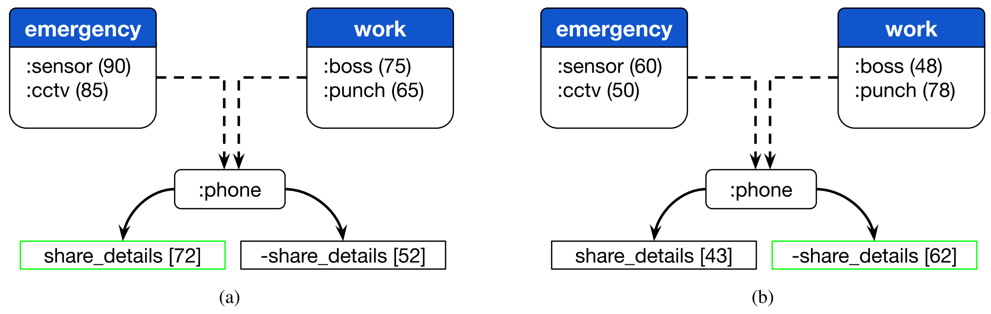

Figure 2. *best* models generated by the *phone* agent after consulting agents with different trust values, belonging to contexts (*emergency* and *work*). (a) Best models computed in the initial case. (b) Best models computed after the trust update phase. After these updates, the privacy decision of *phone* would become not sharing her details if the same decision was being taken again. The dashed arrows show the information flow, and the solid arrows depict the control flow. The numbers in parentheses are the trust values of agents, and each decision predicate is followed by a confidence value shown in brackets. In (a), *phone* decides to share Alice's details in the initial case. In (b), *phone* decides not to share Alice's details after trust updates.

head equally. Each association rule (AR) was automatically transformed into a Disjunctive Datalog rule. The antecedent of each AR was used to generate a context rule such that the conjunction of features implies a context. The context information was then used to generate a contextual norm regarding *Allow* or *Deny* label, yielding a total of 40 Disjunctive Datalog rules.

We then use different 14 scenarios, where there are overall eight IoT devices, and a user interacts with four IoT devices more than once. Each scenario includes one IoT device, so each *share* or *-share* model includes one agent. In each scenario, features are considered as information pieces provided by the IoT device and are associated with a random confidence value between 0 and 100. For each scenario, the user's agent makes an automated decision to share or not to share using TURP. We use the label provided by the user in the dataset as the user feedback for our model. In case of a negative feedback, the trust value of the IoT device is updated accordingly.

We run our experiments with a different range of trust values changing between 40 and 100. The confidence values of features are generated once and assigned to each IoT device. We initialize all IoT devices with the same trust value, and then, the trust values are updated after the user feedback. We observe that even under various unknowns as described earlier, the agent by employing TURP can model the trustworthiness of the other agents and can update its trust values to correct its reasoning after feedback. Based on our qualitative analysis of the scenarios, we identify the following points as important to use TURP in a realistic IoT setting. First, the set of Disjunctive Datalog rules are important in ensuring that the agent can generate adequate models. If the selected *share* model receives negative feedback from the user, the system should ensure that there are some *-share* models to update. Validation of this at the time of setup is useful. Second, initial values for trust determines how conservative the agent will act, where low trust values reinforce *Deny* decisions. Allowing the user to configure this value and making it compatible with the default context enables realistic outcomes. Finally, if the sensors vary in the quality of information they provide, the confidence values need to be set to reflect that to increase the accuracy of the models.

RELATED WORK

Privacy in online social networks has been studied in depth. A group of approaches predict the privacy labels of content that are about to be shared online.[10] Those approaches assume that the privacy of content does not change based on the context. Barth *et al.* present a logical framework where a privacy policy is a set of distribution norms represented as temporal formulas.[11] In their work, users are assumed to be active in a single context, which is provided to the model as an input. On the contrary, here we accommodate multiple contexts and compute the contexts based on the trustworthiness of devices in the system. Another important model is due to Criado and Such, where an agent can learn implicit contexts, relationships, and appropriateness norms to prevent privacy violations.[12] While that work has support for multiple contexts, the trustworthiness of devices has not been dealt with as we do here.

CONCLUSION

TURP enables IoT entities to make context-based privacy decisions, where the context is identified based on information provided from trusted others. TURP enables correct decisions to be taken over time as each entity updates its trust in others after wrong decisions are taken. We conducted a qualitative analysis of IoT scenarios by using TURP. An interesting future direction to consider is sharing of content that belongs to multiple users,[13,14] such as group footage, where sharing it for one user might be appropriate but not for the other as they might be in different contexts.

REFERENCES

1. S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, 2015.
2. P. E. Naeini *et al.*, "Privacy expectations and preferences in an IoT world," in *Proc. 13th Symp. Usable Privacy Secur.*, 2017, pp. 399–412.
3. H. Nissenbaum, "Privacy as contextual integrity," *Washington Law Rev.*, vol. 79, pp. 119–158, 2004.
4. M. Şensoy, J. Zhang, P. Yolum, and R. Cohen, "Poyraz: Context-aware service selection under deception," *Comput. Intell.*, vol. 25, no. 4, pp. 335–366, 2009.
5. C. Fernandez-Gago, F. Moyano, and J. Lopez, "Modelling trust dynamics in the Internet of Things," *Inf. Sci.*, vol. 396, pp. 72–82, 2017.
6. N. Kökciyan and P. Yolum, "Context-based reasoning on privacy in Internet of Things," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 4738–4744.
7. T. Eiter, G. Gottlob, and H. Mannila, "Disjunctive datalog," *ACM Trans. Database Syst.*, vol. 22, no. 3, pp. 364–418, 1997.
8. "Dlvsystem." Accessed: Sep. 14, 2020. [Online]. Available: http://www.dlvsystem.com/
9. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th Int. Conf. Very Large Data Bases*, 1994, pp. 487–499.
10. A. Squicciarini, C. Caragea, and R. Balakavi, "Toward automated online photo privacy," *ACM Trans. Web*, vol. 11, no. 1, pp. 1–29, 2017.
11. A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum, "Privacy and contextual integrity: Framework and applications," in *Proc. IEEE Symp. Secur. Privacy*, 2006, pp. 184–198.
12. N. Criado and J. M. Such, "Implicit contextual integrity in online social networks," *Inf. Sci.: Int. J.*, vol. 325, pp. 48–69, 2015.
13. N. Kökciyan, N. Yaglikci, and P. Yolum, "An argumentation approach for resolving privacy disputes in online social networks," *ACM Trans. Internet Technol.*, vol. 17, no. 3, pp. 27:1–27:22, 2017.
14. R. L. Fogues, P. K. Murukannaiah, J. M. Such, and M. P. Singh, "SoSharP: Recommending sharing policies in multiuser privacy scenarios," *IEEE Internet Comput.*, vol. 21, no. 6, pp. 28–36, Nov./Dec. 2017.

Nadin Kökciyan is currently a Lecturer in Artificial Intelligence with the School of Informatics, University of Edinburgh, Edinburgh, U.K., and a visiting research fellow with the Department of Informatics, King's College London, London, U.K. Her research focuses on developing AI techniques to support decision-making in multiagent systems while preserving privacy. She received the Ph.D. degree in computer engineering from Bogazici University, Istanbul, Turkey. She is the corresponding author of this article. Contact her at nadin.kokciyan@ed.ac.uk.

Pınar Yolum is currently a faculty member with the Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands. She received the Ph.D. degree from North Carolina State University, Raleigh, NC, USA. She serves on the Editorial Boards of *Journal of Autonomous Agents and Multiagent Systems*, *ACM Transactions on Internet Technology*, and *IEEE Internet Computing*. Contact her at p.yolum@uu.nl.