Engineering Adaptive Privacy: A requirements-driven approach for mitigating mobile privacy threats

Inah Omoronyia

Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

Liliana Pasquale

Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

Mazeiar Salehie

Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

Gavin Doherty

Lero – The Irish Software Engineering Research Centre
Trinity College Dublin, Ireland

Bashar Nuseibeh

Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

{April 2012}

Contact

Address ..... Lero
International Science Centre
University of Limerick
Ireland

Phone ..... +353 61 233799

Fax ..... +353 61 213036

E-Mail ..... Lero-AdminSupport@lero.ie

Website ..... http://www.lero.ie/

This work is partially supported by Science Foundation Ireland
under grant no. 10/CE/I1855

Engineering Adaptive Privacy: A requirements-driven approach for mitigating mobile privacy threats

Inah Omoronyia[1], Liliana Pasquale[1], Mazeiar Salehie[1], Gavin Doherty[3], Bashar Nuseibeh[1,2]

[1] Lero – The Irish Software Engineering Research Centre, University of Limerick, Ireland
[2] Department of Computing, The Open University, UK
[3] Lero – The Irish Software Engineering Research Centre, Trinity College Dublin, Ireland

***Abstract*—Mobile devices are an increasingly common part of everyday life. These devices gather and manipulate personal information from and about their users, raising substantial privacy concerns. Frequent changes in the context of use of such devices can blur the boundary between users' public and personal spaces, adding further uncertainty to these concerns. In particular, changing privacy threats can make it difficult for users to adapt their mobile applications to continue to satisfy their privacy requirements, and some degree of automated self-adaptation is essential. There has been little engineering work on studying the impact of a changing context on the design of privacy critical systems. In this paper, we propose a novel approach for software engineering of adaptive privacy in mobile applications. We view privacy threats as the inappropriate disclosure of personally identifiable information. Our approach uses privacy policies, and associated domain and software behavioural models, to logically reason over the contexts that threaten privacy. We generate possible mitigation actions, such as ignoring, preventing, reacting, and terminating interactions that threaten privacy. We implement and evaluate our approach in a prototype tool called *Caprice*. We demonstrate that our approach is computationally feasible, and enables designers to identify plausible privacy threats and to select effective mitigation actions.**

Keywords-privacy; adaptation; mobility; selective disclosure;

I. INTRODUCTION

Consumers and enterprises increasingly rely on mobile systems, such as smart phones, to satisfy their social and business needs. This new generation of systems enable users to form localised, short- and long-lived groups or communities to achieve common objectives. To this end, these systems may need to manipulate user's sensitive information [5], such as location, time, proximity to nearby services, and connectivity to other users. The disclosure of these attributes in an unregulated way can threaten user privacy [6]. For this reason, privacy requirements of users are a critical concern for mobile systems. A representative class of such requirements is *selective disclosure* – deciding what information to disclose, in which context, and the degree of control an individual has over disclosed information.

A key factor affecting selective disclosure in a mobile environment is the frequently changing context, such as changing time, location and activities. These changes blur the boundary between public and personal spaces [1] [2] and can introduce unexpected privacy threats. Additionally, users may be unaware of when and for what purpose sensitive information about them is being collected, analysed or disseminated. This makes it even more difficult for users to adapt their mobile applications to continue to satisfy their privacy requirements.

This challenge calls for a more systematic approach to enable the explicit consideration of privacy in the engineering of mobile software systems. Firstly, it is necessary to continuously examine context changes, such as changing spatio-temporal user attributes, as well as the environmental or regulatory constraints over which such attributes are disclosed. Secondly, mobile systems should be able to reason over changing context to discover privacy threats, and subsequently carryout actions that mitigate these threats. Although there are some methods for addressing privacy at design time [7] (e.g PriS[11], SQUARE for privacy[12], and others [7]), they do not target privacy threats arising from changing context brought about by mobility.

In this paper, we propose a novel approach to engineering adaptive privacy in mobile systems. Our approach aims to support software engineers in the design of applications that appropriately adapt their behaviour to mitigate mobile privacy threats. Our approach helps to identify the contextual properties that have to be monitored in order to detect context changes that might threaten privacy. It also suggests the mitigation actions to be performed at runtime for any privacy threatening contextual change. We build on the notion of *privacy awareness requirement*s to identify attributes that should be monitored in order to detect context changes that might threaten privacy. Privacy awareness requirements are inferred from privacy policies, and their associated domain and software behavioural models. We assume that privacy requirements are operationalised by privacy policies expressed in linear temporal logic (LTL). Our domain models represent the changing context over which the system operates and are expressed in ontology web language (OWL). Behavioural models represent the states and transitions of a system and are represented as finite state machines (FSM).

We then make use of privacy awareness requirements and the above models to analyse privacy threats. The outcome of our analysis includes the discovery of privacy threats depending on the disclosed information. We reason about changes in context that do not satisfy expected privacy policies, and suggest mitigation actions that can be used to ameliorate the threats discovered. The selection of appropriate mitigation actions is based on the sensitivity and obfuscation levels of disclosed information. Sensitivity refers to the criticality or importance of the information to its owners, while obfuscation refers to the granularity of the information.

We implemented our approach in a prototype tool called *Caprice* (http://www.lero.ie/SPARE/Caprice), and evaluated it in a number of ways. Firstly, we demonstrate that, using our approach, a software designer can configure different privacy mitigation strategies; that is, when to ignore, react, prevent or terminate an interaction that is privacy threatening. Secondly, using *Caprice*, we demonstrate that our approach correctly discovers privacy threats. Finally, we demonstrate that our reasoning about privacy threats is computationally viable.

The remainder of the paper is organised as follows. Section II presents some background on privacy and awareness concepts relevant to our overall approach, which is then presented in section III. Section IV describes the privacy awareness requirements used to derive the monitoring needs for adaptive privacy. Section V discusses our proposed privacy threat analysis, while section VI details our approach to threats mitigation. In section VII, we present the implementation of our approach in *Caprice*. Section VIII evaluates and discusses the impact of our approach on the software engineering of privacy critical mobile systems. Conclusions and further work are presented in section XI

II. BACKGROUND

Our research offers three main contributions. First, it describes an approach to identify privacy awareness requirements. Second, it uses these requirements to perform privacy threat analysis. Finally, it supports the selection of a proper mitigation action against discovered privacy threats. Given these contributions this section provides some background concepts on awareness, privacy threats and adaptive privacy.

A. Awareness

Dourish and Bellotti [13] view awareness as "an understanding of the activities of others, which provides a context for your own activities". In requirements engineering, Souza et al. [14] define awareness requirements as the class of requirements about the success or failure of other requirements. Central to awareness is the notion of *context*, *entity*, and *event*. The most frequently cited description of context is that given by Dey [22]: "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*". A popular means of capturing and analysing such context is through the use of domain models (a representation of entities, their attributes with associated possible values and relationships). This is because domain models permit tractable reasoning over the different attribute parameters that a system interacting with its environment can assume [17]. Events and actions are occurrences resulting from context attributes assuming specific parameters (e.g., the door assigned the value 'closed'). This notion of events and actions has been instrumental in modelling the behaviour of software systems. For example, using finite state machines (FSM) [15] events trigger the transition from one state to another given that certain conditions (specific values of context attributes) hold, while actions are used to represent a set of activities that can be performed in a particular state. Context changes occur when events or actions result in the addition of a new attribute, or a change in the assigned parameter of an existing attribute. An operational context then refers to a subset of attributes and their specific values at any particular time. A critical challenge for awareness research is the recognition that context has changed as events or actions occur in the environment, and the inference of current and future operational context based on new events and actions.

B. Privacy threats and adaptation

Interactions that threaten privacy generally fall into two categories: (1) the conduct by which one entity infringes on the rights of another (the subject) to determine for itself whether or not it can perform an action; (2) the conduct by which an entity acquires or discloses information about the subject in a manner for which the subject does not wish to have known or disclosed [9]. The first category is often referred to as the *violation of autonomy*. The second, and the focus of this paper, is normally referred to as the *violation of rights to selective disclosure*. For socio-technical systems, selective disclosure is a boundary regulation process involving the continual management of boundaries between different groups of activities and the degree of disclosure within these groups [3]. Palen and Dourish [4] argue that these boundaries change dynamically as context changes.

In this paper, we adopt contextual integrity [8] as a suitable basis for modelling adaptive privacy in software systems. Contextual integrity posits that the transfer of information from a sender to a receiver in a specific context about a subject (hereafter referred to as a message) is tied to certain transmission principles. This is because contextual integrity represents an explicit model of a sender, receiver, and a subject (hereafter referred to as agents) when disclosing personal information, and the transmission principles that guard the interaction process between these entities [16]. Examples of such transmission principles include notice, consent, confidentiality and reciprocity. These transmission principles are commonly captured in privacy policies [20]. Additionally, contextual integrity provides a means to identify points in the behaviour of a system where the tracking and aggregation of private attributes of users can lead to privacy violations.

The application of contextual integrity to adaptive privacy provides the basis upon which a socio-technical system can evaluate the context of its operation, and adapt where necessary to continuously preserve privacy. An additional component, albeit not mentioned in the contextual integrity framework, is the awareness that users and systems need in order to appropriately evaluate whether the context of their operation abides by associated transmission principles.

Our notion of adaptive privacy is novel as there is no existing work that focuses on aiding the design of adaptation depending on changing privacy threats brought about by mobility. General work on inconsistency management in software engineering has considered so-called "repair actions" [23] to mitigate discovered inconsistencies. However,

despite some similarity to our approach, it does not address privacy. On the other hand, Spiekermann and Cranor do provide a framework for engineering privacy [7], but without focusing on mobility nor on software engineering concerns. Our research brings these two perspectives together. It also differs from traditional requirements monitoring approaches, such as those proposed by Fickas and Feather [24] and by Souza et. al. [14] [30], which do not focus on privacy. Rather, we engineer monitoring activities by identifying context attributes that need to be monitored in order to reason over privacy threatening user interactions.

III. Overall Approach and Running Example

Our approach identifies attributes to monitor at runtime, discover privacy threats that users may encounter and appropriate mitigation actions against such threats. It consists of three steps shown in Figure 1.

The first step considers three inputs from the designer: a set of privacy policies, a behavioural representation of the system, and a set of instantiated operational contexts from the domain model. These inputs are used to identify privacy awareness requirements. This is achieved by identifying attributes that are common to both an operational context and privacy policies. This is in addition to attributes that can be inferred indirectly as a result of realising a specific function in a behavioural model (section IV). The second step performs privacy threat analysis. The analysis involves reasoning over a sequence of operational contexts to identify privacy threats. The inputs into this step are the monitored attributes derived from the privacy awareness requirements. The reasoning seeks to identify operational contexts that do not satisfy expected privacy policies during agents' interaction (section V).

The final step is the selection of mitigation actions against identified privacy threats. Our mitigation strategy is based on the severity of the threat. This severity is calculated through a utility function that is based on the sensitivity and obfuscation levels of disclosed information. Decisions on appropriate mitigation actions are then based on the value of expected utility (section VI). This process is iterative, as a suggested action can result in changes to privacy policies, system behaviour, or new operational context. The next three sections elaborate on each of these steps.

We use a running example of a participatory sensing system [10] to present and assess our approach. In particular, our system aims to help drivers spend less time looking for parking spaces in a city centre. Typical examples include Google Open Spot (openspot.googlelabs.com/) and Roadify (www.roadify.com/). We focus on key functionalities that enable drivers to *post*, *request* and *display* the fuel-optimal, shortest, or fastest routes to an empty parking space. For this example, privacy management requires the capability of drivers in a group to decide the limits of information disclosure to other drivers – about their current location, number of empty spaces, their arrival time, car park name, etc. Effective engineering of adaptive privacy should enable drivers to understand information flows (awareness), weigh the consequences of sharing information (privacy threat analysis), and make informed, context-specific decisions to disclose or withhold information (mitigation action selection).

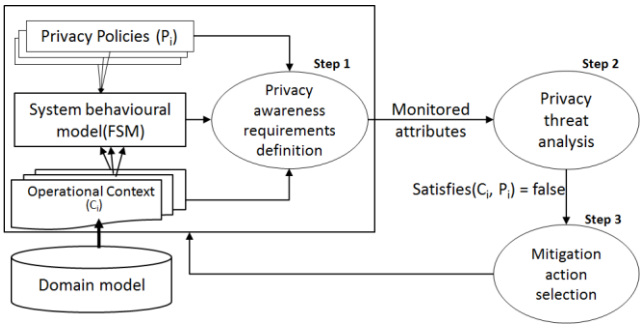


Figure 1 Approach to engineering adaptive privacy

IV. Privacy Awareness Requirements

Privacy awareness requirements identify the set of attributes that need to be monitored to detect context changes that may threaten privacy. Monitoring all attributes can incur performance cost. Therefore, we first instantiate an operational context from a domain model; then, by relating system behaviour to privacy policies we identify a subset of attributes in the operational context to be monitored.

A. Instantiating operational context from domain models

The first element necessary to identify privacy awareness requirements is an operational context instantiated from a domain model. Attribute values of the operational context characterise a specific scenario of information disclosure between two agents. For example, consider a scenario where an agent *A* (sender) transmits a message containing the *location name* of agent *B* (subject) to another agent *C* (receiver). An agent here represents a user's mobile device or a third-party server granted custodian of information about a subject. If *A* and *B* are the same, then *A* is sending message about itself to *C*. In this scenario, an instance of an operational context includes: The values of attributes characterising the agents (e.g., *subjectName* = 'Bill', *senderName* = 'Alex', etc.); attributes characterising the environment of the sender, receiver or subject (e.g. *locationName* ='Foxes St.', *currentTime* = 21.00 etc.) and attributes characterising the satisfaction or failure of transmission principles [16] guarding the message transfer between A and C (e.g. *subjectNotified* = false, *receiverKnown* = true, etc.). A change in context will alter one or more of these attributes.

In addition to the instantiation of a domain model to generate an operational context, there are other properties of such a model necessary for engineering adaptive privacy. For instance, a domain model should describe the different inference relationships between attributes. Other privacy related properties include the sensitivity and obfuscation levels of disclosed attributes. A representation of our domain model is as shown in Figure 2.

Inference relationships allow the deduction of previously unknown information from another disclosed attribute. This can be achieved either via direct implication or aggregation.

They both involve the use of established rules that predict the value of an attribute to some degree of accuracy. Implication inference relations are uni- or bi-directional relations between two attributes. An example of a bidirectional implication inference is *locationName* ⇔ *locationCoordinates* (i.e., if a subject's *locationName* is disclosed, it is possible to deduce the *locationCoordinates* and vice versa). Similarly, the relation *city* ⇒ *country* is unidirectional. Aggregation inference can be deduced by learning patterns that occur in the values of attributes over time. For example, the relation *departureTime* ⇒ *arrivalTime* ∩ *carparkName* infers that the knowledge of the arrival time and car park name may be aggregated to infer the driver's departure time. While some forms of aggregation relations can be bidirectional, we have only considered unidirectional aggregations. Generally, these relationships necessitate monitoring additional attributes to satisfy privacy awareness requirements. For example, assuming *locationName* is private for a subject, there is also a need to monitor the disclosure of *locationCoordinates* of the subject.

Sensitivity level describes the importance of a disclosed attribute to its owner (the subject). Highly sensitive attributes result in greater impact on (or damage to) the subject if disclosed inappropriately. The sensitivity of an attribute can also depend on the context associated with its disclosure. For example, a subject might consider his/her age highly sensitive and not to be disclosed to a cohort of acquaintances found in a social network; in other cases, it is less sensitive compared to the benefit getting 20% off the cost of glasses for over 60's. This variability in sensitivity demands a domain model that traces this property. In this research, we consider sensitivity level varies between very low, low, medium, high and very high.

Obfuscation involves the act of degrading the accuracy and/or precision of a subject's attribute. A highly obfuscated attribute means that it is less likely for a receiver to infer the identity of the subject from the attribute value. Obfuscation can be based on inaccuracy (i.e. lack of correctness) and/or on imprecision (i.e. lack of detail) in disclosed information. Thus, attribute values such as *subjectMobileID* =11223 are less obfuscated (i.e. accurate and precise as it uniquely identifies the subject) compared to *city* ='Limerick' (i.e. accurate and imprecise, because it is not possible to uniquely identify the subject among people living in Limerick by only using this attribute). Thus, the higher the obfuscation level associated with an attribute, the lesser the impact on the subject if the attribute value is disclosed inappropriately.

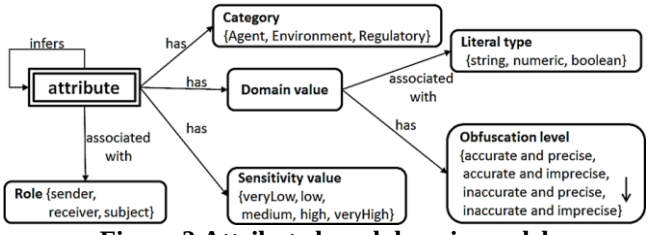


Figure 2 Attribute based domain model

As shown in Figure 2, we consider four different levels of obfuscation. The least level of obfuscation is associated with attributes whose value is precise and accurate.

B. Relating system behaviour to privacy policies

A behavioural model (FSM) shows the different state transitions as the system responds to events and actions triggered by changing context. To satisfy privacy awareness requirements, the key is then to identify attributes that are disclosed as a result of a state transition, and the transmission principles that should be respected during the transition. Given that transmission principles can be prescribed as privacy policies [20], an initial subset of attributes to monitor will consist of those that are both common to an operational context and privacy policies associated with a specific transmission principle. Subsequently, this set is further populated by other attributes having inference relationships with each identified attribute. The syntax and semantics for representing the privacy policies defining a particular transmission principle is defined as follows:

$$IF\ [\text{send (sender, M, receiver)}]\ \ THEN\langle\text{temporalOperator}\rangle\ (C_x)$$
$$UNLESS\ \langle\text{temporalOperator}\rangle\ (C_y)$$

where : Message: M $\vdash \langle$subject × attribute$\rangle$

$\langle$temporalOperator$\rangle \in$ {PREVIOUSLY, EVENTUALLY, LAST-TIME, NEXT-TIME, HENCEFORTH, ALL-TIME}

Condition: $C_i \subseteq \{\text{constraint}_1, \text{constraint}_2, \ldots, \text{constraint}_k\}$
and $\text{constraint}_j \rightarrow [(\text{att}_x \langle \text{arg}_1 \rangle \text{value})]$

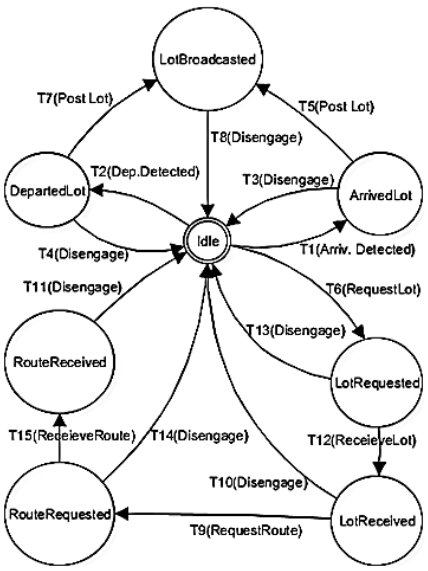
$\text{att}_x \subseteq$ {sender × attribute, receiver × attribute, subject × attribute}

$\langle \text{arg}_1 \rangle \subseteq \{=, <, >, \leq, \geq\}$

and value $\subseteq \{\text{value}_1, \text{value}_2, \ldots, \text{value}_x\}$ *and* value_j is a literal

This representation extends the logic of privacy and utility [18] [19] by defining IF-THEN-UNLESS rules over an FSM. "IF" is associated with a message transmission between a sender and a receiver. The message (M) carries an attribute about a subject. "THEN" and "UNLESS" specify a set of temporal constraints to be satisfied to preserve privacy during the message transmission. Temporal constraints can be expressed in the past (through PREVIOUSLY and LAST-TIME temporal operators), in the future (through NEXT-TIME, HENCEFORTH, EVENTUALLY temporal operators), and in all states (through ALL-TIME operator). Note that constraints are expressed in terms of relations between the attribute of a message and a specific value.

The semantics of defining IF-THEN-UNLESS rules over an FSM is shown in Figure 3. To keep the example simple, the states, transitions and disclosed attributes shown in are not exhaustive. We have used an FSM based on our running example for this purpose. Each state transition (Ti) is associated with the attributes it discloses and its privacy policy (to represent the transmission principle). For example, the transition T7 from the state *DepartLot* to *LotBroadcasted* is triggered by the event *PostLot*. The disclosed at-



	Disclosed attributes	Policy
T1	arrivalTime	P1
T2	departureTime	P2
T3	-	-
T4	-	-
T5	noSpaces, carParkName	P3, P6
T6	currentLocName, currentLocCord	P5
T7	noSpaces, carParkName	P3, P6
T8	-	-

	Disclosed attributes	Policy
T9	currentLocName, currentLocCord	P5
T10	-	-
T11	-	-
T12	noSpaces, carParkName, arrivalTime	P1, P3, P6
T13	-	-
T14	-	-
T15	gpsRoute, carParkName	P4, P6

Policies

P1	Consent	IF[Send(sender, arrivalTime, receiver)] THEN **ALL-TIME**[subjectInformed = true]
P2	Consent	IF[Send (sender, departureTime, receiver)] THEN **PREVIOUSLY**[subjectInformed = true]
P3	Confidentiality	IF[Send (sender, noSpaces, receiver)] THEN **ALL-TIME**[currentTime < 16.00] UNLESS **PREVIOUSLY**[knownReceiver= true]
P4	Confidentiality	IF[Send (sender, gpsRoute, receiver)] THEN **ALL-TIME**[currentTime < 16.00]
P5	Confidentiality	IF[Send (sender, currentLocationName, receiver) ∧ (currentTime > 16.00)] THEN **PREVIOUSLY**[knownReceiver = true]
P6	Confidentiality	IF[Send (sender, carParkName, receiver)] THEN **ALL-TIME**[CurrentTime < 16.00] UNLESS **PREVIOUSLY**[knownReceiver= true]

Figure 3 Defining privacy policies over a mobile system state machine

tributes characterising this transition includes the number of empty spaces found by the subject, the destination car park name, the arrival time and name of the subject. P3 and P6 are the privacy policies that prescribe the transmission principle for T7.

Thus, assuming a driver departing a parking lot wishes to inform other drivers of the vacated lot (transition T7 from *DepartedLot* to *LotBroadcasted* of Figure 3), the set of monitored attributes required to satisfy privacy awareness requirement for T7 include: *noEmptySpaces*, *arrivalTime*, *CarParkName* and *departureTime*. Note that *departureTime* is a member of this set by inference relationship.

V. PRIVACY THREAT ANALYSIS

Given a set of monitored attributes, the next step involves reasoning about which operational context can generate privacy threats. We model mobile users as a group of interacting social agents in specific roles (sender, receiver or subject). These agents perform actions involving personal information in a given operational context. We assume here

Table 1 Interaction history of agents posting and requesting empty parking spaces

t	Operational context (c)	Message Transmission (mt)	Transition(state)
1	arrivalTime=7.00, carParkName=`Ross St.`, carParkLocCord=`57.9N 10.1W`, subjectInformed= `false`, noSpaces=?, currentTime=7.00, knownReceiver= `true`, gpsRoute=?, currentLocName= `Ross St.`, currentLocCord= `60.2N 19.3W`, departureTime=?	**Sender**:A3, **Receiver**:A2, **Subject**:A3 **Message**: **[**arrivalTime=7.00**]**	T1 (ArrivedLot)
2	arrivalTime=7.00, carParkName= `Ross St.`, carParkLocCord=`57.9N 10.1W`, subjectInformed= `true`, noSpaces =5, currentTime=7.00, knownReceiver= `true`, gpsRoute=?, currentLocName= `Ross St.`, currentLocCord= `60.2N 19.3W`, departureTime=?	**Sender**:A3, **Receiver**:A2, **Subject**:A3 **Message**: **[**noSpaces=5, carparkName=`Ross St.`**]**	T5 (LotBroadcasted)
3	-	-	T8 (Idle)
4	arrivalTime=?, carParkName=`?`, carParkLocCord=`57.9N 10.1W`, subjectInformed=`true`, noSpaces =?, currentTime=7.10, knownReceiver= `false`, gpsRoute=?, currentLocName= `Bow Av.`, currentLocCord= `56.2N 23.1W`, departureTime=?	**Sender**:A1, **Receiver**:A2, **Subject**:A1 **Message**: **[**currentLocName= `Bow Av.`, currentLocCord= `56.2N 23.1W`**]**	T6 (LotRequested)
5	arrivalTime=?, carParkName= `Ross St.`, carParkLocCord=`57.9N 10.1W`, subjectInformed= `false`, noSpaces =5, currentTime=7.15, knownReceiver= `true`, gpsRoute=?, currentLocName= `Park St.`, currentLocCord= `56.2N 19.3W`, departureTime=?	**Sender**:A2, **Receiver**:A1, **Subject**:A3 **Message**: **[**noSpaces=5, carparkName= `Ross St.` arrivalTime=7.00**]**	T6 (LotReceived)
6	arrivalTime=?, carParkName= `Ross St.`, carParkLocCord=`57.9N 10.1W`, subjectInformed= `false`, noSpaces =5, currentTime=7.16, knownReceiver= `true`, gpsRoute=?, currentLocName= `Park St.`, currentLocCord= `56.2N 19.3W`, departureTime=?	**Sender**:A1, **Receiver**:A2, **Subject**:A1 **Message**: **[** currentLocName= `Park St.`, currentLocCord= `56.2N 19.3W`**]**	T9 (RouteRequested)
7	arrivalTime=?, carParkName= `Ross St.`, carParkLocCord=`57.9N 10.1W`, subjectInformed= `false`, noSpaces =5, currentTime=7.19, knownReceiver= `true`, gpsRoute= `Park St.->Bee Av.->Ross St`, currentLocName= `Park St.`, currentLocCord= `56.2N 20.3W`, departureTime=?	**Sender**:A2, **Receiver**:A1, **Subject**:A3 **Message**: **[**gpsRoute= `Park St.-> Bee Av.-> Ross St.`, carparkName = `Ross St.`**]**	T15 (RouteReceived)
8	-	-	T11 (Idle)

that each agent is represented by an FSM instance that is used to model its behaviour across a sequence of operational contexts. A state transition is then triggered when an agent interacts with another agent by requesting or receiving information. Each agent can adjust privacy policies associated with its FSM instance based on specific preferences. Agents also have shared knowledge among their group members and keep memory of past interactions. In this way, a subject's decision to consider a specific information request or response as privacy threatening is based on two factors: the history of operational context and what knowledge other agents in the group have about the subject. On the whole, assuming an attribute is being transferred from a sending to a receiving agent as a result of a state transition, privacy threat analysis reasons over what associated agents may know about the subject over time, and if such knowledge will violate the subject's privacy policy.

Table 1 shows a history of interactions involving a group of agents (A1-A3) with a common objective of posting and requesting empty parking spaces. Every row in Table1 represents a message about a subject transferred between a sender and receiver, its operational context and the associated state transition of the subject's FSM instance. In this scenario, A3 mobile device senses arrival in a parking lot; A3 then checks around for available empty spaces in the parking lot and informs A2. Subsequently, A1 is about to arrive the city centre and request for empty nearby parking spaces from A2. In response, A2 informs A1 of parking details it had previously received from A3.

- $K_{agent}\, a_{subject} \Rightarrow a_{subject}$
  (Agents have no false knowledge)
- $K_{agent}\, a_{subject} \wedge K_{agent}\,(a_{subject} \Rightarrow b_{subject}) \Rightarrow K_{agent}\, b_{subject}$
  (Agents can deduce new knowledge from previous knowledge)
- $K_{agent}\, a_{subject} \Rightarrow K_{agent} K_{agent}\, a_{subject}$
  (Agents have awareness of their knowledge)
- $\neg K_{agent}\, a_{subject} \Rightarrow K_{agent} \neg K_{agent}\, a_{subject}$
  (Agents have awareness of their lack of knowledge)
- From $a_{subject}$ infer $K_{agent}\, a_{subject}$ (rule of inference)

Where $K_{agent}\, a_{subject}$ means "agent knows attribute a about subject"

Figure 4 Applying *S5* axioms to group agent interactions

Our privacy threats reasoning approach combines the *S5* axiomatic system [26] (shown in Figure 4) with LTL properties specified in privacy policies. A related approach for reasoning about confidentiality was proposed by Landtsheer and Lamsweerde [21]. We use *S5* to model the cumulative knowledge gained by agents about a subject across a history of interactions, while LTL properties are used to reason over the sequence of operational context associated with gained knowledge to discover privacy threats. For example, based on the interaction history in Table1 and assuming *S5*, a model of accumulative knowledge gained by agents about is represented as follows:

t1: $M, w_1 \vDash [K_{A2}(\text{arrivalTime})_{A3} \wedge \neg K_{A3} K_{A2}(\text{arrivalTime})_{A3}]$

t2: $M, w_2 \vDash [K_{A2}(\text{noSpaces} \wedge \text{carParkName})_{A3} \wedge K_{A3} K_{A2}(\text{noSpaces} \wedge \text{carParkName})_{A3} \wedge K_{A2}(\text{departureTime})_{A3} \wedge \neg K_{A3} K_{A2}(\text{departureTime})_{A3} \wedge K_{A2}(\text{carParkLocCord})_{A3} \wedge \neg K_{A3} K_{A2}(\text{carParkLocCord})_{A3}] + w_1$

t5: $M, w_3 \vDash [K_{A1}(\text{noSpaces} \wedge \text{carParkName} \wedge \text{arrivalTime})_{A3} \wedge \neg K_{A3} K_{A1}(\text{noSpaces} \wedge \text{carParkName} \wedge \text{arrivalTime})_{A3} \wedge K_{A1}(\text{departureTime})_{A3} \wedge \neg K_{A3} K_{A1}(\text{departureTime})_{A3} \wedge K_{A1}(\text{carParkLocCord})_{A3} \wedge \neg K_{A3} K_{A1}(\text{carParkLocCord})_{A3}] + w_2$

t7: $M, w_4 \vDash [K_{A1}(\text{gpsRoute})_{A3} \wedge \neg K_{A3} K_{A1}(\text{gpsRoute})_{A3}] + w_3$

At t1, A2 knows the arrivalTime of A3. Furthermore, A3 does not know that A2 knows her arrivalTime, this is because subjectInformed=false for the value of operational context at t1. Similarly for t2, t5 and t7. No additional knowledge is gained about A3 for the interaction t3, t4 and t6 and t8. Using the *S5* distribution axiom and based on the inference relation that *departureTime* $\Rightarrow$ *arrivalTime* $\cap$ *carparkName*, then both A2 and A1 gain additional knowledge of the departure time of A3 at t2 and t5 respectively. Furthermore, although subjectInformed=true for the value of operational context at t2, A3 still has no knowledge that A2 knows her departure time. This is because, even though *departureTime* is inferred by A2 when the knowledge gained at t1 is combined with t2, the message transmission at t2 suggests that only the *carparkName* of A3 is transmitted to A2. Similarly, given that carParkName $\Leftrightarrow$carParkLocCord, it also infers that A1 and A2 have knowledge of A3's *carParkLocCord*, while A3 has no knowledge that A1 and A3 knows her *carParkLocCord*.

Given the model of knowledge gained by agents about a subject, it is necessary to check that the sequence of operational context for which knowledge is gained by agents satisfies the privacy policy of the subject. The algorithm summarising our privacy threats reasoning approach is given in the appendix of this paper. Using the LTL properties of privacy policies, we generate two sets of privacy threats. The first set is an actual threats checklist, which contains threats resulting from existing sequence of operational context failing to satisfy privacy policies. The privacy policies that generate these threats are bounded by LTL properties expressed in the present or in the past (e.g., PREVIOUSLY, LAST-TIME and ALL-TIME).

The second set is a potential threats checklist containing threats that might occur in case an expected future constraint on agent behaviour and operational context is not satisfied. Such a checklist is populated by privacy policies bounded by LTL properties expressed in the present or future (e.g. NEXT-TIME, HENCEFORTH, EVENTUALLY and ALL-TIME). The potential threats checklist is used to regulate the future behaviour of agents in a group as a result of the knowledge they have attained about a subject. This checklist can decrease or increase depending on the associated LTL

Table 2 Results of privacy threats analysis involving agents posting and requesting parking spaces

Agents Knowledge	*satisfied*(*policy, c,* $w_{i(subj)}$, mt)
w_1	*satisfied*(P1, t1, w_1) = false
w_2	*satisfied*(P2, t2, w_2) = false *satisfied*(P3, t2, w_2) = true *satisfied*(P6, t2, w_2) = true
w_3	*satisfied*(P2, t2, w_3) = false *satisfied*(P3, t2, w_3) = true *satisfied*(P6, t2, w_3) = true
w_4	*satisfied*(P4, t2, w_4) = true *satisfied*(P6, t2, w_4) = true

property. The checklist decreases when the required LTL constraint is satisfied. Alternatively, when the constraint associated with a potential threat fails, then the potential threat is shifted to the actual threats checklist.

During privacy threats reasoning, attributes involved in inference relations are also bound by the constraints defined in privacy policies associated with deduced attributes. For example, at t2, the knowledge gained by agents about A3 (i.e. *carParkName* and *noSpaces*) necessitates that the operational context satisfies the policy P3 and P6 respectively. In addition, t2 is also bound by P2, this is because, the knowledge gained at t1 (*arrivalTime* of A3) can be aggregated with some information disclosed at t2 (*carParkName* of A3) to infer the *departureTime* of A3. Table 2 demonstrates to outcome of the privacy threat reasoning process using the interaction history in table 1.

VI. MITIGATION ACTION SELECTION

Our proposed approach enables the designer to define adaptation rules that select the appropriate action for privacy threat mitigation. We consider four different types of mitigation actions: *ignore*, *react*, *prevent* and *terminate*. A person can choose to *ignore* a threat if the expected severity is low. Conversely, a higher expected severity level might require a different action. In *react*, a person can allow the message to be transferred, but additional conditions need to be satisfied by the sender or receiver to ameliorate the consequence of the threat. *Prevent* involves a person simply objecting to the message transmission between the sender and the receiver. Finally, *terminate* is the action that is selected when threat severity is at the peak level. For this case, the person withdraws from associating with the group objective. The rest of this section describes how a mitigation action is selected in a given situation.

A. Defining utility functions

In our approach, designers define utility functions as the basis for decision making, similar to Tsauro and Kephart [27], and the expected utility value shows how severe is a privacy threat: the lower the utility value, the higher the severity level. The severity of a privacy threat is based on two factors: the sensitivity level (*SL*) and obfuscation level (*OL*) of disclosed information. For an attribute, the expected utility U is the weighted sum of two partial utilities U_1 and U_2, respectively, for the two contributing factors *SL* and *OL*:

Expected Utility: $U(V_a, G_a) = k_{Va}U_1(V_a) + k_{Ga}U_2(G_a)$

where: k_{Va} – Utility weight for sensitivity level of disclosed attribute

k_{Ga} – Utility weight for obfuscation level of disclosed attribute

$U_1(V_a)$ – Severity of threat for a given sensitivity level of disclosed attribute

$U_2(G_a)$ – Severity of threat for a given obfuscation level of disclosed attribute

Assigning weights to each partial utility function provides a means to indicate the impact of corresponding factor on the expected utility. For example, assuming utility weights of k_{SL}=0.4, and k_{OL}=0.6 are respectively assigned to the attribute *currentLocationName*. Then, the obfuscation level of *currentLocationName* is a larger determinant of the expected utility of privacy threat resulting from its inappropriate disclosure. Assigning a utility weight of 0 infers that the associated factor is irrelevant to expected utility. Thus, if k_{OL}=0 for attribute *noSpaces*, then irrespective of the obfuscation level, it does not increase or decrease the expected utility resulting from its inappropriate disclosure.

For our example, we defined a sample utility function for sensitivity and obfuscation levels. The partial utility function of the sensitivity level is $U_1 = \exp(-pSL)$, where SL is the sensitivity level and *p* is the damping factor indicating how utility decreases, depending on the sensitivity of transmitted information. On the other hand, $U_2 = \exp(OL)$, indicates how utility increases, depending on the obfuscation level (*OL*) of transmitted information.

B. Defining adaptation rules

Once the expected utility has been calculated, adaptation rules need to be defined to select the appropriate mitigation action for a specific privacy threat. Figure 5 illustrates contour maps of our example utility function for two different damping factors. Numbers on contours indicate utility values. Figure 5a shows the utility function when a designer configures a subject's relaxed view about the sensitivity of revealed information (*p*=1), while in Figure 5b the subject is more conservative (*p*=5). In the conservative case, the terminate zone is larger and pushes the other zones towards higher utility values in the bottom right of the map. In the conservative case, the system ignores privacy violation only for low *SL*s and high *OL*s.

A set of rules is defined based on these utility values zones to specify when to ignore, prevent, react or terminate an interaction. For our example, the following set of rules is defined by considering the conservative case:

if : $U(V_a, G_a) \geq 3.04$ *Then* IGNORE, *if* : $2.36 < U(V_a, G_a) \leq 3.04$ *Then* REACT

if : $1.69 < U(V_a, G_a) \leq 2.36$ *Then* PREVENT, *if* : $U(V_a, G_a) < 1.69$ *Then* TERMINATE

The system designer is able to define zones for the utility values, i.e. more levels in the contour map, in order to set more general or detailed adaptation rules. The damping factor and weights of partial utilities U_1 and U_2 are set by the designer, but can be adjusted by the user later at runtime.

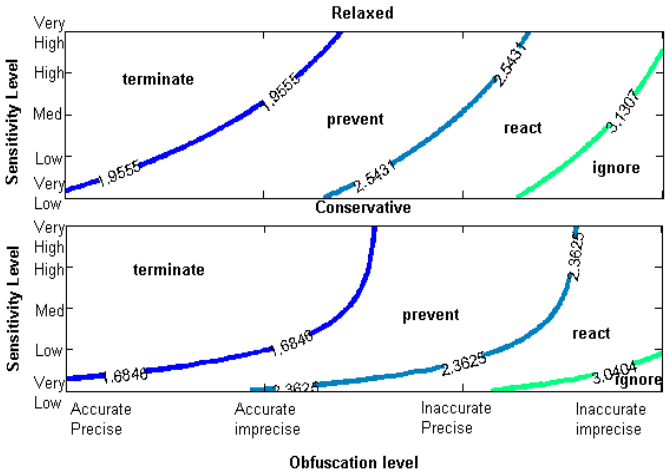


Figure 5- Contour map of the utility function a) relaxed b) conservative cases for information sensitivity level

These adaptation rules are defined for each attribute value in an operational context; e.g., *ArrivalTime*. But as shown in table 1, a message transmission in certain cases can involve multiple attributes. In this case, we use action fusion [25] for selecting the final mitigation action. This means that for each operational context attribute value, designers define a utility function, and adaptation rules to make decision based on each function separately. Then the system aggregates these decisions to determine the final one. For aggregation, the system considers the dominance relation between actions:

$$[terminate] \text{ dom } [prevent] \text{ dom } [react] \text{ dom } [ignore]$$

The system therefore selects the most dominant action as the final decision. For example, if we have three attributes and adaptation rules gives {ignore, ignore, prevent} in terms of utility values, the prevent action dominates the others. Thus, the system does not transmit the message.

Another factor that impacts the mitigation is the frequency of occurring privacy threats. The key assumption here is that if a specific threat continues to reoccur, then it is a pointer that the implemented mitigation measure is not strong enough. Thus, when a threat occurs several times in a specific span, depending on the associated utility value and frequency, the system might need to change its adaptation action. Hence, another set of adaptation rules is required to monitor the extent to which a mitigation action is able to curb a privacy threat. When a specific mitigation action is deemed insufficient, a stronger mitigation action is recommended. An illustration of such rule is as follows:

if : IGNORE and T_f >2 *Then* REACT, *if* : REACT and T_f > 2 *Then* PREVENT

if : PREVENT and T_f > 1 *Then* TERMINATE T_f – *frequency of threat*

VII. IMPLEMENTATION

We developed a proof-of-concept tool called *Caprice* – an environment for engineering adaptive privacy. *Caprice* is a plugin for the Microsoft .Net IDE and consists of three layers as shown in Figure 6. The modelling layer (layer 1) generates the domain, policy and behavioural model of the approach. The domain model is instantiated using an interpreter from a domain knowledge repository represented in OWL format. The policy model retrieves policy statements and associated LTL properties from a policy repository. These statements are then analysed by Prolog.Net - a Prolog engine for the .NET framework [28]. Finally, a behavioural model is generated using Microsoft Automatic Graph Layout [29] to instantiate an FSM of the system.

The second layer is composed of the operational context emulator, the FSM-Policy connector and the agent interaction simulator. The operational context emulator evaluates a sequence of operational contexts based on attributes defined in the domain model. The FSM-Policy Connector overlays FSM state transitions with privacy policies. Using the agent interaction simulator, *Caprice* can then simulate interaction between multiple agents. This is achieved by associating an FSM instance to each agent. Then a random message transfer between agents is simulated using a Monte Carlo simulation algorithm. It is also possible for the designer to customise the policies associated with each FSM instance representing an agent.

In the third layer, for every additional operational context simulated in the second layer, the designer is presented with a runtime view of FSM instances. This includes the possible privacy threats and mitigation actions that can be generated by the added operational context. The privacy awareness engine filters a valuated subset of monitored attributes from the operational context. This is based on inference rules specified in the domain model and constraints specified in the privacy policy. Subsequently, the privacy threats reasoner checks if the operational context of agent interaction satisfies the privacy policies of the associated subject. It does so by first building a model of knowledge gained interacting agents about the subject over a sequence of interactions. Based on associated LTL properties, the privacy threats reasoner then checks if the modelled knowledge will satisfy the privacy policies of the subject for

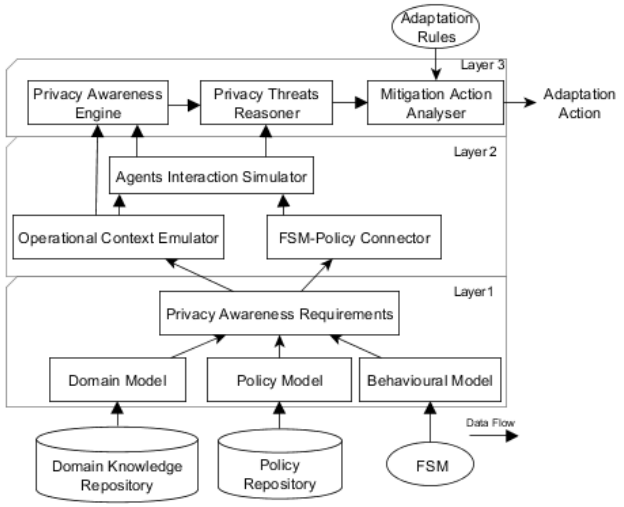


Figure 6 *Caprice* architecture

Table 3 Results of simulating interactions in Caprice at different time points

t	Privacy threating message transmission	Sensitivity levels	Obfuscation levels	Expected Utility	Fused Mitigation Action
20	**Sender**:A3, **Receiver**:A5, **Subject**:A4 **Message**: **[**carParkName= Moore Rd., arrivalTime=17.00, noSpaces=3**]** - T6	carParkName =low, arrivalTime = medium, noSpaces=low	accurate-imprecise accurate-precise accurate-precise	4.5656 3.2453 4.5656	IGNORE IGNORE →IGNORE IGNORE
78	**Sender**:A3, **Receiver**:A2, **Subject**:A4 **Message**: **[**carParkName= Kite St., arrivalTime=16.40, noSpaces=5**]** - T6	carParkName =low, arrivalTime = medium, noSpaces=low	accurate-imprecise accurate-precise accurate-precise	4.5656 3.2453 4.5656	IGNORE IGNORE →IGNORE IGNORE
87	**Sender**:A3, **Receiver**:A5, **Subject**:A4 **Message**:**[**carParkName= Moore Rd., arrivalTime=18.00, noSpaces=1**]** - T6	carParkName =low, arrivalTime = medium, noSpaces=low	accurate-imprecise accurate-precise accurate-precise	4.5656 3.2453 4.5656	IGNORE IGNORE →REACT IGNORE

that operational context. Finally, if the policies are not satisfied, the Mitigation Action Analyser recommends a possible mitigation action based on predefined adaptation rules.

VIII. EVALUATION

In section VI, we demonstrated how a designer can configure adaptation rules. In this section, we evaluate the viability of our approach from two other viewpoints. First, based on the adaptation rules previously defined in section VI, we validate the accuracy and correctness of our approach using *Caprice*. Secondly, we evaluate the efficiency and scalability of our approach. We then discuss the impact of our approach for software engineering of adaptive privacy.

A. Correctness

We used our motivating example to simulate a group interaction involving five agents (A1-A5) in *Caprice*. A total of 100 message transmissions in 100 time steps (t1-t100) were generated during the simulation across varying operational contexts. Each agent is associated with an instance of FSM. The privacy policy assignments were those shown in figure 3, and the adaptation rules in section VI were used.

The results of the simulation showed that our approach accurately detected every operational context that posed a privacy threat. The suggested mitigation actions also accurately reflected the defined adaptation rules. Table 3 shows the simulation results across three different time points where A4's privacy was threatened. The time points involved the agent A3 transmitting *carParkName*, *arrivalTime* and *noSpaces* associated with A4 to A5 (at t20 and t87) and A2 (at t78). In all cases, the sensitivity and obfuscation levels of monitored attributes remained the same. A notable point across these time points is the change in mitigation action at t87. This change is a result of the reoccurrence of the same threat (i.e. same sender and subject attributes) at t78 and t87. Thus, applying the adaption rules to corresponding threat occurrences at t87 results in scaling the fused mitigation action from IGNORE to REACT. As a result of the continuous reoccurrence of a threat initially ignored by A4, a stronger mitigation action is required.

B. Efficiency and scalability

We also evaluated the time complexity of our threat reasoning mechanism. The complexity depends on the different LTL properties that are used for threat reasoning. Hence, the objective was also to provide insights on the degree of complexity that each LTL property introduces into the privacy adaptation approach. The results of complexity analysis for our reasoning algorithm are shown in table 4. This analysis is based on the different LTL properties used for privacy threats reasoning. M represents the number of privacy policies associated with the state transition being analysed. N is the total number of instantiated operational contexts characterising the interaction history. We assumed a worst case where each operational context generates a potential privacy threat (i.e. the number of potential privacy threats is N).

The results in table 4 shows that based on asymptotic time complexity, reasoning over privacy threats using any of the LTL properties is computationally feasible and can be carried out in polynomial time. Reasoning using LTL properties such as PREVIOUSLY and ALL-TIME computationally grows at most quadratically. This is intuitive given that both properties involve reasoning over past behaviour. ALL-TIME involves reasoning over the whole history of operational contexts, and for the worst case scenario reasoning using PREVIOUSLY is equivalent to ALL-TIME. Furthermore, reasoning through other LTL properties such as HENCEFORTH, EVENTUALLY, NEXT-TIME and LAST-TIME has linearly complexity. For this case, only more recent operational context is analysed. Generally, using time complexity analysis, our privacy threats reasoning approach is efficient. The efficiency can further be improved by defining privacy policies using LTL properties whose growth is computationally linear.

Our approach is scalable even when networks involve hundreds or thousands of agents. For a worst case scenario, increasing the number of agents will result in increased history of operational context (i.e. increase in the value of N). We argue here that based on the results in table 4, our approach is computationally scalable to even larger interaction networks. This is because, irrespective of the value of N, the computation can be achieved in polynomial time. Our ap-

Table 4 Complexity of threat reasoning algorithm based for different LTL properties

LTL Property	O(g(N))	LTL Property	O(g(N))
HENCEFORTH	O(max(M,N))	LAST-TIME	O(M)
EVENTUALLY	O(MN)	PREVIOUSLY	O(MN)
NEXT-TIME	O(max(M,N))	ALL-TIME	O(MN)

proach also scales when the context attributes grows. This is due to the fact that our technique narrows the context space, by explicitly selecting the attributes to be evaluated to perform threat reasoning.

C. Discussion

Our approach relies on the designer to select and associate privacy policies to FSM state transitions. While this might pose an overhead to a generic software development process, we argue that this is worthwhile for investigating the functional properties of privacy critical applications. Furthermore, privacy policy, domain models and FSMs are models with which software designers are likely to be familiar. In our approach we are able to model user interactions and privacy concerns based on *S5* knowledge axioms and LTL. Thus, unambiguously identifying disclosed private attributes about a subject, and the operational context over which the sender disclosed such attributes to the receiver. This therefore lays the foundation for a privacy-oriented model-checking mechanism. Such a mechanism can potentially validate systems against privacy threats.

Our approach suggests threat mitigation actions to the designer by categorised adaptation actions -ignore, react, prevent and terminate based on the severity of privacy threat. We have not focused on the semantics of these categories and the details of how they can be implemented and applied to the system. We expect that an adaption action will require the designer to either implement an adaptation manager that applies the suggested mitigation action, or implement a feature that will enable the user carry out mediation actions. Either choice can remove/alter disclosure behaviour or ask the user to make context dependent decisions. Furthermore, although a participatory sensing system has been used to illustrate our approach, we suggest that our approach is generalisable to any other domain where disclosure can be modelled as the transfer of information between agents, such as social networks and energy consumption and distribution in a smart grid.

Removing or altering disclosure behaviour typically involves updating state transitions by adding or removing states in the FSM. Alternatively, privacy policies of the user can be refined. For example, at the time point 87 of table 3, reacting to the privacy threat can involve temporally truncating the state transition T6 for interaction involving agent A3. Alternatively, the privacy policy associated with T6 can be updated in a manner that accommodates the associated operational context. In other cases, the operational attributes of a state transition can be updated to reduce the lack of knowledge of the subject regarding an impending threat. Thus, based on the knowledge model presented in section IV A, the lack of knowledge by A3 at *w2* and *w3* can be ameliorated by updating the operational attributes of T5 for A3's FSM instance (i.e. Message: **[**noSpaces=?, carparkName=**?** departureTime **= ?]).** On the other hand, asking the user to make a context dependent decision creates a socio-technical challenge that can be addressed via privacy awareness. A more detailed insight into these mitigation measures is the subject of further work.

IX. Conclusion and further work

This paper has proposed a novel approach to engineering adaptive privacy. Our approach supports designers of context sensitive software systems in discovering runtime privacy threats and possible adaptation actions to mitigate the consequences of these threats. Our approach utilises privacy policies and domain and behavioural models of a system-to-be, combining these to derive a set of attributes to monitor. Based on these attributes, our approach enables the analysis of usage behaviour that can threaten privacy, and recommends actions to mitigate the threats. Recommended mitigation actions are based on the nature of disclosed information, and the frequency of occurrence of the threats. We implemented our approach in a tool called *Caprice*, which functions as a plugin in the .net development environment.

An evaluation of our work has given us confidence in its usefulness. We showed how adaptation rules can be configured to enable designers to examine different adaptation thresholds based on the expected severity of threat. By simulating a group interaction process involving a number of agents in *Caprice*, we demonstrated that our approach discovers plausible privacy threats and suggests appropriate mitigation actions. We also evaluated the time complexity of our privacy threat reasoning. Our results show that our analysis is computationally scalable and can be achieved in polynomial time, suggesting some promise of scalability.

Further work will investigate the semantics of the different categories of mitigation strategies we proposed in this paper. We have only considered agent interactions within a single group, and, for this reason, future work will focus on extending our approach for multiple groups.

Acknowledgements

This work was supported, in part, by SFI grant 10/CE/I1855 (for CSET2) to Lero, and by a Microsoft Software Engineering Innovation Foundation Award (2011).

References

[1] S. Lahlou, M. Langheinrich and C. Röcker, "Privacy and trust issues with invisible computers," Commun. ACM, vol. 48, no. 3, pp. 59-60, 2005.

[2] A. Acquisti, and J. Grossklags, "Privacy and Rationality in Individual Decision Making," IEEE Security and Privacy, vol. 3, no. 1, pp. 26-33, 2005.

[3] I. Altman, Privacy regulation: culturally universal or culturally specific? J. of Social Issues, 33:3, 66-84. 1977.

[4] L. Palen, and P. Dourish , Unpacking "privacy" for a networked world. In Proc. of the SIGCHI conf on Human factors in comp. sys, ACM, NY, USA, 129-136, 2003.

[5] G. Gay, Context-Aware Mobile Computing: Affordances of Space, Social Awareness, and Social Influence (Synthesis Lect on Human-Centered Info.), Morgan and Claypool Publishers, 2009.

[6] M. Langheinrich., A Privacy Awareness System for Ubiquitous Computing Environments, Ubiquitous Computing, Lecture Notes in Computer Science G. Borriello and L. Holmquist, eds., 315-320: Springer Berlin / Heidelberg, 2002.

[7] S. Spiekermann, and L. F. Cranor, Engineering Privacy, Sof. Eng., IEEE Trans on,.35(1) 67-82, 2009.
[8] A.B arth, A. Datta, J. Mitchell, and Nissenbaum, H., Privacy and Contextual Integrity: Framework and Applications. In Proc of the IEEE Symposium on Security and Privacy. IEEE Computer Society, Washington, DC, USA, 184-198, 2006.
[9] Beardsley E. L., Privacy & Personality, editors Pennock, J. R, and Chapman J. W., Transaction publishers, 2007.
[10] Deborah L. Estrin. 2010. Participatory sensing: applications and architecture. In Proc. MobiSys '10. ACM, New York, NY, USA, 3-4.
[11] Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. 2008. Addressing privacy requirements in system design: the PriS method. Requir. Eng. 13, 3 (August 2008), 241-255.
[12] A. Bijwe and N. R. Mead, .Adapting the SQUARE Process for Privacy Requirements Engineering, Technical Note, CMU/SEI-2010-TN-022, July 2010
[13] P. Dourish, and V. Bellotti, "Awareness and coordination in shared workspaces," in Proc. of the 1992 ACM conf on CSCW, Toronto, 1992, pp. 107-114.
[14] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness Requirements for Adaptive Systems," in Proc. SEAMS '11. ACM,2011, pp.60–69.
[15] D. Harel, Politi, Michal (1998). Modeling Reactive Systems with Statecharts, the STATEMATE Approach. McGraw-Hill. p. 258. ISBN 0-07-026205-5.
[16] H. Nissenbaum, Privacy as Contextual Integrity. Washington Law Review, Vol. 79, No. 1, 2004.
[17] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, Anand Ranganathan, Daniele Riboni, A survey of context modelling and reasoning techniques, Pervasive and Mobile Computing, Volume 6, Issue 2, April 2010, Pages 161-180.
[18] D. Kifer and B. Lin. 2010. Towards an axiomatization of statistical privacy and utility. In Proceedings of PODS '10. ACM, New York, NY, USA, 147-158.
[19] A. Barth, J. Mitchell, A. Datta, and S. Sundaram. 2007. Privacy and Utility in Business Processes. In Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF '07). IEEE Computer Society, Washington, DC, USA, 279-294.
[20] A. Datta, J. Blocki, N. Christin, H. DeYoung, D.Garg, Jia, L., Kaynar, D.K., and Sinha, A. Understanding and Protecting Privacy: Formal Semantics and Principled Audit Mechanisms. In Proceedings of ICISS. 2011, 1-27.
[21] R. De Landtsheer and A. van Lamsweerde. 2005. Reasoning about confidentiality at requirements engineering time. In Proceedings of the 13th ACM SIGSOFT ESEC/FSE-13. ACM, New York, NY, USA, 41-49.
[22] A.K Dey,: Understanding and Using Context. Personal Ubiquitous Computing 5(1), 4–7(2001).
[23] C. Nentwich, W. Emmerich, and A. Finkelstein. 2003. Consistency management with repair actions. In Proceedings of the 25th International Conference on Software Engineering (ICSE '03). IEEE Computer Society, Washington, DC, USA, 455-464.
[24] S. Fickas and M. S. Feather, "Requirements Monitoring in Dynamic Environments," in Proc. of the 2nd Int. Symposium on Req. Eng., 1995, pp. 140–147.
[25] M. Salehie and L. Tahvildari, "Towards a goal-driven approach to action selection in self-adaptive software," Software: Practice & Experience, vol. 42, no. 2, pp. 211–233, 2012.
[26] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, Reasoning about Knowledge. MIT Press, 1995.
[27] G. Tesauro and J. O. Kephart. 2004. Utility Functions in Autonomic Systems. In Proceedings of ICAC '04. IEEE Computer Society, Washington, DC, USA, 70-77.
[28] http://prolog.codeplex.com/
[29] L. Nachmanson, George Robertson, and Bongshin Lee. 2007. Drawing graphs with GLEE. In Proceedings of GD'07, Seok-Hee Hong, Takao Nishizeki, and Wu Quan (Eds.). Springer-Verlag, Berlin, Heidelberg, 389-394
[30] W. N. Robinson: A requirements monitoring framework for enterprise systems. Requir. Eng. 11(1): 17-41 (2006)

Appendix: Algorithm for reasoning over interaction history to discover privacy threats

```
Input: MessageTramsmission mt;
List OperationalContext; FSMTransition stateTransition;
AgentsKnowledgeModel w_i(subj); potentialThreatsCheckList;
Output: List potentialThreatCheckList; —(updated)
List actualThreatsCheckList;

int i = 1;
foreach PotentialThreat pt in potentialThreatCheckList do
    ContextInstance c = OperationalContext [i];
    if (satisfied(pt.policy,c,w_i(subj), pt.MessageTransmission) == false) and
       (pt.policy.TemporalProperty == (NEXT-TIME or
       ALL-TIME or HENCEFORTH)) then
        threatsCheckList.add(mt,tp,p);
        potentialThreatCheckList.remove(mt,tp,p);
    end
    else if (satisfied(pt.policy,c,w_i(subj), pt.MessageTransmission) == true)
    then
        potentialThreatCheckList.remove(mt,tp,p);
    end
end
foreach PrivacyPolicy p in stateTransition do
    tp = p.TemporalProperty;
    if (tp==NEXT-TIME) then
        potentialThreatCheckList.add(mt, tp, p);
        continue;
    end
    else if (tp== HENCEFORTH or ALL-TIME) then
        potentialThreatCheckList.add(mt, tp, p);
    end
    foreach ContextInstance c in OperationalContext do
        if (i=1 and tp==LAST-TIME ) then
            i = i + 1;
            continue;
        end
        if ((satisfied(p,c,w_i(subj), mt) == false) and (tp==EVENTUALLY)) then
            potentialThreatCheckList.add(mt,tp,p); break;
        end
        else if (satisfied(p,c,w_i(subj), mt) == false) then
            threatsCheckList.add(mt,tp,p);
        end
        if (tp==LAST-TIME or HENCEFORTH ) then
            break;
        end
        else if (tp==PREVIOUSLY and satisfied(p,c,w_i(subj), mt) == true) then
            break;
        end
    end
end
```