UNICORN FRAMEWORK: A USER-CENTRIC APPROACH TOWARD FORMAL VERIFICATION OF PRIVACY NORMS

by

Rezvan Joshaghani

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

May 2019

© 2019
Rezvan Joshaghani
ALL RIGHTS RESERVED

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Rezvan Joshaghani

Thesis Title: UNICORN Framework: A User-centric Approach Toward Formal Verification of Privacy Norms

Date of Final Oral Examination: 11 March 2019

The following individuals read and discussed the thesis submitted by student Rezvan Joshaghani, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Hoda Mehrpouyan, Ph.D.	Chair, Supervisory Committee
Dianxiang Xu, Ph.D.	Member, Supervisory Committee
Elena Sherman, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Hoda Mehrpouyan, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

ABSTRACT

In the development of complex systems, such as user-centric privacy management systems with multiple components and attributes, it is important to formalize the process and develop mathematical models that can be utilized to automatically make decisions on the information sharing actions of users. While valuable, the current state-of-the-art models are mostly based on enterprise/organizational privacy perspectives and leave the main actor, i.e., the user, uninvolved or with limited ability to control information sharing actions. These approaches cannot be applied to a user-centric environment since user privacy policies are dynamic because they change based on the information sharing context and environment. In this thesis, we focused on developing the main core of the framework which is the privacy formalization and verification engine that allows for the guided and flexible specification of users privacy policies. The formalization and verification engine reasons about the users privacy rules to find privacy violating information sharing actions and ensure that the privacy norms are unambiguous and consistent. Utilizing these privacy norms the framework monitors users information sharing actions to detect privacy violations. In cases that an action is not compliant with the privacy norms, the framework utilizes a game theoretic approach to generate a privacy decision model. This model enables the users to proceed with the violating action without compromising their privacy by suggesting an information negotiation protocol based on the information sensitivity, users trust, and the reward of information sharing action.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

WEF – World Economic Forum

FTC – Federal Trade Commission

OSN – Online Social Network

HIPAA – Health Insurance Portability and Accountability Act

RACL – Restricted Access/Limited Control

CI – Contextual Integrity

XML – Extensible Markup Language

P3P – Platform for Privacy Preferences Project

EPAL – Enterprise Privacy Authorization Language

XACML – eXtensible Access Control Markup Language

LTL – Linear Temporal Logic

PIA – Privacy Interface Automata

WS-BPEL – Web Service Business Process Execution Language

DEDL – Dynamic Epistemic Deontic Logic

LFP – Least Fixed Point

GLBA – GrammLeachBliley Act

DL – Description Logic

GC – Garbled Circuit

HE – Homomorphic Encryption

UBM – User Behavior Model

PPM – Privacy-Preserving Model

SMT – Satisfiability Modulo Theories

EL – Expression Language

SpEL – Spring Expression Language

LIST OF SYMBOLS

P	Set of agents
T	Set of information attributes
p	Agent
t	Information attribute
κ	Knowledge state
a	Command action
sh	Share Command
st	Stop Command
Act	Communication Action
$\mathcal{P}(T)$	Power-set of T
K	Set of knowledge states
κ_0	Initial State
δ	Transition function
ρ	Role
τ	Attribute Type
$\mathcal{T}$	Set of Attribute Types
$\mathcal{R}$	Set of Roles

AT	Partial map from information attributes to attribute types
AR	Maps an agent to a nonempty set of roles
ψ	Environmental Condition
v	Environmental Variables
φ	Sequencing Condition
$\mathcal{A}$	Set of Access Permission
n	Privacy Norm
$\mathcal{N}$	Set of Privacy Norms
$\mathcal{L}_{\downarrow}$	Projection Language
$\hat{K}$	Set of Abstract States
c	Transition Pre-condition
m	Monitor
$\widehat{Act}$	Abstract Transition
MS	Map to abstract state
MT	Map to abstract transition
Σ_i	Privacy Decisions of Player i
u	Utility Function
s	Information Sensitivity
pr	Permanency
fd	Frequency over Recipient Diversity
tr	Trust

f Frequency

d Duration

CHAPTER 1

INTRODUCTION

A Privacy Bill of Rights was endorsed by the White House in 2012, a response to an increasingly loud objection of citizens to the lack of privacy and fair information practices guidelines [26]. The predicament was not only recognized by the US government, but also has 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) [50] and "Recommendations for businesses and policymakers" by the Federal Trade Commission (FTC) [16]. Despite all these efforts, ubiquitous online monitoring of users' activities [36] and scandalous data breaches, e.g. Facebook and Cambridge Analytica, continue to haunt Online Social Network (OSN) users [2, 15]. These privacy breaches are often due to lack of regulatory standardization, thus, delegating it to users to manage what information should be shared with whom and when. However, keeping track of this vast amount of information sharing can be an overwhelming task [57]. Therefore, ample tools and algorithms should be developed and provided to users 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 consists of: access control languages [4, 41, 49], different privacy settings in applications, and formal privacy policies [5, 14, 19, 45]. 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 information sharing actions. In addition, uniform privacy regulations like HIPAA or a corporation's privacy policies are domain-specific and static with a little or no change over time. However, the user's privacy policies are dynamic and change based on the information sharing context and environment. In addition to dynamicity, the privacy framework should provide the user with customizable policies since the definition of privacy varies from user to user based on their personality, cultural background, etc.

The proposed framework is considered as the first step toward our efforts to build a user-centric privacy management system. In this paper, we focus on developing the main core 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 find privacy violations and ensure that the privacy policy is unambiguous and consistent. The underlying formalization utilizes two formal models, the user's behavior model, and the privacy-preserving behavior model. The user's behavior model represents all the user's information transfers. The privacy verification is performed by mapping each user's behavior 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 [1] 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.

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

The future work will extend the current effort to include other layers, i.e. user privacy requirement elicitation, identifying and categorizing information shared by users, and establishing the relationship between a user and recipients.

1.1 Thesis Statement

- Design and Implementation of a users-centric privacy formalization and verification engine that specifies the privacy norms from the user's perspective.
- Describing run-time approaches in the privacy framework to detect privacy disclosure events and inconsistent privacy norms.
- Upon creation of privacy verification framework, suggesting a privacy negotiation protocol that enables users to negotiate private data with the third party services.

CHAPTER 2

RELATED WORKS

For over 120 years researchers have been studying privacy in different settings of technological advances [53, 58]. The first privacy theory emerged when newspapers started to publish personally intrusive articles and photographs [53]. This led to seclusion and non-intrusion theory of privacy that defined the user's privacy as "the right to be left alone" [58] or being free from intrusion [23]. As new technologies were introduced such as databases containing the personal information of the users [53] the information-related privacy concerns [47] emerged. To address these concerns researchers developed the control [59], limitation [21], and Restricted Access/Limited Control (RACL) [40] theories to enable users to control and limit their privacy while sharing 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." [54] 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, [43] 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 had 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) [49], Enterprise Privacy Authorization Language (EPAL) [4], eXtensible Access Control Markup Language (XACML) [41], and Confab [25]. 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 [33, 44, 55].

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. [35] proposed a technique that translated the privacy specification of web services to LTL formulas. Then a Privacy Interface Automata (PIA) was 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. [32] also proposed an approach to enforce privacy requirements using role-based access control and LTL. The authors base their technique on behavior automata that model the system behavior (gathering or using data) and access control automata, which enforce the privacy policies. Kouzapas et al. [31], 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

and Surridge [22] 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. [45], 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, [30, 46] extended [45] to include time characteristics to the privacy language by adding time interval and LTL which led to the creation of timed privacy frameworks for social media. Both frameworks used a social network model and privacy policies as properties for model checking [8] verification.

While the variety of implementations based on the theory of limitation, control and RACL continues to grow, with the advent of CI, Barth et al. [9] have proposed the formalization of CI theory of privacy. The authors 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 . [19] 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, user privacy is dynamic in nature and as relationships and user's requirements changes so should the privacy norms. These changes can result in privacy policy conflicts. Therefore, Breaux et al. [14] 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 developed based on the organizational privacy requirements which are not compatible with individual users privacy requirements. Table 2.1 summarizes the formal privacy studies that are related to this research.

2.1 Privacy Policy Negotiation Protocol

Compared to the existing negotiation methods [29,48,52] that heavily involve users in decision making or compromise user's privacy for reaching an agreement on sharing information, in the proposed method the user does not need to change the privacy settings to share information that might be disclosing and the user is able to share

Table 2.1: Relate works in formal privacy policies

Paper	Formalism used	Privacy Policy Domain	Evaluation Method	Evaluated Policy
[45]	Epistemic logic	Social Networks	Model Checking	Facebook, Twitter
[30]	Epistemic logic, LTL	Social Networks	Model Checking	Facebook, Twitter
[31]	π-calculus, Privacy calculus	Legal definition of privacy-Enterprise	Model Checking	Electronic traffic pricing, Speed-limit enforcement
[22]	Labeled Transition System	Cloud Services	Model-driven analysis over all transitions	OPERANDO project trials-Health care area
[5]	dynamic epistemic deontic logic (DEDL)	Enterprise- Web services	Model Checking	Authors example of a website policy
[19]	least fix point logic (LFP)	Enterprise	Model Checking	HIPAA-GLBA
[14]	description logic (DL)	Enterprise multi stakeholder privacy policies	Logical conflict detection algorithm	Facebook-Zynga- AOL Advertising
[35]	LTL,interface automata	Web services	Model Checking (SPIN)	Online shopping scenario
[32]	Role base access control and LTL	Enterprise	Model Checking (SAL symbolic model checker)	Authors privacy scenarios
[41]	XML	Access control at the enterprise level	Security Monitors	-
[4]	XML	Access control at the enterprise level	Security Monitors	-
[49]	XML	Internet users and websites	Security Monitors	-

such information in a secure non disclosing manner. To make this goal happen we need to use secure multi-party protocol as part of our privacy negotiation. The secure multi-party protocols were designed as a solution to the problem of comparing different values without revealing the values. The major methods in this area are:

Garbled Circuit (GC) This function representation was used in the first multi-party protocol, by Yao [62, 63]. Yao introduces the multi-party protocol as a solution to the millionaire problem in which there are two millionaire that want to know who is the richest among them without disclosing the amount of their wealth. As a solution, both parties inputs are given to a Garbled circuit that represents computation function method, and the result of the circuit based on the inputs is the answer to the comparison of the inputs.

Oblivious Transfer In oblivious transfer, the sender sends messages to the client but doesn't keep the order of the messages. Later Yao's work was expanded by Jakobsson and Yung [27] by defining the socialist millionaire problem in which the two parties want to know if they have the same amount of wealth or not. Continuing these studies different methods for secure multi-party computation was proposed. An efficient way of oblivious transfer was proposed by Naor and Pinkas [42].

Homomorphic Encryption(HE) A form of encryption that is the result of a specific type of computation on the encrypted data equals to the result of the data as if the same computation was done on it and then the result was encrypted [56].

Set intersection Being able to share an integer privately enables us to share datasets privately. Freedman et al. [20] introduced multi-party computation of

set intersection known as PSI-CA using homomorphic encryption and Hamming Distance. Later [12, 13] built their framework upon PSI-CA as a cryptography foundation and used Jacquard distance and Mini-hash to compute the similarity of two sets and employ the PSI-CA as the protocol.

Development in this area resulted in creating generic tools that perform secure multi-party protocols like TASTY [24], Fairplay [38], and VIFF [17]. Table 2.2 presents some detail on each tool. Tasty was developed after Fairplay and VIFF and it contains the functionalities of both tools. In addition, FairplayML can be used as a function language in Tasty.

Table 2.2: Comparison of 3 generic tools for secure multi-party protocol

	TASTY	Fairplay	VIFF
Function Description	Boolean circuit, secure multi-party computation language, arithmetic circuit based functions	Boolean circuit, secure multi-party computation language	arithmetic circuit based functions
Protocol Implementation	GC, HE	GC	HE
Tool Language	Python	Java	Python

The mentioned mechanisms and libraries are not applicable to all information types and comparisons. Also, they are computationally expensive which makes them impractical solutions for every communication in a user-centric environment. Besides, in order for users to be able to use such protocols both sides of the communication have to implement and use the protocol which may not be the case with third party services.

CHAPTER 3

METHODOLOGY

This research extends the concept of contextual integrity [9] to provide mathematical models and algorithms that enables the creation 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 Behavior Model (UBM) 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 UBM to its corresponding action in PPM and the actions without mapping are marked as privacy-violating. The rest of this section explains the above concepts in details.

3.1 The User Behavior Model (UBM)

In this research, we draw from the formal definition of entities that construct *Information Communication* to model user's information sharing behavior with the recipients, which are defined as agents [5, 9]. 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. 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 $(Alice, \{\text{home address}, \text{credit card number}\})$ 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 the 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 either to share or to stop sharing information with an agent. Each share, sh, or stop 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 a user intends 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 Behavior Model (UBM))*** *Let* $UBM = (K, Act, \rightarrow, \kappa_0)$ *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.*
- $\delta : K \times Act \mapsto K$ *is a transition function, transform the system state with actions* $(a, p, \widetilde{t}\,)$ *as follows:*
  - $\delta(\kappa, (sh, p, \widetilde{t}\,)) = \kappa'$*, where* $\kappa' = \kappa \cup \{(p, \widetilde{t}\,)\}$,
  - $\delta(\kappa, (st, p, \widetilde{t}\,)) = \kappa'$*, 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 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 attributes that contained t_2 resulting in $\kappa' = \{(p, \{t_1\})\}$. For the sequential information sharing model, we consider a scenario where a 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 user's information sharing activities at run-time. Therefore, based on the proposed UBM 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 it is inefficient to define a separate privacy norm for each ρ (role) and τ (attribute type), the proposed model abstracts these two elements. These abstractions allow to have the same information disclosure norms for 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 $AT : \mathcal{P}(T) \rightharpoonup \mathcal{T}$. 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 3.1, and 3.2 demonstrate an example of this hierarchy structure and some attributes and attribute types in that structure. The dashed lines represent the

mapping between attributes and their type and the solid lines depict the order relation between the attribute and types.

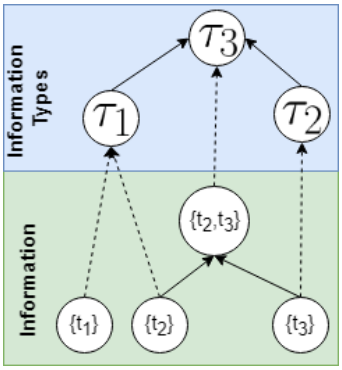

Figure 3.1: An example of the partial order of the attributes and attribute types where the top layer show the attribute types and the bottom layer show the information themselves.

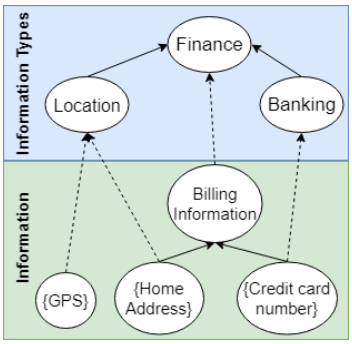

Figure 3.2: t_1 =GPS information, t_2 = home address, and t_3 = credit card number. The middle layer represents the information that are used together for example the credit card number and the home address go together for billing information that is a considered as financial type.

Similar to [9] 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 framework, the partial order $\leq$ 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., $\rho_2 \leq \rho_1$. The

order between roles implies the amount of relative privacy restriction of them where $\rho_2 \leq \rho_1$ 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 system adds additional roles that related to ρ through $\leq$. In other words, the set of roles for p should be closed under $\leq$. 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 follows:

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 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, the 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, to define temporal conditions in privacy norm, a simplified regular expression will suffice, e.g the precedence of two communication actions or a constant occurrence a communication action can be sufficiently expressed 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:

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

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 element 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. Table 3.1 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.

Table 3.1: 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	$\rho_1 < \rho_2$	$c_2 \Leftrightarrow c_1$ $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	$c_2 \Longrightarrow c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	$c_2 \Longrightarrow c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	$c_2 \Longrightarrow c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	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, $c_1 \Longrightarrow c_2$, $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$. BFr can have less restrictive access, $c_2 \Longrightarrow c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	Share Fin with Fr when c1 and s1, share Loc with BFr when c2 and s2. Fin should be guarded the same or better, $c_2 \Longrightarrow c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$. BFr can have less restrictive access $c_2 \Longrightarrow c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	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, $c_2 \Longrightarrow c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	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, $c_2 \Longrightarrow c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	Since Loc and Bank are incomparable then those norms should always be consistent.
C	$\rho_1 = \rho_2$	$c_1 \Longrightarrow c_2$ $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$	$c_2 \Longrightarrow c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	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 $c_1 \Longrightarrow c_2$, $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$. 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, $c_2 \Longrightarrow c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$. 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	$\rho_1 < p > \rho_2$	$c_1 \Longrightarrow c_2$ $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$	$c_2 \Longrightarrow c_1$ $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$	$c_2 \Leftrightarrow c_1$ $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	$c_2 \Leftrightarrow c_1$ $\mathcal{L}(s_1) = \mathcal{L}(s_2)$	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 $c_1 \Longrightarrow c_2$, $\mathcal{L}(s_2) \subseteq \mathcal{L}(s_1)$. 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 $c_2 \Longrightarrow c_1$, $\mathcal{L}(s_1) \subseteq \mathcal{L}(s_2)$. 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.

Definition 2 ***(Trivial 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 3.1 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 3.1 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 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. The regular expression consistency checking process is described in Algorithm 1.

The process of mapping is described Line 2 and 3 of Algorithm 1 for φ_1, φ_1 which calls a function described in Algorithm 3. To achieve a similar language to compare φ_1, φ_2 we traverse over each one. 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

Algorithm 1: Substituting attribute types to reach a same alphabet

Input: Two regular expressions in form of $\varphi_1 = \{(\rho, \tau_1), (\rho, \tau_2), \ldots, (\rho, \tau_k)\}$ and $\varphi_2 = \{(\rho', \tau'_1), (\rho', \tau'_2), \ldots, (\rho', \tau'_n)\}$

Output: **True** if φ_1 is consistent with φ_2 and **False** otherwise

```
1  function regexConsistencyCheck(φ1, φ2)
2      map1=findReducedAlphabet(φ1,φ2);
3      map2=findReducedAlphabet(φ2,φ1);
4      φ̃1=regexSubstitution(map1,φ1,k);
5      φ̃2=regexSubstitution(map2,φ2,n);
6      if ρ > ρ' then
7          if L(φ̃2) ⊆ L(φ̃1) then
8              return True;
9          else
10             return False;
11     else if ρ < ρ' then
12         if L(φ̃1) ⊆ L(φ̃2) then
13             return True;
14         else
15             return False;
16     else if ρ1 ∩ ρ'1 ≠ ∅ or ρ == ρ' then
17         if L(φ̃1) == L(φ̃2) then
18             return True;
19         else
20             return False;
21     else
22         return True;
```

Algorithm 2: Algorithm that create the same alphabet for two regular expressions

Input: Two regular expressions in form of $\varphi_1 = \{(\rho, \tau_1), (\rho, \tau_2), \ldots, (\rho, \tau_k)\}$ and $\varphi_2 = \{(\rho', \tau'_1), (\rho', \tau'_2), \ldots, (\rho', \tau'_n)\}$

Output: A map of the φ_1 children that exist in φ_2

```
1 function findReducedAlphabet(φ1, φ2)
2     forall τi ∈ φ1 do
3         map1={};// A map that hold the attribute types and it
              possible substitutions
4         forall τ'j ∈ φ2 do
5             if τ'j.isChildOf(τi) then
6                 map1.add(τi, τ'j); // add τ'j to the set of possible
                      mappings for τi
7             else if hasCommonChild(τi, τ'i) then
8                 map1.add(τi, getCommonChildren(τi, τ'j))
9     return map1
```

Algorithm 3: Substitution algorithms for a regular expression that reduces the language based on the mapping of the uncommon alphabet.

Input: A regular expression φ,map of substitutions and the $|\varphi|$

Output: The substituted regular expressions $\widetilde{\varphi}$

```
1 function regexSubstitution(map, φ, size)
2     for i from 1 to size do
3         if !map.get(τi).isEmpty() then
4             psb=allPossibleSubs(map1.get(τi));
5             φ̃1=φ1.subForRegex(τi,psb);
6     return φ̃;
```

attribute types in both φ_1, φ_2 the roles in them decide the consistency of two regular expression.

In line 4 and 5 of Algorithm 1 calls a function from Algorithm 3 to substitute the uncommon parts of the regular expressions by the common alphabet retrieved form Algorithm 2. In Algorithm 3 the function "allPossibleSubs" generate all the possible substitution for attribute type τ_i. 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 has 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 a same alphabet the consistency of the regular expressions can be decides based on their roles.

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 3.1 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 3.1 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 $(\hat{K}, \widehat{Act}, c, m)$ representing $n_i = ((\rho, \tau), \psi, \varphi) \in \mathcal{N}$ where $\hat{K} = \{\mathcal{P}(\mathcal{A})\}$, $c = \psi$ is the pre-condition and m is a monitor representing φ regular expression. The transition set $\widehat{Act}$ members are 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 $MS(p, t)$ exists, and the environmental variables satisfy ψ and $m(MT(a, p, t))$ 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.

3.3 Negotiation Protocols in Case of Invalid Behavior

This framework reports the violating actions to the user and allows the user to decide whether he wants to proceed with the action or not. However, if the users decide to continue with the action, they can compromise their privacy. As a solution, the proposed framework creates a decision model for invalid actions that enables the user to share the information without compromising the privacy. To create the decision model we define the information sharing as a game form $\Gamma(N, (\Sigma_i)_{i \in N})$ where $N = \{1, \cdots, n\}$ is a finite number of recipient agents and Σ_i is a set privacy decisions that they can play [11]. We can compute u_i for each strategy to show the utility of each player after information sharing. The utility function u shows how satisfied the users is with the communication action. The utility function defined in this paper depends on four parameters that according to recent literature are contributing factors

to the users privacy decisions [37, 60, 61]. The first parameter is the privacy policies. The privacy policies dictate the amount of the information that the user is willing to share with the other agents which brings us to the second and third parameters the "information sensitivity" and "trust". For each information attribute the user can assign a sensitivity value. Also, for each recipient agent a trust value is assigned. These three parameters capture the nature of access permission; However, recent studies [34, 37] show that there is another contributing factor known as "reward" that makes privacy decisions more than an access control mechanism. The following section describes each one these parameters in detail and then creates a utility function upon them.

The $u < 0$ is an indication of privacy violating action due to lack of an access permission and $u = 0$ is an special case that the access permission exists but the privacy conditions are not satisfied. In such cases when the $u \leq 0$ and the user also requires a result from the service then there are two options available. The first option is to ignore the privacy norms and share the information. Although convenient, ignoring the privacy norms can have serious consequences. The second option is to use different privacy mechanisms based on the parameters of the utility function. These mechanisms allow the users to share information without jeopardising their privacy.

3.3.1 Information Sensitivity

The users assign different values to different information attributes. Although, these value assignments for users do not have a metric and they are subjective, when the information attribute value is transformed into a measurable metric for example expressing the values in dollar amount, users become more privacy cautious [51, 61]. Since privacy preferences vary from person to person, the relative sensitivity of

information attributes is also different for different users. For the purposes of this research, we assume that these values are learned through some mechanism from the user behaviour and are available to our decision model.

We define the information sensitivity s for each information attribute that is not a combined attribute. Then, for each combined attribute we sum-up the sensitivity of its children and add a sensitivity of s' to it. This way an information attribute with more children is considered more sensitive since it is more revealing.

Information sensitivity $s = (pr, fd)$ consists of parameters pr as information attributes permanency and fd as the frequency of information sharing to the diversity of the recipients. A more permanent information attribute has greater sensitivity compared to the less permanent attributes. For example, the user's birthday is a permanent information attribute, however, the current GPS values of the user's device are less permanent. Therefore, we define pr as the permanency factor of information attribute t. The permanency factor for a combined information attribute equals to the permanency factor of its child with the greatest permanency factor.

Another good indicator of information sensitivity is the frequency that an information attribute is shared [51]. The information attributes that are shared more frequently can be considered as less sensitive. However, the frequency alone is not reliable enough. In order to have a better sense of how sensitive information is to the user, we need to look at the ratio of frequency to the recipient diversity. This way, if a user is sharing an information attribute frequently with a large number of recipients, he is considering this information attribute less sensitive compared to an information attribute that is shared frequently with only a few. For example, the users that have a habit of adding location tags to their posts on social media can be considered as users that assign a low sensitivity value to their current location. Although the

frequency to recipient diversity ratio can be used to separate the dependency between the information sensitivity and trust parameters, it is not sufficient for expressing the user relationship with the recipients. Hence, we need to define trust as a separate factor.

3.3.2 Trust

Studies show that trust and information sensitivity have a direct relationship with each other [34,60]. Users tend to share more sensitive information with the recipients that are more trusted. We define the amount of trustworthiness of a recipient with the variable $tr \in [0, 1]$ where 1 represents total trust in the recipient and 0 indicates an untrustworthy recipient. Trust is computed as a function of information sensitivity s, frequency f, and duration d of the sharing. We assign trust values to the recipient agents and based on the agents trust-score, agents are clustered into different trust roles. Therefore, a role that is more specialized has a higher score compared to its child roles. In this setting, the roles are the boundaries of trust in addition to their societal semantics. Thus, if an agent belongs to a specific role but the trust score is not in the range of the roles trust score then the agent has to be moved to a role that matches its trust-score. This way, we can capture the dynamic relationship between the user and the recipient agents as their relationship gets stronger or fades away. The formal definition of trust is as follows:

$$tr(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

where z_i is the sum of s, f, d of each communication action with the agent p_i

3.3.3 Reward

The reward is a quantifier on how rewarding the information sharing is to the user. A simple method of computing the reward is subtracting the profit gained by information sharing from the cost of disclosure. The literature that exists in this area considers two categories of rewards, the monetary reward, and the none monetary reward. Also, the literature categorizes reward as with and without incentive. An example of information sharing with monetary reward can be a mobile application that offers the users an ad-free application only if the user shares the current location with the application. Also, an example of a non-momentary reward Can be a Facebook account that collects users data instead of providing free services that connect users on the application platform. Although it might seem that Facebook is providing free services to the users, they are actually paying Facebook by sharing their data that is used for the targeted advertisements to the account holders. The reward of information sharing is not always getting a service, it can also be by making a complex process more convenient. Computing the amount of reward for each communication action is a hard task; however, for the purposes of this research, we are assuming that there is a mechanism that can be used to return the reward of a communication action.

3.3.4 Privacy Game

We are assuming that each privacy game has one player who is the user and the other player is the information recipient. The recipient of the information can be a service, another human or an application. The users goal is to keep their information as private as they can and the opponent player goal is to get most information possible.

We can categorize the opponent players based on their game strategies into different groups. The rest of this section will describe the different possible player.

Third Party Services Strategies

Take it or leave it: These are the players that they only have one game strategy. Most services and applications fall into this category. The "take it or leave it" players only provide services to the user if the user provides them with all the information they need. Therefore, if the information request from these services be against the users' policies the utility function of the users $u(s, t, r) < 0$ for all the strategies and negotiation is not possible with this type of players.

Greedy Players: The greedy players are the developed version of "Take It or Leave It" players. They have a minimum requirements for information sharing but they always ask for more than the minimum they need and they don't reveal that providing the rest of the information is optional. However, when the user denies them the optional information they still provide services. The applications that request access permissions but still provide complete or partial services when the permission access is denied are greedy players. The Greedy players number of strategies is based on the number of extra information they need.

Let $S_g = \{s_0, \cdots, s_k\}$ be all the strategies that a greedy player has. The s_k being the first move of the greedy player where it asks for all the information(the minimum requirement plus the extra ones) and s_0 be the strategy that it only asks for the required information. Therefore, it is possible to negotiate with them.

Luring Players: The Luring players are the players that provide an additional reward for sharing information with them. They can be considered as greedy players but they try to increase the users utility function by providing rewards for each extra information they want. An example of luring players are the applications that remove the advertisements if the user shares his or her location with them.

3.3.5 User's Game Strategy

For each privacy game, we assume that vector $Vec_p =< t_0, \cdots, t_k >$ is all the information attributes that opponent agent p requires to perform the service. Also, we assume that the user is unaware of the p playing strategy. However, the user knows whether the information sharing is rewarding or not. The users wins the game when they get the service they want. The game ends either by winning or running out of strategies. The user starts the game with strategy

$$s_0 = \{(sh, p, t) | \exists n = (\rho', \tau', \psi, \varphi) s.t. \rho' = AR(p) \wedge \tau' = AT(t)\}$$

which is only sharing the information attributes with p that are allowed by privacy norms. If the user does not win with s_0 the system sorts elements of $\overline{Vec_p} = Vec_p - s_0$ in ascending order based on the value of $\frac{Information\,Sensitivity}{Reward}$. Moreover, if the communication action is not rewarding the vector is sorted by the information sensitivity alone. Then we create strategy $s_i = s_{i-1} \cup v_i \, where \, v_i \in \overline{Vec_p}$ until the game ends. The sharing mechanism for v_i is available in Table 3.2 where S, T, R are information sensitivity, trust and reward respectively. Also, H, M, L indicate high, medium and low vales and $+, -$ describe whether the communication action is rewarding or not.

Table 3.2: Sharing mechanism for violating information types

(S,T,R)	**Mechanism**
(H,H,+)	Control Loop
(H,L,+)	Secure Computation
(H,M,+)	Obfuscation
(H,H,-)	Control Loop
(H,L,-)	Secure Computation
(H,M,-)	Obfuscation
(L,H,+)	Control Loop
(L,L,+)	Control Loop
(L,M,+)	Control Loop
(L,H,-)	Control Loop
(L,L,-)	Control Loop
(L,M,-)	Control Loop
(M,H,+)	Control Loop
(M,L,+)	Secure Computation
(M,M,+)	Obfuscation
(M,H,-)	Control Loop
(M,L,-)	Secure Computation
(M,M,-)	Obfuscation

CHAPTER 4

IMPLEMENTATION AND PERFORMANCE EVALUATION

4.1 Implementation

As a proof of the concept, we prototyped the proposed framework in the Java programming language [1]. Figure 4.1 depicts a diagram of the implementation's architecture. The proposed framework is modularized into three layers:

- User interface layer which takes the user's intentions in a structured format.
- Translation layer which translates the frameworks from UI to privacy norms and formal notation.
- Verification layer that evaluates norms consistency and compliance of the information sharing action with privacy norms.

In the following sections describe the implementation details of each of the components in each layer.

4.1.1 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

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

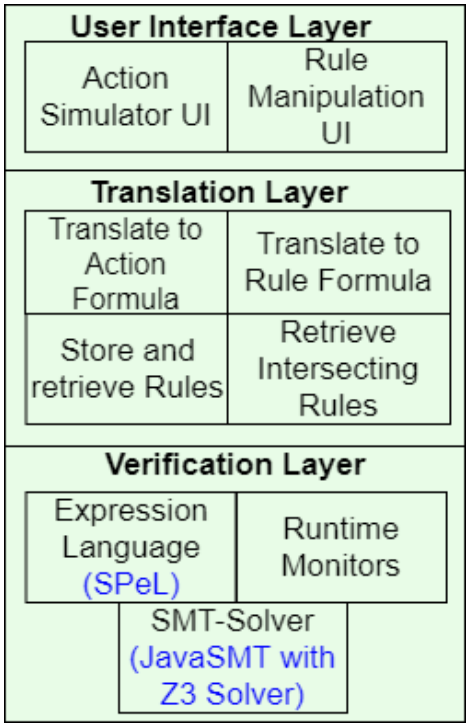

Figure 4.1: The architecture of user-centric privacy framework. The blue components show the libraries and technologies used in the proposed framework.

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. The UI was created with JavaFX framework and the Figure 4.2 depicts UI of the framework.

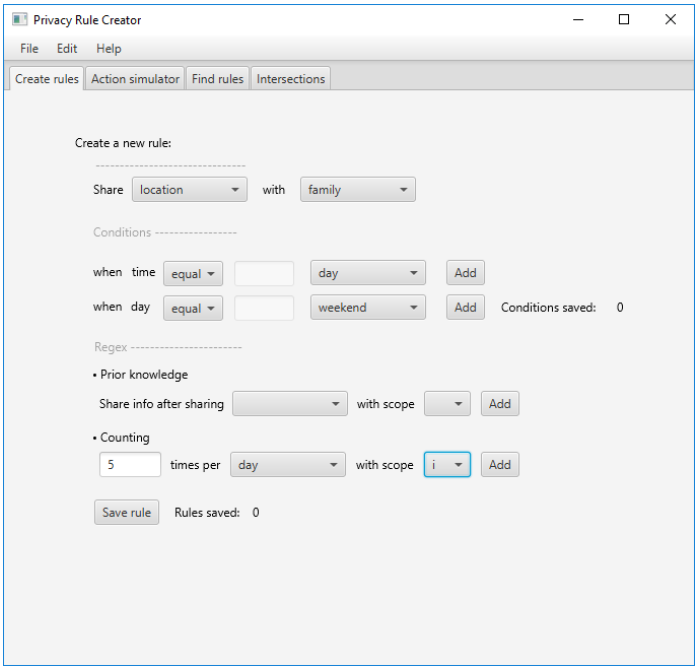

Figure 4.2: The norm creation UI.

4.1.2 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. In this framework, MariaDB version 10.2 was used as the database. To evaluate each action with the attribute t and the agent p, norms that have roles where $\rho = AR(p)$ and attribute type $\tau = AT(t)$ are retrieved from the norm table and sent to the verification layer.

4.1.3 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.

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 translates them to satisfiability modulo theories (SMT) solver formulas. Then the SMT solver proves that the implication or equivalency relation holds and 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 new norm is consistent it will be added to the database. The implementation of the proposed framework utilizes JavaSMT [28] with the Z3 solver version 4.3.2 [18] for consistency checking over the environmental variables and "brics" automate library version 1.12-1 [39] for sequencing conditions.

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 keys. 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 are 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 4 shows the general steps taken to implement the information sharing action verification process. Considering the implementation bellow, in the next section we discuss the performance evaluation of the proposed framework.

Algorithm 4: Action verification algorithm.

```
1  Input: ISA (An information sharing action)
2  Output:Boolean value indicating the verification result.
3  norms=[]
4  recipientGroups=ISA.recipient.getGroups()
5  for g in recipientGroups do
6  |  norms.append(getnorms(ISA.InformationType, g))
7  if (norms.size> 0) then
8  |  for j in norms do
9  |  |  if !(j.evalEnvironmentalCondition (ISA.environment)) then
10 |  |  |  return false
11 |  |  else
12 |  |  |  if !(j.evalSequencingCondition(ISA)) then
13 |  |  |  |  return false
14 |  return true
15 return false
```

4.2 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 is 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 groups.

Table 4.1 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 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. The performance evaluation was over 2 environmental variables time and day and the allowed sequencing conditions were "a after b" and "a occurred k times". Also, notice that the performance of the action verification depends on the performance of the 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 4.1: Action verification performance evaluation results. The columns show the response time for Access Permission (AP), Environmental Conditions (EC), Sequencing Conditions (SC) and All shows the average verification time for a norm that have all the elements.

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 the 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 4.2 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 bottle neck 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 4.2: 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

CHAPTER 5

CONCLUSION AND FUTURE WORK

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. Furthermore, users' privacy policies are dynamic in nature and evolve to reflect the changes in users' relationships and present situations. However, current privacy management systems cannot address these privacy needs adequately since they are not designed based on the users' privacy perspective. This issue in privacy management systems decreases their usability among everyday users and puts unaware users at risk of information disclosure. 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 used in user devices, which compared to servers and general purpose computers that are targeted in the existing work 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 [10] shows that the evaluation of the regular expressions has significantly less overhead

compared to LTL. Furthermore, our framework similar to [7] allows for adding environmental conditions to the privacy norms which makes the policy creation more accurate. In addition, for privacy enforcement and verification our framework utilizes a two layered mapping approach from a transition system to a guarded transition system which has less overhead compared to model checking techniques. Further, the proposed framework allows suggested privacy decisions based on information sensitivity, trust and reward. There are multiple privacy negotiation mechanisms that utilize obfuscation and secure computation, without compromising their privacy. However, these methods are not always applicable to all information types.

Since this work is our first step toward a formal dynamic user-centric privacy model which is creating a formal model of privacy policy, the burden of providing input for the rule creation process is on the user. In our future work, the current user interface will be removed, and rule inputs will be created automatically utilizing text analysis, speech recognition and AI algorithms that can infer a user's privacy policy based on the users relationships and information sharing behavior. Also, at this point of our work the framework assumes that the values for computing information sensitivity, trust and reward is given. Another assumption of our framework is that all the environmental variables have deterministic values for the evaluation and cannot handle non-deterministic situations. In our future works we intend to address the above limitations by expanding the logic to overcome non-deterministic situations, increase the number of users, and add new rule formulas to our language.

REFERENCES

[1] 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. Privacy and human behavior in the age of information. *Science*, 347(6221):509–514, 2015.

[3] Alessandro Acquisti and Jens Grossklags. Privacy and rationality in individual decision making. *IEEE security & privacy*, 3(1):26–33, 2005.

[4] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (epal). *IBM Research*, 2003.

[5] Guillaume Aucher, Guido Boella, and Leendert Van Der Torre. A dynamic logic for privacy compliance. *Artificial Intelligence and Law*, 19(2-3):187, 2011.

[6] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. *Foundations of Artificial Intelligence*, 3:135–179, 2008.

[7] Susana Alcalde Bagüés, Andreas Zeidler, Carlos Fernandez Valdivielso, and Ignacio R Matias. A user-centric privacy framework for pervasive environments. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1347–1356. Springer, 2006.

[8] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of model checking*. MIT press, 2008.

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

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

[11] Adam Bjorndahl, Joseph Y Halpern, and Rafael Pass. Language-based games. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[12] Carlo Blundo, Emiliano De Cristofaro, and Paolo Gasti. Espresso: efficient privacy-preserving evaluation of sample set similarity. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 89–103. Springer, 2013.

[13] Carlo Blundo, Emiliano De Cristofaro, and Paolo Gasti. Espresso: efficient privacy-preserving evaluation of sample set similarity. *Journal of Computer Security*, 22(3):355–381, 2014.

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

[15] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach. *The Guardian*, 17, 2018.

[16] Federal Trade Commission et al. 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.

[17] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *International Workshop on Public Key Cryptography*, pages 160–179. Springer, 2009.

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

[19] Henry DeYoung, Deepak Garg, Limin Jia, Dilsun Kaynar, and Anupam Datta. 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*, pages 73–82. ACM, 2010.

[20] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.

[21] Ruth Gavison. Privacy and the limits of law. *The Yale Law Journal*, 89(3):421–471, 1980.

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

[23] Jamal Greene. The so-called right to privacy. *UC Davis L. Rev.*, 43:715, 2009.

[24] Wilko Henecka, Ahmad-Reza Sadeghi, Thomas Schneider, Immo Wehrenberg, et al. Tasty: tool for automating secure two-party computations. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 451–462. ACM, 2010.

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

[26] White House. Administration discussion draft: Consumer privacy bill of rights act of 2015. *Retrieved on November*, 15:2015, 2015.

[27] Markus Jakobsson and Moti Yung. Proving without knowing: On oblivious, agnostic and blindfolded provers. In *Annual International Cryptology Conference*, pages 186–200. Springer, 1996.

[28] Egor George Karpenkov, Karlheinz Friedberger, and Dirk Beyer. Javasmt: A unified interface for smt solvers in java. In *Working Conference on Verified Software: Theories, Tools, and Experiments*, pages 139–148. Springer, 2016.

[29] Dilara Keküllüoğlu, Nadin Kökciyan, and Pınar Yolum. Strategies for privacy negotiation in online social networks. In *Proceedings of the 1st International Workshop on AI for Privacy and Security*, page 2. ACM, 2016.

[30] Ivana Kellyérová. A real-time extension of the formal privacy policy framework. 2017.

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

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

[33] 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. Spatio–temporal xacml: the expansion of xacml for access control. *International Journal of Security and Networks*, 10(1):56–63, 2015.

[34] Chechen Liao, Chuang-Chun Liu, and Kuanchin Chen. Examining the impact of privacy, trust and risk perceptions beyond monetary transactions: An integrated model. *Electronic Commerce Research and Applications*, 10(6):702–715, 2011.

[35] Jiajun Lu, Zhiqiu Huang, and Changbo Ke. Verification of behavior-aware privacy requirements in web services composition. *JSW*, 9(4):944–951, 2014.

[36] Mary Madden, Aaron Smith, and Jessica Vitak. Digital footprints: Online identity management and search in the age of transparency. 2007.

[37] Gianclaudio Malgieri and Bart Custers. Pricing privacy–the right to know the value of your personal data. *Computer Law & Security Review*, 34(2):289–303, 2018.

[38] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, volume 4, page 9. San Diego, CA, USA, 2004.

[39] Anders Møller. dk.brics.automaton – finite-state automata and regular expressions for Java, 2017. `http://www.brics.dk/automaton/`.

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

[41] Tim Moses. Privacy policy profile of xacml v2. 0. *Oasis standard, OASIS*, 2, 2005.

[42] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[43] Helen Nissenbaum. Privacy as contextual integrity. *Wash. L. Rev.*, 79:119, 2004.

[44] Minolini Nithyanandam. An active rule-based system for xacml 3.0. 2016.

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

[46] Raúl Pardo, César Sánchez, and Gerardo Schneider. Timed epistemic knowledge bases for social networks. In *International Symposium on Formal Methods*, pages 185–202. Springer, 2018.

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

[48] Sarah Rajtmajer, Anna Squicciarini, Jose M Such, Justin Semonsen, and Andrew Belmonte. An ultimatum game model for the evolution of privacy in jointly managed content. In *International Conference on Decision and Game Theory for Security*, pages 112–130. Springer, 2017.

[49] Joseph Reagle and Lorrie Faith Cranor. The platform for privacy preferences. *Communications of the ACM*, 42(2):48–55, 1999.

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

[51] Sarah Spiekermann and Jana Korunovska. Towards a value theory for personal data. *Journal of Information Technology*, 32(1):62–84, 2017.

[52] Jose M Such and Michael Rovatsos. Privacy policy negotiation in social media. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 11(1):4, 2016.

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

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

[55] Que Nguyet Tran Thi and Tran Khanh Dang. 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*, pages 253–258. IEEE, 2012.

[56] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

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

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

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

[60] Andree E Widjaja, Jengchung Victor Chen, Badri Munir Sukoco, and Quang-An Ha. Understanding users' willingness to put their personal information on the personal cloud-based storage applications: An empirical study. *Computers in Human Behavior*, 91:167–185, 2019.

[61] Shu Yang and Kanliang Wang. The influence of information sensitivity compensation on privacy concern and behavioral intention. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 40(1):38–51, 2009.

[62] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[63] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.