Formal Specification and Verification of User-centric Privacy Policies for Ubiquitous Systems

Rezvan Joshaghani
rezvanjoshaghani@u.boisestate.edu
Boise State University
Boise, Idaho

Stacy Black
stacyblack@u.boisestate.edu
Boise State University
Boise, Idaho

Elena Sherman
elenasherman@boisestate.edu
Boise State University
Boise, Idaho

Hoda Mehrpouyan
hodamehrpouyan@boisestate.edu
Boise State University
Boise, Idaho

ABSTRACT

As our society has become more information oriented, each individual is expressed, defined, and impacted by information and information technology. While valuable, the current state-of-the-art mostly are designed to protect the enterprise/ organizational privacy requirements and leave the main actor, i.e., the user, uninvolved or with the limited ability to have control over his/her information sharing practices. In order to overcome these limitations, algorithms and tools that provide a user-centric privacy management system to individuals with different privacy concerns are required to take into the consideration the dynamic nature of privacy policies which are constantly changing based on the information sharing context and environmental variables. This paper extends the concept of contextual integrity to provide mathematical models and algorithms that enables the creations and management of privacy norms for individual users. The extension includes the augmentation of environmental variables, i.e. time, date, etc. as part of the privacy norms, while introducing an abstraction and a partial relation over information attributes. Further, a formal verification technique is proposed to ensure privacy norms are enforced for each information sharing action.

CCS CONCEPTS

• **Security and privacy → Logic and verification**; • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Privacy, Formal Methods, User-Centric Policies

ACM Reference Format:
Rezvan Joshaghani, Stacy Black, Elena Sherman, and Hoda Mehrpouyan. 2019. Formal Specification and Verification of User-centric Privacy Policies for Ubiquitous Systems. In *23rd International Database Engineering Applications Symposium (IDEAS'19), June 10–12, 2019, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331076.3331105

IDEAS'19, June 10–12, 2019, Athens, Greece

ACM ISBN 978-1-4503-6249-8/19/06…$15.00
https://doi.org/10.1145/3331076.3331105

1 INTRODUCTION

A Privacy Bill of Rights was endorsed by the White House in 2012, a response to an increasingly loud objection of citizens on the lack of privacy and fair information practices guidelines [20]. The predicament was not only recognized by the US government, but has also been investigated and studied at the international stage and has resulted in reports such as "Rethinking personal data: Strengthening trust" by the World Economic Forum (WEF) [40] and "Recommendations for businesses and policymakers" by the Federal Trade Commission (FTC) [12]. Despite all these efforts, ubiquitous online monitoring of users' activities [29] and scandalous data breaches, i.e. Facebook and Cambridge Analytica, continue to haunt Online Social Network (OSN) users [2, 11]. These privacy breaches are often due to a lack of regulatory standardization. Hence, the onus is on the user to take control of: what types of information should be shared with whom and when. However, controlling and managing the information sharing parameters could be a cumbersome and difficult process [15, 21, 44]. Therefore, ample tools and algorithms should be developed and provided to users so they are able to define and enforce their own customized, unambiguous privacy policies and have control over how their information is shared. The state-of-the-art research on privacy management mostly consist of: access control languages [4, 33, 39], different privacy settings in applications, and formal privacy policies [5, 10, 14, 22, 36]. While valuable, the previous works are mostly based on enterprise/organizational privacy management and leave the main actor, i.e., the user, uninvolved or with limited ability to control the information sharing parameter. In addition, existing privacy regulations like HIPAA or a corporation's privacy policies are domain-specific and static with a little or no change over time. On the other hand, the user's privacy policies are dynamic and changing based on many factors, i.e. context, environment, relationship status, etc. In addition to dynamicity, the privacy framework should provide the user with the ability to adapt the policies to their own personal needs, since the definition of privacy varies from person to person based on their personality, cultural background, etc [30].

In order to move towards a more practical solution, this paper proposes a framework to build a user-centric privacy management

system. We focus on developing the main core of this framework, which is the *privacy formalization and verification engine* that allows for the guided and flexible specification of users' privacy intentions. The formalization and verification engine performs formal reasoning about the user's privacy rules to detect privacy violations. Further, the proposed approach ensures that the defined privacy policy is unambiguous and a consistency checking approach is proposed so that all the exiting and newly defined policies are consistent with one another. The underlying formalization utilizes two formal models: 1- the user's information sharing model, and 2- the privacy-preserving model. The user's information sharing model represents all the user's information sharing activities to others. The privacy verification is performed by mapping each user's information sharing parameters (known as a state) to a state in the privacy-preserving model; a state with no mapping indicates a privacy violation. As a proof of concept, the privacy formalization and verification engine is implemented as a Java program that detects privacy violations as the user shares information in real-time. Since this framework is targeted for smart devices, which usually have low memory and low processing power, its performance was evaluated on both a PC and a Raspberry Pi model B to show the practicality of our approach.

The future work will extend the current effort to include: user privacy requirement elicitation, identification and categorization of the information shared by users, and detection of the relationship changes between a user and recipients.

The rest of the paper is organized as follows: section 2 provide the related works. Section 3 has a detailed description of our formalism and verification engine, and the implementation details of our framework are given in section 4. Moreover, the performance evaluation of the proposed framework is given in section 5. Finally, section 6 draws the conclusion of this paper and discusses the future works of our approach.

2 RELATED WORKS

For over 120 years researchers have studied privacy in different settings of technological advances [41, 45]. The first privacy theory emerged when newspapers started to publish personally intrusive articles and photographs[41]. This led to seclusion and non-intrusion theory of privacy that defined the user's privacy as "the right to be left alone" [45] or being free from intrusion [18]. As new technologies were introduced such as databases containing the personal information of the users [41] the information-related privacy concerns [38] emerged. To address these concerns researchers developed the control [46], limitation [16], and Restricted Access/Limited Control (RACL) [32] theories to enable users to control and limit their privacy while share information with others. In RACL theory, the user's privacy is implied as "a situation with regard to others [if] in that situation the individual…is protected from intrusion, interference, and information access by others." [42] The control, limitation and RACL theories assume a rigid definition of privacy, while in the current technological era the meaning of privacy changes based on the societal norms. To address this issue, Nissenbaum proposed the Contextual Integrity (CI) theory of privacy, [34] where privacy behaviors are affected by the context of the information sharing environment.

To implement the above theories, privacy policy languages were created based on the theories of limitation, control and RACL. The early privacy languages were either created by augmentation of access control languages or have the same structure of specifying policies as a set of access roles and information categories in a structured format like Extensible Markup Language (XML). Some well-known examples of such Languages are Platform for Privacy Preferences Project (P3P) [39], Enterprise Privacy Authorization Language (EPAL) [4], eXtensible Access Control Markup Language (XACML) [33], and Confab [19]. The early version of these languages lacked temporal modalities that were solved in the extended versions of them such as adding spatio-temporal attributes to XACML [27, 35, 43].

Another common formalism for privacy is based on transition systems where the policies are specified as action and state of information sharing. Formalizing privacy policies were based on the privacy-preserving and privacy-violating actions in the system. Also, in this formalism, the temporal characteristic of privacy was modeled using Linear Temporal Logic (LTL). Lu et al. [28] proposed a technique that translated the privacy specification of web services to LTL formulas. Then a Privacy Interface Automata (PIA) is presented to transform the messaging structure extracted from the web service business process execution language (WS-BPEL) into an automaton, creating their privacy policy model. Krishnan et al. [26] also proposed an approach to enforce privacy requirements using role-based access control and LTL. Their technique contains behavior automata that model the system behavior (gathering or using data) and an access control automata which enforce the privacy policies. Kouzapas et al. [25], combined the π-calculus and privacy calculus to verify privacy policies formally. Their framework has a type system to capture privacy related notations and a language for expressing the privacy policies. Grace et al. [17] proposed a model of user-centric privacy with a labeled transition system, which compares the cloud service privacy policies with the users' privacy preferences. However, while they provide customizable privacy preferences, they do not consider environmental variables in their model. Although this group specifies the privacy utilizing a formal semantic and considers the temporal modalities, the action based modeling of the system is not scalable [5].

The scalability issue in action based systems were addressed by Aucher et al. [5] that proposed to specify the privacy policies over the knowledge that the information sharing action exposes to the recipients of the information. In this model, privacy policy is specified as allowed and prohibited knowledge rather than actions, and different actions can result in different knowledge exchange. They used dynamic epistemic deontic logic (DEDL) as the foundation of their language. The authors define information sharing conditions as permitted or forbidden knowledge and the proposed language does not support temporal modalities. Also, Pardo et al. [36], presented a formal language for privacy policy, using epistemic logic for social network models. However, their formal privacy policy did not contain time features; later, [24, 37] extended [36] to include time characteristics to the privacy language by adding time interval and LTL which led to the creation of timed privacy framework for social media. Both frameworks used a social network model and privacy policies as properties for model checking [7] verification.

while a verity of implementation based on the theory of limitation, control and RACL continues to grow, another group of studies focused on the implementation of CI theory of privacy. Barth et al. [8] have utilized first-order logic and LTL to model the transfer of knowledge between agents during the information sharing activities that are governed by Nissenbaum's concept of *norms*. In this context, a positive norm is defined as a permission that allows information sharing activity and a negative norm prevents the information sharing activity. Further, implementation of CI was extended by DeYoung et al . [14] to include the notion of purpose and self-reference based on their Privacy Least Fixed Point (LFP) framework. The proposed framework resulted in the broader formalization of HIPAA and GLBA privacy laws.

The above approaches assume that the privacy policies will be created in a manner that are consistent with one another. However, privacy is dynamic in nature and as relationships and user's requirements changes it is required for privacy policies to change. These changes can result in privacy policy conflicts. Therefore, Breaux et al. [10] proposed Eddy that utilized CI. The goal of their research was to find privacy conflicts in multi-stakeholder privacy policies. In order to achieve that goal, natural language policies are translated to Description Logic (DL)[6] so it can be used in the formal reasoning process to investigate whether the policies are consistent. Eddy and many other frameworks that are based on CI theory are designed and develop based on the organizational privacy requirements which are not compatible with individual users privacy requirements.

For that reason, this paper defines and formalizes a user-centric privacy model utilizing CI theory. The next section describes the details on the methodology of our framework.

3 A FORMAL MODEL FOR USER-CENTRIC PRIVACY MANAGEMENT

This research extends the concept of contextual integrity [8] to provide mathematical models and algorithms that enables the creations and management of privacy norms for individual users. The extension includes the augmentation of environmental variables, i.e. time. date, etc. as part of the privacy norms, while introducing an abstraction and a partial relation over information attributes.

The proposed framework is based on two sets of formal models: 1- User's Information Sharing Model (UISM) that represents the information sharing activities in real-time, and 2- Privacy-Preserving Model (PPM) that formally specifies the user's privacy requirements. Finally, the privacy verification is performed by mapping each action in UISM to its corresponding action in the PPM. In the case of not being able to map an action a privacy violation is detected and reported to user to get confirmation. The rest of this section explains the above concepts in details.

3.1 User Information Sharing Model (UISM)

UISM is designed based on the formal definition of entities that construct *Information Communication* machanism based on agent. This is done to model user's information sharing behavior with the recipients, which are defined as agents [5, 8]. Hence, P is defined as a set of agents that are the recipient of the information sent from the user. For example, Alice and Bob are agents that the user shares information with them. In addition, T is a set of attributes that defines the information shared with $p \in P$ such as "home address" or "credit card number".

From the above definitions, a knowledge state κ is defined as a set of tuples of the form $(p, \{t_1, \ldots, t_k\})$, which describes the attributes $t_i \in T$ that is shared with an agent p. For example $(Alice, \{\text{home address, credit card number}\})$ means that Alice knows about the "home address" and "credit card number". As a result, if agents have no knowledge about the user then κ can be an empty set. Therefore, the absence of tuples for p indicates that the agent p possesses no information about the user, i.e., the elements $(p, \emptyset) \notin \kappa$. Thus, κ can be defined as follows where P is a set of agents and $\mathcal{P}(T)$ is the power set of attributes,

$$\kappa \subseteq \emptyset \cup (P \times (\mathcal{P}(T) \setminus \emptyset))$$

For brevity we use $\widetilde{t}$ to represent an element of $\mathcal{P}(T)$, i.e., $\{t_1, \ldots, t_k\}$.

In the proposed framework the user can perform two commands to share or stop sharing information with an agent. Each share, *sh*, or stops sharing, *st* command results in a communication action which we define as a triple $(a, p, \widetilde{t})$, where $a \in \{sh, st\}$. For example, when user intend to share his/her home address with Alice, the following communication action has to be performed: $(sh, Alice, \{\text{home address}\})$. Thus, all possible communication actions can be defined as

$$Act = \{sh, st\} \times P \times (\mathcal{P}(T) \setminus \emptyset)$$

Based on the entities defined so far, the user's behavior model could be defined by a transition system where each state represents the information shared with the agents. Further, each transition is triggered by the communication action performed by the user.

Definition 1. ***(The User Information Sharing Model (UISM)*** *Let $UISMM = (K, Act, \rightarrow, \kappa_0)$ be a 4-tuple transition system where:*

- *K is a finite set of knowledge states κ.*
- *$\kappa_0 \in K$ is the initial state $\kappa_0 = \emptyset$ (no initial disclosures).*
- *Act is a set of communication actions.*
- *$\rightarrow \subseteq K \times Act \times K$ is a transition relation, transform the system state with actions $(a, p, \widetilde{t})$ as follows:*
  - *$\kappa \xrightarrow{(sh,p,\widetilde{t})} \kappa'$, where $\kappa' = \kappa \cup \{(p, \widetilde{t})\}$,*
  - *$\kappa \xrightarrow{(st,p,\widetilde{t})} \kappa'$, where $\kappa' = \kappa \setminus \{(p, \widetilde{t'}) \mid \widetilde{t} \cap \widetilde{t'} \neq \emptyset\}$.*

It is important to note that the proposed model differentiates between the sequentially/simultaneously sharing of t_1 and t_2 with p. The sequential sharing results in $\kappa_1 = \{(p, \{t_1\}), (p, \{t_2\})\}$ while the simultaneous sharing results in $\kappa_2 = \{(p, \{t_1, t_2\})\}$. In κ_2 if the action $(sh, p, \{t_1, t_2\})$ occurs $(p, \{t_1, t_2\})$ is added to the new knowledge set. Thus a state contains all the three tuples $\kappa_3 = \{(p, \{t_1\}), (p, \{t_2\}), (p, \{t_1, t_2\})\}$. On the other hand, the performance of the stop command (st, p, t_2) on κ_3 will result in deletion of all the information attribute that contained t_2 from $\kappa' = \{(p, \{t_1\})\}$.

For the sequential information sharing model, we consider a scenario where user first shares his "GPS" information with Alice, second shares his "home address" with her, and third shares his billing information which is a combination of {home address, credit card number} with Alice. If the commutation action of stop sharing "home address" with Alice occurs then all the tuples that contain

"home address" like billing information will be removed from the state.

3.2 Privacy-Preserving Model (PPM)

The Privacy-Preserving Model is designed to manage and govern user's information sharing activities at run-time. Therefore, based on the proposed UISM in the previous section, PPM model is required to govern the transitions between knowledge states according to the norms that the *user* specifies.

Since in a user-centric approach is inefficient to define a separate privacy norm for each ρ (role) and τ (attribute type), the proposed model abstracts these two elements. This abstraction allows to have the same information disclosure norms with a set of agents or disclose a collection of attributes in a similar manner. For example, the user could share her current location with all transportation applications, or the user could share her credit and debit cards' numbers with her close family members. The following section describes the structure of the abstractions.

3.2.1 Abstractions and Conditions. Let $\mathcal{T}$ be a set of *attribute types* and let AT be a partial map $AT : \mathcal{P}(T) \mapsto \mathcal{T}$. That is, AT maps $\widetilde{t}$ to an attribute type $\tau \in \mathcal{T}$. We can impose a partial order $\preceq$ on τ based on the subset relation between AT's domain elements $\widetilde{t}$. We say that $\tau_1 \preceq \tau_2$ if there are exist $\widetilde{t_1}$ and $\widetilde{t_2}$ such that $AT(\widetilde{t_1}) = \tau_1$, $AT(\widetilde{t_2}) = \tau_2$ and $\widetilde{t_1} \subseteq \widetilde{t_2}$.

Figure 1a, and 1b demonstrate an example of hierarchy structure and some attributes and attribute types in that structure. The dashed line represents the mapping between an attribute and its type and the solid lines depict the order relation between the attribute and types.

Similar to [8] that defines the concept of role abstraction, we define a set of *agent roles* $\mathcal{R}$ that can be assigned to an agent p. An agent can be assigned to multiple roles and roles are partially ordered based on their implication relation of their semantics.

In this paper, the partial order $\preceq$ on $\mathcal{R}$ is predefined as an input to the model, such that the role, ρ_1, "close friend" implies the role, ρ_2, "friend", i.e., $\rho_2 \preceq \rho_1$. The order between roles implies the amount of relative privacy restriction of them where $\rho_2 \preceq \rho_1$ means that ρ_2 is more restrictive compared to ρ_1.

In this approach each agent must be associated with at least one role. Thus, we define the agent role as a function AR that maps an agent to a nonempty set of roles: $AR : P \mapsto \mathcal{P}(\mathcal{R}) \setminus \emptyset$. When role ρ is assigned to an agent p, then the systems adds additional roles that related to ρ through $\preceq$. In other words, the set of roles for p should be closed under $\preceq$. For example, if the agent p is assigned the role "close friend" ρ_1, then the system adds "friend" role ρ_2 to p as well, resulting in $AR(p) = \{\rho_1, \rho_2\}$.

For brevity to show the roles and information attributes that have a common child but are not in a partial relation with each other we use the $< child >$ notation as follow:

(1) $\rho_1 < p > \rho_2 = \exists p \in \mathcal{P} : \rho_1 \in AR(p) \wedge \rho_2 \in AR(p) \wedge \rho_1 \not\preceq \rho_2 \wedge \rho_2 \not\preceq \rho_1$
(2) $\tau_1 < t > \tau_2 = \exists \widetilde{t} \in \mathcal{P}(T) : AT(\widetilde{t}) \preceq \tau_1 \wedge AT(\widetilde{t}) \preceq \tau_2 \wedge \tau_1 \not\preceq \tau_2 \wedge \tau_2 \not\preceq \tau_1$

Using these abstractions the user can define ***access permissions*** $\mathcal{A}$ as a subset of $\mathcal{R} \times \mathcal{T}$ such that if an element $(\rho, \tau) \in \mathcal{A}$ then all agents with role ρ are allowed to access attributes with type τ.

The above abstractions of roles and information attributes provide a better flexibility in defining privacy norms. However, this definition is not complete yet, as it does not take into the consideration the environmental conditions where the information is disclosed to the recipients and has no sensitivity over the patterns and sequence of the information disclosure. Imagine, user is interested in restricting access of agents in ρ role to its attribute type τ to a particular time interval during a work day. Moreover, the user might allow only up to two (ρ, τ) accesses per such interval.

In order to overcome this limitation, our formalism introduces the logic for environmental conditions ψ and temporal conditions φ to the definition of the privacy norm. In this model, environmental conditions are represented a set of variables V, where each $v \in V$ describes the state of an environment such as system's time, day and other attributes. Then, V is partitioned into subsets V_i by variables' type like integers, boolean, reals and so on. It is assumed that each type has a set of predicates $Pred_i$ and set of syntax rules to construct such predicates from the variables and non-logical symbols, e.g., constants. Then an environmental condition (Ψ) is expressed as a propositional logic over those predicates and variables, i.e., $v \in V_i$, $pred_i \in Pred_i$ as follows:

$$\psi ::= \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid pred_i, \forall V_i \in V$$

While $Pred_i$ could be produced by an arbitrary complex yet decidable theory for the data type such as Presburger arithmetic for integers, we argue that less complex theories could be adequate[3]. For example, for integer environmental variables V_I and boolean V_B environmental variables the following grammar could be sufficient to express basic and easily comprehensible predicates $pred_i$:

$$pred_I ::= v \leq n \mid v < n \mid v == n, v \in V_I, n \in \mathbb{Z}$$
$$pred_B ::= v \mid true \mid false,\ v \in V_B$$

The next entity that is defined as part of the privacy norm is the temporal condition φ. In order to keep the conditions flexible and generic, we utilize temporal logic expressions to describe temporal features of the privacy requirements. While Linear Temporal Logic (LTL) is very popular in expressing broad range of liveness conditions, they are difficult to read and understand. Utilizing LTL requires a strong mathematical background, and is cumbersome for an average system modeler to implement. Further, for the purpose of defining temporal conditions in privacy norm a simplified grammar will suffice, i.e define the precedence of two communication actions or a constant occurrence a communication actions can be sufficiently defined by the concatenation and Kleen star operations over $\mathcal{A}$ (the alphabet):

$$\varphi, \phi ::= (\rho, \tau) \mid \varphi \cdot \phi \mid \varphi^*, (\rho, \tau) \in \mathcal{A}$$

The Φ notation is used to represent a set of φ, in which each φ for a given role ρ, can be expressed as a regular expression that allows sharing attributes of type τ_2 after the sharing of attributes of type τ_2 as follows:

$$\varphi = \mathcal{A}_1^* \cdot ((\rho, \tau_1) \cdot \mathcal{A}_1^* \cdot (\rho, \tau_2))^* \cdot \mathcal{A}_1^*$$

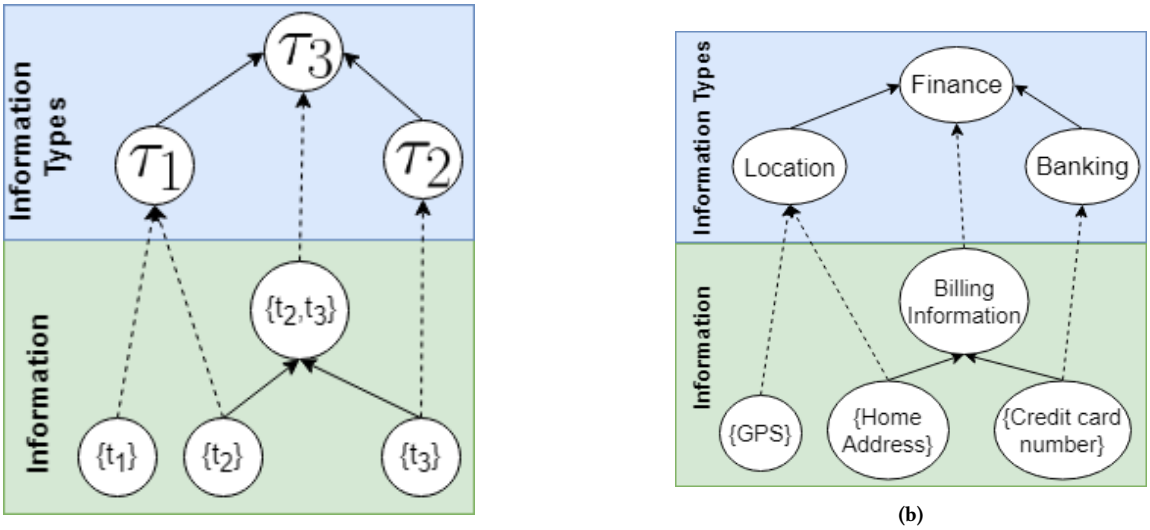

Figure 1: (a) An example of the partial order of the attributes and attribute types where the top layer show the attribute types and the bottom layer show the information themselves. (b) t_1 =GPS information, t_2 = home address, and t_3 = credit card number. The middle layer represents the information that are used together for example the credit card number and the home address go together for billing information that is a considered as financial type.

Here $\mathcal{A}_1 = \mathcal{A} \setminus \{(\rho, \tau_1), (\rho, \tau_2)\}$ In addition, the repetition of an event up to a constant k times could be expressed with the following formula, where the power operator describes the number of times a regular expression should be repeated.

$$\varphi = \mathcal{A}_2^*((\rho, \tau) \cdot \mathcal{A}_2^*)^k$$

where $\mathcal{A}_2 = \mathcal{A} \setminus \{(\rho, \tau)\}$.
Now that we have defined each elements in the privacy norm, the next section describes the formal specification of the privacy norm and techniques to ensure the consistency of the privacy requirements.

3.2.2 Norms and their Consistency. In this research, norms are the formal definition of user's privacy requirements that are used to govern user's information sharing behavior. In order to minimize the risk of unwanted information sharing, we assume that if an action is not explicitly defined as part of the user's privacy policies then it is forbidden. Therefore, the only type of norms that the user defines are positive norms, i.e., *allowed* norms.
In this context norm is formulated as a relation between access permission, environmental, and temporal conditions. Hence, norm is represented as a tuple $((\rho, \tau), \psi, \varphi,)$, where $(\rho, \tau) \in \mathcal{A}$ and $\psi \in \Psi$, $\varphi \in \Phi$. The first element of the tuple represents the privacy policy, while the second and the third elements of the tuple describe the conditions under which the transfer of information should occur. The set of such is referred to as a set of norms $\mathcal{N}$.
The set $\mathcal{N}$ has the uniqueness property, that is, only one tuple with the given (ρ, τ) values is allowed in the set. However, the uniqueness property is not sufficient to ensure the consistency of the privacy norms due to the partial relations that exist among the roles and attribute types. Thus, in order to utilize $\mathcal{N}$ for privacy management and detection of information disclosure, a consistency check is required. The Table 1 demonstrates a detailed explanation with examples of the different possible cases of role and attributes types that two norms can have during consistency checking. The row headers show the roles and the column headers show the attribute types. The cells in gray are the example of their above conditions.

Definition 2. ***(Consistent Norms)*** *Two norms $n_1 = ((\rho_1, \tau_1), \psi_1, \varphi_1)$ and $n_2 = ((\rho_2, \tau_2), \psi_2, \varphi_2)$ are consistent when one of the four consistency conditions holds:*

C1. $\nexists p \in \mathcal{P} : \rho_1 \in AR(p) \wedge \rho_2 \in AR(p)$, that is, the norms defined for the roles with no common agents. (Table 1 row G)

C2. $\nexists \widetilde{t} \in \mathcal{P}(T) : AT(\widetilde{t}) \preceq \tau_1 \wedge AT(\widetilde{t}) \preceq \tau_2$, that is, norms are defined for attribute types with no common information attribute.(Table 1 column 5)

Before defining the last two conditions of consistency, we propose some limitations over the access permission and sequencing conditions of the privacy norms. Since both of these elements are defined for a specific roles and attribute type parameters, the first restriction is defined over the roles so that the same role should be used in the access permission and the sequencing condition of a norm. In the absence of this restriction, it is possible to create two norms that have a consistent sequencing condition but inconsistent access permission or vice versa. In addition, this restriction enforces a constant role across the regular expression of the sequencing condition that reduces the regular expression's complexity by eliminating the need for a homomorphic function over the roles. The second restriction is defined over the attribute types, $\forall \tau \in \varphi \quad \tau_i \not\preceq \tau_j \quad 0 \leq i, j \leq n$ (An attribute type and its children

Table 1: The possible consistency cases based on the roles and information attribute types relations and the constrains over the conditions that result in consistency. The notations Fr=Friends, BFr=Best Friends, $CoWr$=Co-Workers, Fml=Family, Loc=Location, Fin=Finance, $Hlth$=Health, and $Bank$=Banking information

		1	2	3	4	5
		$\tau_1 < \tau_2$	$\tau_2 < \tau_1$	$\tau_1 = \tau_2$	$\tau_1 < e > \tau_2$	$\tau_1 < none > \tau_2$
		$Loc < Fin$	$Loc < Fin$	$Loc = Loc$	$Fin < Loc > HLth$	$Loc < none > Bank$
A	$\rho_1 < \rho_2$	$c_2 \Leftrightarrow c_1$ $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	$c_2 \implies c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	$c_2 \implies c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	$c_2 \implies c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	True
B	$Fr < BFr$	Share Loc with Fr when c1 an s1, share Fin with BFr when c2 and s2. Fin should be guarded the same or better, $c_1 \implies c_2$, $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$. BFr can have less restrictive access, $c_2 \implies c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	Share Fin with Fr when c1 and s1, share Loc with BFr when c2 and s2. Fin should be guarded the same or better, $c_2 \implies c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$. BFr can have less restrictive access $c_2 \implies c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	Share Loc with Fr when c1 and s1, sare Loc with Bfr when c2 and s2. Loc should be guarded at least the same way, $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$. BFr can have less restrictive conditions, $c_2 \implies c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	Share Fin with Fr and Health with BFr (or vice versa) which can share Loc. Loc should be guarded at least the same way $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$. BFr can have less restrictive condition, $c_2 \implies c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	Since Loc and Bank are incomparable then those norms should always be consistent.
C	$\rho_1 = \rho_2$	$c_1 \implies c_2$ $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$	$c_2 \implies c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	False	$c_2 \Leftrightarrow c_1$ $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	True
D	$Fr = Fr$	Share Loc with Fr when c1 and s1, share Fin with Fr when c1 and s2. Fin should be guarded the same or better way $c_1 \implies c_2$, $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$. Fr should have at least the same access, $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$.	Share Fin with Fr when c1 and s1, share Loc with Frien when c2 and s1. Fin should be guarded the same or better way, $c_2 \implies c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$. Fr should have at least the same access $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	There should be only one rule for the same role and attribute type - the uniqueness property	Share Fin with Fr when c1 and s1, share Health with Fr when c2 and s2, which can share the same attribute Loc. Loc should be guarded at least the same way $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$. Fr should have the same access $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	Since Loc and Bank are incomparable then those norms should always be consistent.
E	$\rho_1 < p > \rho_2$	$c_1 \implies c_2$ $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$	$c_2 \implies c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	$c_2 \Leftrightarrow c_1$ $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	$c_2 \Leftrightarrow c_1$ $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	True
F	Fr Anna CoWr	Share Loc with Fr when c1 and s1, share Fin with CoWr when c2 and s2, which have Anna as a common agent. Fin should be guarded the same or better way $c_1 \implies c_2$, $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$. Fr and CoWrk should have at least the same access to Loc $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_2) = \mathcal{L}(s_1)$, since they share an agent.	Share Fin with Fr when c1 and s1, share Loc with CoWrk when c2 and s2, which have Anna a common agent. Fin should be guarded better than Loc $c_2 \implies c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$. Fr and CoWrk should have at least the same access to Loc $c_2 \Leftrightarrow c_1$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$, since they share an agent.	Share Loc with Fr when c1 and s1, share Loc with CoWrk, when c1 and s2, which have Anna as a common agent. Loc should be guarded the same way $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$. Fr and Cowrk should have the least the same access to Loc, $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$, since they share an agent.	Share Fin with Fr when c1 and s1, share Health with CoWrk when c2 and s2, which have Anna as a common agent. Loc should be guarded at least the same way $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$. Fr and CoWrk should have the same access to Loc $c_1 \Leftrightarrow c_2$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$, since they share an agent.	Since Loc and Bank are incomparable then those norms should always be consistent.
G	$\rho_1 < none > \rho_2$	True	True	True	True	True
H	Fr, none, Fml	Since Fr and Fml are incomparable then those norms should always be consistent.	Since Fr and Fml are incomparable then those norms should always be consistent.	Since Fr and Fml are incomparable then those norms should always be consistent.	Since Fr and Fml are incomparable then those norms should always be consistent.	Since Fr and Fml are incomparable then those norms should always be consistent.

are not allowed to exist in the same regular expression). This restriction ensures that all the communication actions are inspected not only for the super-type τ, that is explicitly inferred from the communication action, but also for all the children of τ that will be implicitly revealed by that communication action. Without this restriction, it is possible to create a regular expression that allows for sharing an attribute type and its children consecutively while it is not taking into the account that the children are shared more than once.

Further, the comparisons of the access permission component of the norms are conducted based on the partial relations that exists over the roles and attribute types. In addition, the comparison between the environmental conditions is implemented based on the Boolean algebra. To examine the sequencing conditions for consistency, we need to compare the regular expressions. the comparison of two regular expressions is not possible if they do not share the same alphabet. Therefore, we need to introduce a mechanism that projects the language of one regular expression to the other one and brings the regular expressions to a common alphabet.

Definition 3. ***(Projection of the Language)*** *Let φ_1 and φ_2 have the following symbols to be tracked:*

$$\varphi_1 = \{(\rho, \tau_1), (\rho, \tau_2), \ldots, (\rho, \tau_k)\}$$

$$\varphi_2 = \{(\rho', \tau_1'), (\rho', \tau_2'), \ldots, (\rho', \tau_n')\}$$

We define $\widetilde{\varphi_1} = \mathcal{L}_{\downarrow}(\varphi_1)_{\varphi_2}$ as the projection of φ_1 on φ_2 where $\mathcal{L}_{\downarrow}$ receives a regular expression and maps it to another one. To achieve a similar language to compare φ_1, φ_2 we traverse over the attribute types. For each attribute type, we check for its children or another attribute type that has a common child in the other regular expression and add the children or the common child to a set in a map. After traversing over all the attribute types in both φ_1, φ_2 to substitute the uncommon parts, we generate all the possible substitution for attribute type τ_i exist in the map. The substitution for τ_i for reaching a common language is a disjunctive regular expression. The disjunctive regular expression is generated as follows. Let sub be a set of all τ_i children and common children that have been found in the other regular expression. We define $\widetilde{sub} = \mathcal{P}(sub) \setminus \emptyset$. For each $s \in \widetilde{sub}$ we generate all the permutations of elements of s and add them to the regular expression with disjunction operator. For example, $sub = \{\tau_a, \tau_b\}$ then $\widetilde{sub} = \{\{\tau_a\}, \{\tau_a\}, \{\tau_a, \tau_b\}\}$ and the result of the regular expression that is used for substitution is $\tau_a|\tau_b|\tau_a\tau_b|\tau_b\tau_a$. After reaching the same alphabet, the consistency of the regular expressions can be decides based on the norms' access permission.

C3 . $\rho_1 < \rho_2$ and either $\tau_1 \preceq \tau_2$ or $\tau_2 \preceq \tau_1$ then $\psi_1 \implies \psi_2 \wedge \mathcal{L}_{\downarrow}(\varphi_1)_{\varphi_2} \subseteq \mathcal{L}_{\downarrow}(\varphi_2)_{\varphi_1}$, that is, n_2 is for a specialized role ρ_2 of ρ_1 and its attribute type τ_2 encompasses τ_1 or vise verse then environmental condition of ψ_2 should be the same or less restrictive than of ψ_1 and its regular expression φ_2 should describe the same or less restricted projected language than of φ_1.(Table 1 row A,C and columns 1,2,3)

C4 . $\rho_1 < p > \rho_2$ *or* $\tau_1 < t > \tau_2$ *then* $\psi_1 \Leftrightarrow \psi_2 \wedge \mathcal{L}_{\downarrow}(\varphi_1)_{\varphi_2} = \mathcal{L}_{\downarrow}(\varphi_2)_{\varphi_1}$*. If there is at least one agent that can be assigned to both unrelated roles or an information attribute that share a common child then the environmental conditions and the projected language of the regular expressions must be equivalent.(Table 1 row E and columns 4)*

3.2.3 Policy Compliance Verification. The set of norm $\mathcal{N}$ defines a Privacy-Preserving Model, (PBM) which describes compliant information communication actions at the level of attribute type and agent role abstraction levels. The knowledge states of PBM are consists of tuples (ρ, τ), which indicate that at least one agent with ρ role know about attribute represented by τ. The transitions represent the abstracted communication actions $\widehat{Act}$ from $\{sh, st\} \times \mathcal{R} \times \mathcal{T}$ guarded by conditions Φ and Ψ defined in $\mathcal{N}$.

Definition 4. ***(Privacy-Preserving Model)*** *is a set of observers over norms* $\mathcal{N}$ *where each observer is a tuple of* $(\widehat{K}, \widehat{Act}, c, m)$ *representing* $n_i = ((\rho, \tau), \psi, \varphi) \in \mathcal{N}$ *where* $\widehat{K} = (\rho, \tau)$*,* $c = \psi$ *is the pre-condition and m is a monitor representing* φ *regular expression. The transition* $\widehat{Act}$ *is given to Monitor m to update the state of the monitor.*

3.2.4 Verification. To ensure that the user's behavior is compliant with the privacy policy, we need to map the current state and the next state of user's behavior model to the privacy preserving behavior model.

Definition 5. ***(Mapping from user behavior to privacy preserving domain)*** *Let* $MS : K \rightarrow \widehat{K}$ *be a surjective function, where* $MS(p, \widehat{t}) = \{(\rho, \tau) | \rho = AR(p), \tau = AT(\widetilde{t})\}$ *and* $MT : Act \rightarrow \widehat{Act}$ *where:*

$$MT(a, p, t) = \{(\rho, \tau) | \rho \in AR(p) \wedge \tau \in AT(t)\} \; if \; a = sh$$

In the case that there is no mapping for the next state in the PPM, the communication action that triggered that transition will be reported to the user as disclosing.

Definition 6. ***(Valid user behavior)*** *Let user behavior system be at state k that maps to* $\widehat{k}$ *in the privacy preserving behavior model and the action* (sh, p, t) *happens. If* $MP(p, t)$ *exists, and the environmental variables satisfy* ψ *and* $m(MS(a, p, t))$ *is in the final state then the communication action Act is valid.*

The goal of privacy rules is to prevent the user from entering into a privacy violating states.

After reporting a privacy-violating action the user can ignore it and the framework allow the information sharing to happen. All this communication happens through the user interface of the framework. The next section provides implementation details of the framework's components.

4 IMPLEMENTATION

As a proof of the concept, we prototyped the proposed framework in the Java programming language [1]. Figure 2 depicts a diagram of the implementation's architecture. The blue components show the libraries and technologies used in the proposed framework. The proposed framework is modularized into three layers:(1) User interface layer which takes the user's intentions in a structured format,(2) Translation layer which translates the frameworks from UI to privacy norms and formal notation, (3) Verification layer that evaluates norms consistency and compliance of the information sharing action with privacy norms.The following sections describe the implementation details of each of the components in each layer.

[1]https://github.com/wxyzabc/UserCentricPrivacy

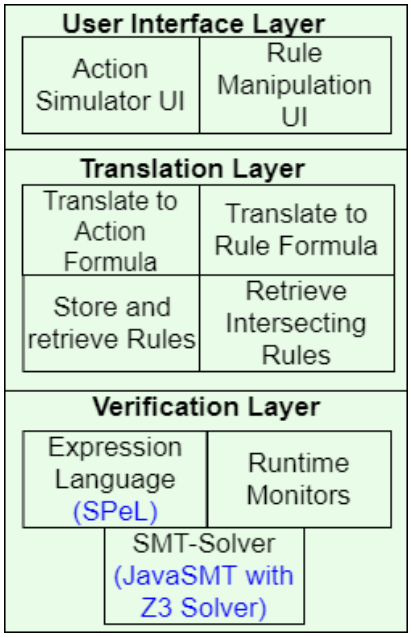

Figure 2: The architecture of user-centric privacy framework.

4.1 User Interface Layer

The user interface (UI) layer facilitates interactions between the user and the proposed framework. Through the UI the user can add and view the existing privacy norms and get privacy violation reports. The UI is designed to conceal the complexity of the underlying formalism and verification from the user. The UI hides the complexities by allowing the users to express their privacy intentions as a structured input. Using the UI the user can select the role and attribute type from a drop-down list. To create the environmental conditions, the user can provide arbitrary inputs for environmental variables or choose between predefined conditions e.g., daytime, nighttime, weekends. Also, the user can specify the desired information sequence in the form of precedence or repetition templates like "X happens after Y" or "X happens k times". These templates will be translated to sequencing conditions.

4.2 Translation Layer

The translation layer receives the structured input from the UI and translates it into formal notation. The formal notations and maps described in the methodology section can be implemented as tables in a database. The norm are stored in the norms table where the table attributes are the role, attribute type, the environmental conditions, and the DFA state of the sequencing conditions. The primary key of the norms table is the pair of (ρ, τ). The system

queries the database to retrieve the norms in order to either verify an action or check the consistency of a new norm. To evaluate each action with the attribute t and the agent p, norms that have roles where $\rho = AR(p)$ and attribute type $\tau = AT(t)$ will be retrieved from the norm table and sent to the verification layer.

4.3 Verification Layer

This layer verifies the information sharing actions compliance with the privacy norms and the consistency of a new norm with existing norms. If an information sharing action violates the privacy norms or a new norm causes inconsistency, then this layer sends a violation report to the UI to inform the user. The user can ignore the violation caused by the information sharing action and allow the information to be shared. With an inconsistent norm, the user has to change the new norm so that it will be consistent with other norms. The rest of this section describes the verification method of information sharing actions and privacy norms in more detail.

4.3.1 Verification of norms for Inconsistency. When a new norm is created, the framework checks the consistency of the new norm with the existing norms. Based on the consistency constraints in section 3.2.2 the framework first ensures that the new norm access permission does not exist in the database. Then the new norm's environmental conditions are checked for consistency. The framework parses the string of the environmental conditions and changes them to SMT solver formulas. Then the SMT solver needs to prove that the implication or equivalency relation holds and it is always valid. Validation assessment of formula f by SMT solvers is done by proving that $\neg f$ is unsatisfiable, hence f always evaluates to true. By proving that there is no combination of variables that satisfy $\neg f$ it can be concluded that f is a tautology. In a case that the solver finds a solution to $\neg f$, the user is asked to change the inconsistent new norms. Further, since efficiency is important in real-time systems, we need to assign a time limit for the solver. If the solver times out or returns UNKNOWN the user will be notified. Finally, if the norm was consistent it will be added to the database. The implementation of the proposed framework utilizes JavaSMT [23] with the Z3 solver version 4.3.2 [13] for consistency checking over the environmental variables and "brics" library version 1.12-1 [31] for sequencing conditions.

4.3.2 Verification of Actions for violation. For each action (sh, p, t) , the framework finds the attribute type of t and the role of p. Then the privacy norms tables are queried to find the norms with the access permission $(AR(p), AT(t))$ as their primary key. If the query returns no results, it means that no norm allows sharing information t with agent p. However, If the query returns results, it indicates that there exists a mapping from a state in UBM to a state in the PPM. Then the framework checks for the satisfaction of the environmental conditions and sequencing conditions before taking the transition to the mapped state.

Since the norm conditions are dynamic, they cannot be hard-coded in the verification engine. Therefore to check the environmental variables a mechanism is needed to enable the verification engine to handle change in the conditions. Therefore, the conditions are formed and evaluated at run-time based on the stored environmental constraints in the database. For the implementation of such a mechanism that allows for dynamic manipulation and evaluation of conditions, the Expression Languages (EL) can be used. EL receives an object and a logical expression as a string and evaluates whether the object properties satisfy the expression or not. In our implementation,the current snapshot of the environment is given to the EL as the input object that has the environmental values and the EL expression string is the environmental constraints of the retrieved privacy norms. This framework employs Spring Expression Language (SpEL) [1] as the EL library. EL only checks for the satisfaction of the environmental conditions and if they are not satisfied then the transition guard is not satisfied. Therefore, the action violates the privacy model. However, if the environmental conditions are satisfied then we check for the satisfaction of sequencing conditions.

Sequencing conditions implemented as run-time monitors from the regular expressions stored in the database. A run-time monitor is a deterministic finite automaton (DFA) that is created based on a regular expression. The DFA representing the sequencing condition has a pointer to its current state and changes its state with the occurrence of information sharing actions. If the new state in the DFA monitor is not a final state, then the action is not valid, and the system reports the violation to the user. Different libraries exist for creating run-time monitors such as AspectJ, but the monitors created by them are static. Therefore, a change in one of the regular expressions demands a reset in all the monitors. In the proposed framework the regular expressions are dynamic, and changing a regular expression only causes a reset in the corresponding DFA. Another method for implementing the sequencing conditions is to store a history of information sharing actions; however, with each information sharing action, the history will grow, and to potentially infinite size. With the run-time monitors, the number of the DFAs are constant and equal to the number of the norms with sequencing conditions. Algorithm 1 shows the general steps taken to implement the information sharing action verification process.

Algorithm 1: Action verification algorithm.

```
Input: CA (Communication action)
Output:Boolean value indicating the verification result.
norms=[]
roles=CA.recipient.getRoles()
types=CA.informationAttribute.getType()
for r in roles and t in types do
    norms.append(getnorms(r,t))
if (norms.size> 0) then
    for j in norms do
        if !(j.evalEnvironmentalCondition (CA.environment))
          then
            return false
        else
            if !(j.evalSequencingCondition(CA)) then
                return false
    return true
return false
```

Considering the above implementation, in the next section we discuss the performance evaluation of the proposed framework.

5 PERFORMANCE EVALUATION

The proposed framework is designed for user-centric applications; therefore, it should have acceptable performance on smart devices such as smart-phones, internet of things devices and etc. The main challenge in this area is that usually, these devices have low memory and computational power. Since detection of privacy violations in such applications supposed to be real-time, a framework with a substantial performance overhead cannot deliver the desired results. Therefore, our implementation was tested for performance evaluation on a Raspberry Pi model B with 700 MHz CPU, 512 MB RAM and running Raspbian 4.9 operating system. As well as a PC with 3.0 GHz AMD Phenom II X4 945 processor, with 8 GB of memory and Windows 7 operating system. The privacy policy created for this test contained 81 privacy norms over 12 attribute types and 16 roles which 8 of them have nonempty intersections with another groups.

Table 2 shows the results of the information sharing action verification performance evaluation. The number in each column indicates the average verification response time for each part of the a privacy norm. The average was computed for 20 information sharing actions which half were privacy violating actions and the other half were non-violating actions. Also, notice that the performance of the action verification depends on the performance of underlying database software and expression language library. In the implementation of our framework, we used MariaDB version 10.2 database and SpEL 3.1.0 as the EL library.

Table 2: Action verification performance evaluation results. The columns show the response time for Access Permission (AP), Environmental Conditions (EC), Sequencing Conditions (SC).

Machine	Action Verification		
	AP	EC	SC
PC	1.5 ms	0.5 ms	3.5 ms
Pi B	39 ms	6 ms	540 ms

The average time for the consistency check performance evaluation on the PC was 39 ms and for Raspberry Pi model B was 849 ms. Also, notice that the performance of this consistency checking depends on the performance of the underlying solver and the domain size of the environmental variables (since the solvers are faster when the search domain is smaller). For example, in our implementation, the norms time conditions were specified as (hours×100+minutes) and time intervals could be subsets of each other. Table 3 shows the SMT-solver performance for constraints with 5,10,20,50,100, and 500 environmental variables. The over-head of bric library for language sunset and equivalency is around 7ms on average. However, the projection algorithm is the bottleneck since it computes the permutation of the information types that are needed for substitution in the regular expression. Due to this drawback the framework limits the number of children for each attribute to 5 children.

Table 3: Performance of consistency checking for Environmental Variables

Number of Variables	5	10	20	50	100	500
Implication time (ms)	26	28	30	40	35	66
Equivalency time (ms)	32	34	35	46	41	67

6 CONCLUSION AND FUTURE WORKS

Administrating and managing users' privacy is a major challenge in the digital age. Privacy has a different meaning to different users depending on their personality, age, social status, cultural background, and many other factors. However, current privacy management systems cannot address these privacy needs adequately since they are not designed based on the users' privacy perspectives. Therefore, the lack of user-centric privacy management tools and algorithms limits users' ability to have control over their data sharing activities and puts unaware users at risk of information disclosure. In order to overcome these limitations, the proposed framework provides a privacy formalism and verification engine to specify and model privacy from the user's perspective. Moreover, as a proof of concept, a framework was implemented and tested based on the described formalism. In the proposed model, the contextual integrity theory has been customized to address the privacy needs of individual users. Further, the user-centric privacy framework is meant to be utilized in the new generation of smart devices and IoT, which compared to servers and general purpose computers, have lower memory and computational power. These limitations justify the use of regular expressions instead of Linear Temporal Logic (LTL) in our paper since empirical evidence [9] shows that the evaluation of the regular expressions has significantly less overhead compared to LTL.

The future work will eliminate the current user interface and user's privacy norms will be generated automatically utilizing text analysis, speech recognition, and AI algorithms that can infer user's privacy policies based on the user's relationships and information sharing behaviors.

ACKNOWLEDGMENTS

The authors would like to thank National Science Foundation for its support through the Computer and Information Science and Engineering (CISE) program and Research Initiation Initiative(CRII) grant number 1657774 of the Secure and Trustworthy Cyberspace (SaTC) program: A System for Privacy Management in Ubiquitous Environments.

REFERENCES

[1] [n. d.]. Spring Expression Language. https://docs.spring.io/spring/docs/3.0.x/reference/expressions.html. Accessed: August 31, 2018.
[2] Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. 2015. Privacy and human behavior in the age of information. *Science* 347, 6221 (2015), 509–514.
[3] Alessandro Acquisti and Jens Grossklags. 2005. Privacy and rationality in individual decision making. *IEEE security & privacy* 3, 1 (2005), 26–33.
[4] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. 2003. Enterprise privacy authorization language (EPAL). *IBM Research* (2003).
[5] Guillaume Aucher, Guido Boella, and Leendert Van Der Torre. 2011. A dynamic logic for privacy compliance. *Artificial Intelligence and Law* 19, 2-3 (2011), 187.
[6] Franz Baader, Ian Horrocks, and Ulrike Sattler. 2008. Description logics. *Foundations of Artificial Intelligence* 3 (2008), 135–179.
[7] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. 2008. *Principles of model checking*. MIT press.

[8] Adam Barth, Anupam Datta, John C Mitchell, and Helen Nissenbaum. 2006. Privacy and contextual integrity: Framework and applications. In *Security and Privacy, 2006 IEEE Symposium on*. IEEE, 15–pp.

[9] Ilan Beer, Shoham Ben-David, and Avner Landver. 1998. On-the-fly model checking of RCTL formulas. In *International Conference on Computer Aided Verification*. Springer, 184–194.

[10] Travis D Breaux, Hanan Hibshi, and Ashwini Rao. 2014. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering* 19, 3 (2014), 281–307.

[11] Carole Cadwalladr and Emma Graham-Harrison. 2018. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian* 17 (2018).

[12] Federal Trade Commission et al. 2012. Recommendations for Businesses and Policymakers. *Washington, DC (http://www. ftc. gov/sites/default/files/documents/reports/federal-trade-commission-report-protecting-consumer-privacy-era-rapid-change-recommendations/120326privacyreport. pdf)* (2012).

[13] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

[14] Henry DeYoung, Deepak Garg, Limin Jia, Dilsun Kaynar, and Anupam Datta. 2010. Experiences in the logical specification of the HIPAA and GLBA privacy laws. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*. ACM, 73–82.

[15] Michael D Ekstrand, Rezvan Joshaghani, and Hoda Mehrpouyan. 2018. Privacy for All: Ensuring Fair and Equitable Privacy Protections. In *Conference on Fairness, Accountability and Transparency*. 35–47.

[16] Ruth Gavison. 1980. Privacy and the Limits of Law. *The Yale Law Journal* 89, 3 (1980), 421–471.

[17] Paul Grace and Michael Surridge. 2017. Towards a model of user-centered privacy preservation. (2017).

[18] Jamal Greene. 2009. The So-Called Right to Privacy. *UC Davis L. Rev.* 43 (2009), 715.

[19] Jason I Hong and James A Landay. 2004. An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM, 177–189.

[20] White House. 2015. Administration discussion draft: Consumer Privacy Bill of Rights Act of 2015. *Retrieved on November* 15 (2015), 2015.

[21] Rezvan Joshaghani, Michael D. Ekstrand, Bart Knijnenburg, and Hoda Mehrpouyan. 2018. Do Different Groups Have Comparable Privacy Tradeoffs? *At Moving Beyond a One-Size Fits All Approach: Exploring Individual Differences in Privacy, a workshop at the ACM Conference on Human Factors in Computing Systems (CHI)* (2018).

[22] Rezvan Joshaghani and Hoda Mehrpouyan. 2017. A Model-Checking Approach for Enforcing Purpose-Based Privacy Policies. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*. IEEE, 178–179.

[23] Egor George Karpenkov, Karlheinz Friedberger, and Dirk Beyer. 2016. JavaSMT: A unified interface for SMT solvers in Java. In *Working Conference on Verified Software: Theories, Tools, and Experiments*. Springer, 139–148.

[24] Ivana Kellyérová. 2017. A Real-Time Extension of the Formal Privacy Policy Framework. (2017).

[25] Dimitrios Kouzapas and Anna Philippou. 2017. Privacy by typing in the pi-calculus. *arXiv preprint arXiv:1710.06494* (2017).

[26] Padmanabhan Krishnan and Kostyantyn Vorobyov. 2013. Enforcement of privacy requirements. In *IFIP International Information Security Conference*. Springer, 272–285.

[27] Ki Young Lee, Aleum Kim, Ye Eun Jeon, Jeong Joon Kim, Yong Soon Im, Gyoo Seok Choi, Sang Bong Park, Yun Sik Lim, and Jeong Jin Kang. 2015. Spatio–temporal XACML: the expansion of XACML for access control. *International Journal of Security and Networks* 10, 1 (2015), 56–63.

[28] Jiajun Lu, Zhiqiu Huang, and Changbo Ke. 2014. Verification of Behavior-aware Privacy Requirements in Web Services Composition. *JSW* 9, 4 (2014), 944–951.

[29] Mary Madden, Aaron Smith, and Jessica Vitak. 2007. Digital Footprints: Online identity management and search in the age of transparency. (2007).

[30] Hoda Mehrpouyan, Ion Madrazo Azpiazu, and Maria Soledad Pera. 2017. Measuring Personality for Automatic Elicitation of Privacy Preferences. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*. IEEE, 84–95.

[31] Anders Møller. 2017. dk.brics.automaton – Finite-State Automata and Regular Expressions for Java. `http://www.brics.dk/automaton/`.

[32] James H Moor. 1997. Towards a theory of privacy in the information age. *ACM SIGCAS Computers and Society* 27, 3 (1997), 27–32.

[33] Tim Moses. 2005. Privacy policy profile of XACML v2. 0. *Oasis standard, OASIS* 2 (2005).

[34] Helen Nissenbaum. 2004. Privacy as contextual integrity. *Wash. L. Rev.* 79 (2004), 119.

[35] Minolini Nithyanandam. 2016. An active rule-based system for XACML 3.0. (2016).

[36] Raúl Pardo, Musard Balliu, and Gerardo Schneider. 2017. Formalising privacy policies in social networks. *Journal of Logical and Algebraic Methods in Programming* (2017).

[37] Raúl Pardo, César Sánchez, and Gerardo Schneider. 2018. Timed Epistemic Knowledge Bases for Social Networks. In *International Symposium on Formal Methods*. Springer, 185–202.

[38] Joseph Phelps, Glen Nowak, and Elizabeth Ferrell. 2000. Privacy concerns and consumer willingness to provide personal information. *Journal of Public Policy & Marketing* 19, 1 (2000), 27–41.

[39] Joseph Reagle and Lorrie Faith Cranor. 1999. The platform for privacy preferences. *Commun. ACM* 42, 2 (1999), 48–55.

[40] J Rose and C Kalapesi. 2012. Rethinking personal data: Strengthening trust. *BCG Perspectives* 16, 05 (2012), 2012.

[41] Herman T Tavani. 2007. Philosophical theories of privacy: Implications for an adequate online privacy policy. *Metaphilosophy* 38, 1 (2007), 1–22.

[42] Herman T Tavani and James H Moor. 2001. Privacy protection, control of information, and privacy-enhancing technologies. *ACM SIGCAS Computers and Society* 31, 1 (2001), 6–11.

[43] Que Nguyet Tran Thi and Tran Khanh Dang. 2012. X-STROWL: A generalized extension of XACML for context-aware spatio-temporal RBAC model with OWL. In *Digital Information Management (ICDIM), 2012 Seventh International Conference on*. IEEE, 253–258.

[44] Giuseppe A Veltri and Andriy Ivchenko. 2017. The impact of different forms of cognitive scarcity on online privacy disclosure. *Computers in Human Behavior* 73 (2017), 238–246.

[45] Samuel D Warren and Louis D Brandeis. 1890. The right to privacy. *Harvard law review* (1890), 193–220.

[46] Alan F Westin. 1968. Privacy and freedom. *Washington and Lee Law Review* 25, 1 (1968), 166.