Facilitating Privacy Related Decisions in Different Privacy Contexts on the Internet by Evaluating Trust in Recipients of Private Data

Indrajit Ray and Sudip Chakraborty

Abstract Every time a user uses the Internet, a wealth of personal information is revealed, either voluntarily or involuntarily. This often causes privacy breaches, specially if the information is misused. Ideally, a user would like to make a reasoned decision about who to release her information to and what to release. For this purpose, we propose using the level of trust that a user has on the recipient regarding not to misuse her private data. To measure this trust level, we adapt the vector model of trust proposed earlier. We formalize a notion of privacy context and show how a privacy context ontology can be used to determine trust values for previously unencountered situations.

1 Introduction

Researchers are getting increasingly concerned about protecting the user's privacy in an electronic world. Unfortunately, most of us would find it difficult to provide a concrete definition of privacy with enough information to be able to apply it to our real lives. As individuals, each of us have unique needs and views of what constitute personal and private data [1]. The task is considerably more difficult when we have to define what privacy means to us as we use the Internet. Efforts to define and develop technologies that support the specification of consumer privacy requirements as well as help protect them, are evolving at a considerably slower pace. Efforts like the Platform for Privacy Preferences (P3P) Project of the World Wide Web consortium [4] and the related Privacy Bird project [3], and works based on the *k-anonymity* and *ℓ-diversity* models, provide solutions to some facets of electronic consumer privacy. For example, the P3P project attempts to provide a framework for service providers to express their privacy policies to the user with the goal that a user can form a reasoned opinion about the state of her privacy at the service provider. The related work on Privacy Bird [3] provides a user-friendly mechanism by which a user can determine if a service provider's P3P policies match the user's privacy preferences. The understanding is that such compliance will enhance the user's trust

Indrajit Ray and Sudip Chakraborty
Computer Science Department, Colorado State University, Fort Collins, CO 80523 e-mail: indrajit, sudip@cs.colostate.edu

Please use the following format when citing this chapter:

Ray, I. and Chakraborty, S., 2008, in IFIP International Federation for Information Processing, Volume 278; *Proceedings of the IFIP TC 11 23rd International Information Security Conference*; Sushil Jajodia, Pierangela Samarati, Stelvio Cimato; (Boston: Springer), pp. 605–619.

in the service provider. However, P3P is only able to provide a technical mechanism by which service providers can describe their use of personal information. P3P does not provide mechanisms by which policies are enforced. Nor can policies be used to verify or prove that the services accurately reflect the original policies. The *k-anonymity* model [12, 14], *ℓ-diversity* model [8] and similar works like [11], on statistical databases [5], and deductive databases [2] address the problem of releasing personal information so that the subjects of the data cannot be identified uniquely. Proponents argue that these efforts enable the users to act on what they see and thereby help protect their private information. However, often privacy is breached by factors that the users cannot see or control – for example, misjudged trust and misuse of information. Thus, these models and technologies solve only parts of the problem of protecting user privacy.

The last observation indicates that trust plays an important role in the problem of preserving privacy. Ultimately, the user needs to trust the recipient with her private information. A number of researchers have previously explored the idea of modeling privacy using a trust centric approach [6, 13, 9]. Goecks and Mynatt treat reputation and trust as separate independent entities and propose an approach to combine trust networks with reputation to provide privacy [6]. Shand et al. [13] rely on recommendation to direct the sharing of private information. Nguyen and Mynatt [9] address the problem of trust in pervasive computing environment. Their goal is to make the user more aware of privacy issues. The goal of enhancing consumer confidence in privacy practices of service providers has been explored by privacy seal programs such as TrustE (http://www.truste.org). However, it relies heavily on policy statements similar to P3P statement.

We believe that trust based approach to preserving privacy is promising. The problem with this group of work is that the trust models used are not very expressive. Moreover, none of these works discuss how to evaluate trust for the purpose of privacy preservation. In this work, we adapt and extend the vector model of trust we had proposed earlier [10] to help the user decide how much to trust the recipient of private data to preserve her privacy. We specify the user's different Internet activities like browsing a website, downloading content, purchasing, etc., as privacy contexts. The user is likely to have different privacy preferences for different contexts and may switch context anytime during an online session. Sometimes the user may not have enough information to calculate trust about a trustee in a new context. Or, the user may have no predefined preference rules in that context. We show how, in the above scenarios, the user can extrapolate a trust or a privacy preference rule-set using trust and preference rules for existing contexts. For this purpose, we define an ontology of privacy contexts containing a similarity relationship between different contexts. This similarity relationship is represented by a context similarity graph. Using the degree of similarity between contexts, the user can extrapolate trust or can set up privacy preferences in the context for which she does not have any a priori information.

The rest of the paper is organized as follows: Section 2 describes the vector trust model summarizing the main features and discussing our adaptation. Section 3 describes how this model can be used to evaluate the trust in the recipient for privacy

related issues. In section 4 we formalize the notion of privacy context. We show in section 5 how we can reason about privacy preferences and trust in different privacy contexts based on information about related contexts. Finally, we conclude in section 6 with some discussion on our future plans to extend this work.

2 The Trust Model

We adapt our previously proposed trust model [10]. Trust is specified as a trust relationship between a truster, A, a trustee, B, in a particular context, c and denoted as $(A \xrightarrow{c} B)$. The trust relationship is expressed as a vector where components are the parameters influencing trust. We identify three such parameters and express each of them in terms of a numeric value in the range $[-1,1] \cup \{\perp\}$. The three parameters may not have equal importance in determining a trust level. The *trust policy vector* specifies a normalization factor that gives the relative weight of each parameter. Applying the normalization factor to the trust relationship gives a *normalized trust relationship*, which we denote by $(A \xrightarrow{c} B)^N$. We also associate a numeric value in the range $[-1,1] \cup \{\perp\}$ to this normalized trust relationship. If the trust value is $1(-1)$ then we call the trustee completely trustworthy (untrustworthy). The trustee is semi-trustworthy (semi-untrustworthy) if the trust value is between $(0,1)$ $((-1,0))$. The truster professes a neutral trust level about the trustee if the trust value is 0. We use the symbol $\perp$ to denote an unknown trust value. We define the following properties of $\perp$. If $\mathbf{R}$ is the set of real numbers and $a \in \mathbf{R}$ then (i) $a \times \perp = \perp \times a = \perp$, (ii) $a + \perp = \perp + a = a$, (iii) $\perp + \perp = \perp$ and (iv) $\perp \times \perp = \perp$.

Trust Parameters

The three parameters for evaluating trust relationship – *properties*, *experience* and *recommendation* – are formalized as follows.

Properties:

Definition 1. The *properties* of the truster regarding a trustee for a particular context is defined as a measure of the characteristic attributes or information of the trustee for which the truster can have some assertion to be truly related to the trustee.

Each truster must initially define its own criteria for the gradation of properties regarding a particular entity. We can have *positive properties*, *negative properties*, and *neutral properties* where positive properties contribute towards increase in trust, negative properties contribute towards increase in distrust and neutral properties contribute neither way. To assign a value to the *properties* component, the truster must assign a value between -1 and $+1$ for each attribute of the trustee. This is done using a function, called *property evaluation function*. It is formally defined as,

Definition 2. Let p be a property. The property evaluation function of truster A, denoted by P_A is defined as a function that associates a value in the range $[-1,1] \cup \{\perp\}$ with the property p. Formally, $P_A(p) = v$, where $v \in [-1,1] \cup \{\perp\}$.

Different trustees may have properties that are not exactly same, but similar. For example, a trustee may use SSL as a communication protocol, whereas other uses TLS. Though the trustees use different protocols, they have similar properties as both of them use 'secure communication protocol'. However some trustee may not use any secure protocol at all during communication. To differentiate between these, we categorize properties of trustee into classes and subclasses where each class (or, subclass) has a finite set of possible items. For example, *communication protocol* used by the trustee is considered to be a property and thus can be represented as a class with possible items like SSL and TLS. The truster may choose to have subclasses within a property class. For example, within communication protocol, the truster may look for subclasses like *encryption algorithm* and *key-size*. The subclass key size may be, for example, represented by $\{[1-56],[57-128],[129-512],[513-1024]\}$. The truster needs to build these classes and subclasses according to her own criteria. To evaluate the property, the truster assigns value to each class (and subclass) as well as each of the attributes contained in them.

The truster A gathers the property information and evaluates it using P_A. If the truster is unable to obtain any information about the existence of any of the elements of a particular class (or subclass) CL, the class (or subclass) is considered to be empty and the truster assigns a value $\perp$ for the whole class (or subclass). Therefore, we extend the definition of property evaluation function for a class or subclass $CL = \{p_1, p_2, \ldots, p_n\}$ as

Definition 3. The property evaluation function extended for a class or subclass is denoted by $P_A : CL \rightarrow [-1,1] \cup \{\perp\}$ and is defined as

$$P_A(CL) = \begin{cases} \{v_1, v_2, \ldots, v_n\} & \text{where } \forall i, v_i = P_A(p_i) \\ \perp & \text{if } \forall i, P_A(p_i) = \perp \end{cases}$$

We do not dictate how a truster designs the function P_A. It will depend on the truster's domain knowledge, the scheme and trust evaluation criteria. Average of the property values gives the value for the component *properties*. If the truster is aware of k attributes of the trustee, then properties of trustee B according to truster A in context c is evaluated as ${}_AP_B^c = \frac{1}{k}\sum_{i=1}^{k} v_i$ where $v_i \in [-1,1] \cup \{\perp\}$, $\forall i = 1,2,\ldots,k$. v_i is the value assigned to i^{th} attribute of B. Note, ${}_AP_B^c = \perp$ is different from ${}_AP_B^c = 0$. Value 0 implies that after evaluating the information, the truster's decision is neutral. The value '$\perp$' implies "lack of information", i.e., there is not enough data to determine 'properties' of the trustee.

Experience:

Definition 4. The *experience* of a truster about a trustee is defined as the measure of the cumulative effect of a number of events that were encountered by the truster with respect to the trustee in a particular context and over a specified period of time.

The trust of a truster on a trustee can change because of the the truster's experiences with the trustee in the particular context. Each event that can influence the degree

of trust is interpreted by the truster as either a *positive event*, a *negative event* or a *neutral event*. We believe, events far back in time does not count as strongly as very recent events for computing trust values. Hence we introduce the concept of *experience policy* which specifies a length of time interval subdivided into non-overlapping intervals.

Definition 5. An *experience policy* specifies a totally ordered set of non-overlapping time intervals together with a set of non-negative weights corresponding to each element in the set of time intervals.

The whole time period $[t_0, t_n]$ is divided in such intervals and the truster A keeps a log of events occurring in these intervals. If e_k^i denote the k^{th} event in the i^{th} interval, then $v_k^i = +1$, if $e_k^i \in \mathbf{P}$, $v_k^i = -1$, if $e_k^i \in \mathbf{Q}$, and $v_k^i = 0$, if $e_k^i \in \mathbf{N}$, where **P** = set of all positive events, **Q** = set of all negative events and **N** = set of all neutral events.

The *incidents* I_j, corresponding to the j^{th} time interval is the sum of the values of all the events, positive, negative, or neutral for the time interval. If n_j is the number of events that occurred in the j^{th} time interval, then $I_j = \sum_{k=1}^{n_j} v_k^j$. If there is no event in $[t_{j-1}, t_j]$, then $I_j = \perp$.

Events far back in time does not count as strongly as very recent events for computing trust. We give more weight to events in recent time intervals than those in distant intervals. To accommodate this, we assign a *non-negative* weight w_i to the i^{th} interval such that $w_i > w_j$ whenever $i > j$. To ensure that we compute the weight w_i for the i^{th} interval as $w_i = \frac{i}{S}$, $\forall i = 1, 2, \ldots, n$, where $S = n(n+1)/2$. Then the *experience* of A with regards to B in context c is given by ${}_AE_B^c = (\sum_{k=1}^{n} w_k I_k)/(\sum_{k=1}^{n} n_k)$.

Recommendation:

Definition 6. A *recommendation* about a trustee is defined as a measure of the subjective or objective judgment of a recommender about the trustee to the truster.

We assume that the user is a member of a 'community' where each member can act as client (truster), service provider (trustee) or simply a third party (peer). The trust value of a truster on a trustee can change because of a *recommendation* for the trustee, provided by other members of the community. However, some members are more attributable to provide a 'good' feedback. Therefore truster partition the whole community in two different groups – attributable sources, having a trust relationship of certain level with the truster; and non-attributable sources, having no or very low valued trust relationship. For computing recommendation, the truster considers only those feedback that are provided by attributable sources. In section 3 we show how a truster choose the attributable sources.

A recommender sends her opinion or feedback about the trustee in the specified context, in terms of a numeric value in the range $[-1, 1] \cup \{\perp\}$. If Ψ is a group of n recommenders, $v(A \xrightarrow{c} r_j)^N \ \ (> 0)$ is trust value of j^{th} member (r_j) in Ψ, and $F_j = j^{th}$ recommender's feedback about trustee B in context c, then the truster A computes the recommendation component ${}_AR_B^c$ as, ${}_AR_B^c = (\sum_{j=1}^{n} [v(A \xrightarrow{c} r_j)^N \times F_j]) / \sum_{j=1}^{n} v(A \xrightarrow{c} r_j)^N$.

Note, the truster A has a trust relationship with the recommender r_j. We consider the context of this trust relationship as *negotiating*. Requesting for a recommendation and receiving it can be viewed as a negotiation where the truster is concerned about certain amount of her privacy. For example, the truster may be interested to share the following information with the recommender but does not want to make this public: (i) her identity (may be the IP address or other identifying information), (ii) identity of the trustee, (iii) the details of recommendation request, (iv) the context in which the trustee is being evaluated.

Scaling the recommendation score based on the trust relationship between the truster and the recommender has one important benefit. Suppose that the recommender tells a lie about the trustee in the recommendation in order to gain an advantage with the truster. If the truster does not have trust on the recommender to a great degree then the trust on the recommendation will be low with the truster.

Normalization

Having determined the values for each component of trust we specify the simple trust relationship between the truster A and the trustee B in a context c as $(A \xrightarrow{c} B)$. During evaluation of a trust value, a truster may assign different weights to the different factors that influence trust. The weights indicate relative importance of the parameters. We capture this factor using the concept of a *normalization policy* and is represented by a vector called *trust policy vector*.

Definition 7. The *trust policy vector*, ${}_AW_B$, is a vector that has the same number of components as there are parameters for influencing trust. The elements are real numbers in the range $[0,1]$ and the sum of all elements is equal to 1.

The normalized trust relationship $(A \xrightarrow{c} B)^N$ is obtained from the simple trust relationship, after combining the former by component-wise multiplication with the trust policy vector. Suppose ${}_AW_B = [w_P,\ w_E,\ w_R]$ is the trust policy vector, where $w_P, w_E, w_R \in [0,1]$ and $w_P + w_E + w_R = 1$. Then $(A \xrightarrow{c} B)^N = {}_AW_B \cdot (A \xrightarrow{c} B) = [w_P \times_A P_B^c,\ w_E \times_A E_B^c,\ w_R \times_A R_B^c]$.

Value of Trust Relationship

The values of the components of normalized trust relationship $(A \xrightarrow{c} B)^N$ are added to obtain the value corresponding to the trust relationship. This value is denoted by $v(A \xrightarrow{c} B)^N$. Formally, $v(A \xrightarrow{c} B)^N = {}_A\hat{P}_B^c + {}_A\hat{E}_B^c + {}_A\hat{R}_B^c$ where ${}_A\hat{P}_B^c = w_P \times_A P_B^c$, ${}_A\hat{E}_B^c = w_E \times_A E_B^c$, and ${}_A\hat{R}_B^c = w_R \times_A R_B^c$.

3 Preserving Privacy Using The Trust Model

We look into the privacy preservation scheme from a client's perspective. That is, we investigate how a user can have a reasonable control over her privacy while interacting with a server. Before each transaction, a user evaluates the trustworthiness of the server using the trust model described in section 2. To evaluate this trust the client

uses information about characteristics of the server, her personal experience with the server, and information that she gathers from other members in the community. In the following sections, we describe how properties, experience, and recommendation can be evaluated.

Evaluating *Properties*

To quantify the 'properties' component of the trust relationship, the client *A* (truster) needs to gather information about the attributes of the server *B* (trustee) and classify them. We give some examples of classes and subclasses of attributes of a trustee that a truster may define to evaluate properties component. (a) *Communication protocol* – Presence of a secure communication protocol like SSL helps preventing confidentiality breach, integrity breach, identity theft and thereby can prevent other indirect violations of privacy. In this class the truster may have the following subclasses: (i) Encryption algorithm – which encryption algorithm (e.g., AES or DES or RSA) is being used, (ii) Key-type – what type of key (e.g., symmetric key or asymmetric key) is used, (iii) key-size – what is the key size (e.g., 56-bit or 128-bit or 512-bit), (iv) Message digest algorithm – what type of message digest algorithm is used (e.g., MD5 or SHA), (v) Authentication – what authentication mode is used (e.g., authentication of both ends, or only *B*'s authentication or, it is totally anonymous), (vi) Key exchange – which key exchange algorithm is used (e.g., RSA or Diffie-Hellman), (b) *Credential* – Presence of a certificate from a well-known certifying authority (e.g., Verisign) about policies, methods and tools applied and used by *B* in a particular transaction. The truster *A* can have following subclasses: (i) Certifying authority – who the certifying authority is (i.e., how well-known the certifying authority is), (ii) Validation period – how long the certificate is valid. For example if it is an old certificate and is still valid for sufficiently long, then that would create a positive impression about *B*.

Once some or all of these information are available, *A* evaluates 'properties', according to her own policies, as described in section 2.

Evaluating *Experience*

Most of the information that goes toward forming the properties of the trustee *B* in a particular privacy context by itself does not necessarily enhance/diminish the truster's trust on *B*. This is because majority of the above criteria are examples of self-assertions. There is no guarantee that *B* conforms to these self-assertions. *B*'s behavior as an entity (it includes behavior as a service provider, recommender or just as a community member) in a transaction manifests in the form of *events*. If there are events that conforms to the properties that *A* has gathered then these events will be termed positive. If the events are contrary to the properties then they are negative. A false or misleading recommendation is also a negative event. Otherwise the events are neutral.

Categorizing an event to positive or negative depends on the truster *A*'s policy, specific activities and violations. Experience is computed by counting how many times (i.e., in how many events) *B* has deviated from or conformed to self-assertions or provided wrong information. During a specific period of time, number of devia-

tions from the stated self-assertions give number of negative events in that period. The events where B adhered to the self-assertions or provided correct feedback generate positive events.

Evaluating *Recommendation*

As mentioned earlier, evaluation of recommendation involves measuring the feedback provided by other members in the community. Note, however, a group of malicious members can send false good/bad reviews about the server (trustee) to influence the trust decision of the client (truster). The server may or may not be a member of that malicious group. To diminish the effect of such collusion while computing the recommendation, that is to select 'attributable sources' from the whole community, we propose the concept of 'trusted neighbors'. Note, we do not use the term 'neighbor' to mean the physical distance (in terms of length or hop) of a member from the client. We intend to measure how 'close' the member is with the client in terms of trust relationship. We now discuss how a truster builds this set.

Trusted Neighbors

Let there be m members in the community M. To choose the trusted neighbor set, a truster A sets up a neighbor_trust threshold τ_A^{nbr}. Then A broadcasts a ('neighbor_invitation') message to each of the members with whom A has a trust relationship and the value of the trust relationship is $\geq \tau_A^{nbr}$. A considers all members as her trusted neighbor from whom she gets back acceptance message. Therefore 'trusted neighbors' can be defined as

Definition 8. The trusted neighbors of a truster A is the set $TNBR_A$ of all members j where the trust value of j as evaluated by A is greater than or equal to the neighbor_trust threshold set by A and A receives an acceptance of neighbor_invitation from j. Formally, we can write, $TNBR_A = \{j \in M |\ v(A \xrightarrow{c} j)^N \geq \tau_A^{nbr} \wedge A$ receives acceptance of neighbor_invitation$\}$.

The next algorithm formally describes the process of creating trusted neighbor set.

Algorithm 1 *Get the trusted neighbors*
Input: *(i) M – the community of members, (ii) A – the truster whose neighbor set is to be determined, (iii) $\tau_A^{nbr}(>0)$ – neighbor_trust threshold set by A*
Output: *$TNBR_A$ – set of trusted neighbors of A*

Procedure $FindTrustedNeighbor(M, A, \tau_A^{nbr})$
begin
  $TNBR_A = \{\}$;
  for *each* $j \in M$
    if $v(A \xrightarrow{c} j)^N \geq \tau_A^{nbr}$
      Send 'neighbor_invitation' message to j;
      if *A receives an acceptance of neighbor_invitation from j*
        $TNBR_A = TNBR_A \cup \{j\}$;
  return $TNBR_A$;
end

After computing the trust, the truster checks the privacy policy of the trustee. If it conforms with her privacy preferences in that context, then she controls the disclosure of her information based on the evaluated degree of trust.

4 Privacy Context

As mentioned in section 2, a trust relationship between A and B is never absolute. In privacy platform, a user's trust on another user (service provider or recommender) will depend on how the other user is capable of keeping A's privacy in a specific context. For example, A (truster) can trust the entity B (trustee) to protect her private information collected during a registration procedure. However, that does not necessarily mean that A also trusts B to protect the private information collected while A is making a purchase from B. This leads us to associate a notion of *privacy context* with a trust relationship.

A user typically performs different activities during an online session. These activities can be categorized by their type. We denote each type as a 'context' of user activity. For example, a user may *search* for some document, and when found, she may *download* the corresponding file. The above involves two different contexts of activities, 'searching' and 'downloading'. Some examples of context are *Browse, Download, Purchase, Register, Log-in* etc. We assume the universe of contexts is finite. We observe that context should be defined such that the model is interoperable. Different entities often use different words to describe the same context. Alternately, the same word can be used for describing different contexts. These are example of semantic conflicts in the use of terminology. To solve these problems we borrow some ideas from the work on ontologies [7, 15]. Next, we present our privacy context ontology.

Privacy Context Ontology

Our ontology consists of a set of contexts together with relationships defined among them. First, we formally define the privacy context and later define the relationships between them.

Definition 9. A privacy context C is represented by a set of semantically equivalent keywords, denoted by $keywords(C)$.

Each keyword in $keywords(C)$ is used to describe the privacy context C. The keywords in $keywords(C)$ are semantically equivalent because they express the same context. For each context C we assume that the set $keywords(C)$ is non-empty and finite. Also, for any two semantically distinct privacy contexts C_1 and C_2, we require $keywords(C_1) \cap keywords(C_2) = \emptyset$. That is, any keyword belongs to exactly one context.

We give an example to illustrate the notion of privacy context. Consider the usual registration process in an Web-service. Some sites call it *registration*, some call it *register*, and some sites specify it as *sign-up*. All these different terminologies describe the same process. Therefore we specify any of these privacy contexts by the keyword set {register, registration, sign-up}. Using this notion, we define equality of two contexts as

Definition 10. Two privacy contexts C and C' are said to be equal, denoted by $C = C'$, if and only if $keywords(C) = keywords(C')$.

In the above example, the privacy contexts *register* and *sign-up* are equal.

Relationship Between Privacy Contexts

We define a relation called 'similarity' between distinct privacy contexts. This relation is defined for every pair of contexts in the privacy context set. The similarity relation is reflexive, symmetric, but not transitive. Each similarity relationship is associated with a *degree of similarity*. For two contexts C and C', we denote the degree of similarity by the symbol $sim(C, C')$. This indicates the semantic closeness of the two contexts. Since two distinct privacy contexts related by similarity are not exactly identical, the degree of similarity is denoted as a fraction. The exact value of the fraction is determined by the truster using her domain knowledge. Therefore, for two privacy contexts C and C' we have,

$$sim(C, C') = \begin{cases} 1 & \text{if } C = C' \\ 0 & \text{if } C \text{ and } C' \text{ are unrelated} \\ d \in (0,1) & \text{otherwise} \end{cases}$$

The similarity relationship will be used in setting up privacy preference rule-set when there is no such preference available in the privacy preference repository for a new context. It will also be used to calculate the initial trust about the trustee on that new context. The degree of similarity together with the trust on the entity in known privacy context will be used to extrapolate the trust on the entity in the new privacy context.

Privacy Context Similarity Graph

The privacy contexts and the similarity relationships between them is represented using a single graph which we refer to as the *privacy context similarity graph*. Each node n_i in the graph corresponds to a context C and is labeled by the set $keywords(C)$. We draw a weighted, undirected edge between two nodes n_i and n_j if degree of similarity between the corresponding contexts is between $(0,1)$. The weight on the edge indicates the degree of similarity between the nodes n_i and n_j. We formally define privacy context similarity graph as

Definition 11. A privacy context similarity graph $PCSG = \langle \mathcal{N}, \mathcal{E} \rangle$ is a weighted undirected graph satisfying the following conditions

1. $\mathcal{N}$ is a set of nodes where each node n_i is associated with a privacy context C_i and is labeled with $keywords(C_i)$, which is the set of keywords associated with the privacy context C_i.
2. For each edge $(n_i, n_j) \in \mathcal{E}$, the weight on the edge (n_i, n_j), denoted by $w(n_i, n_j)$, is in $(0,1)$ and equals to $sim(C_i, C_j)$, where C_i and C_j are represented by n_i and n_j respectively.

Figure 1 gives an example of privacy context similarity graph that involves four privacy contexts – *Browse*, *Search*, *Register* and *Log-in*. The weights are assigned by the truster according to her domain knowledge about these four contexts.

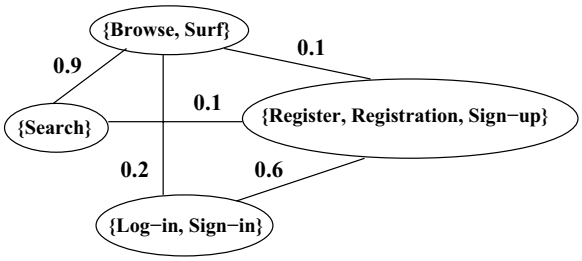

Fig. 1 An example privacy context similarity graph

5 Reasoning about Privacy Preferences and Trust in Different Privacy Contexts

A user (truster) is likely to have different privacy preferences for different privacy contexts, that is user's privacy preferences depend on underlying contexts. In other words, the user will perform certain actions, during a communication with a trustee, in one context and other actions in a different context. For example, a user may disclose her address information while making a 'purchase', but not when she is just 'searching' or 'downloading' something on or from the Web. Such actions, that are to be performed during a communication in a particular privacy context, are specified by the user, according to her own policies, as a set of rules. We call this rule-set for a particular privacy context as *privacy preference rule-set* and formally define it as

Definition 12. A user's privacy preference rule-set for a privacy context C, denoted by $\mathscr{R}_C$, is a set of rules regarding the actions or steps to be performed by the user (truster) when interacting with another entity (trustee) in the privacy context C.

A user (truster) can add/delete/modify these rule-sets according to her own policies. The privacy context keeps switching as the user continues her online activities, thereby continuously changing her privacy preferences. A user maintains a *privacy preference repository* where she keeps her privacy preference rule-sets for entities (trustees) in specific privacy contexts. However, the user may have a privacy preference rule-set for an entity in some context C, but may not have any preference rule-set in context C' in the repository. In this scenario the user, while interacting with that entity, needs to make decision about using some existing privacy preference rule-set.

A user also maintains a trust repository where she keeps the trust about entities in different privacy contexts. For a particular trustee, the user will not have a trust in a new privacy context, irrespective of whether she has or does not have a privacy preference rule-set for that context. After setting up a rule-set the user needs to initiate a trust relationship with the trustee in the new privacy context. This initial trust is calculated using the trust on the trustee in some existing privacy context.

Using an existing privacy preference rule-set from the repository when encountering a new privacy context, or an existing privacy context for which no rule-set

is available is reasonable only when the new context is 'similar' to the context for which a privacy preference rule-set is available in the repository. This is also true when extrapolating the initial trust in the new privacy context. The initial trust should be calculated from the trust on the trustee in some 'similar' privacy context available in trust repository. A privacy context may be related to several other privacy contexts through 'similarity' relationship. Nonetheless, we need to find out which context or set of contexts is conceptually closest to the given context. In other words, we need to find the privacy context or set of privacy contexts that has the highest similarity degree with the given privacy context. For this, we first define the concept of *closest privacy context*.

Definition 13. Let C be a privacy context. The set of privacy contexts $\mathscr{C}(C) = \{C_1, \ldots, C_n\}$ is defined to be *closest* to C if the following conditions hold:

1. for all $i \neq j, 1 \leq i, j \leq n$, $sim(C, C_i) = sim(C, C_j)$
2. for all $i = 1, \ldots, n$, $sim(C, C_i) = max(sim(C, C'))$, where C' is any privacy context that is related to the privacy context C.

Note, the set $\mathscr{C}(C)$ can be a singleton set. The following algorithm describes the method for finding the closest privacy context(s) of a given privacy context.

Algorithm 2 *Get the closest privacy context*
Input: *(i) C – the privacy context whose closest one needs to be determined. (ii) PCSG – the privacy context similarity graph in which C is a privacy context*
Output: $\mathscr{C}(C)$ *– set of privacy contexts closest to C*

Procedure $FindClosestContext(C, PCSG)$
begin
  $\mathscr{C}(C) = \{\}$, $relatedContext(C) = \{\}$;
  for *each* $C_i \in PCSG$
    if *there is an edge between nodes corresponding to C and* C_i *in PCSG*
      $relatedContext(C) = relatedContext(C) \cup \{C_i\}$;
  for *each* $C_j \in relatedContext(C)$
    if $sim(C, C_j) = max(sim(C, C_k))$ *where* $C_k \in relatedContext(C)$
      $\mathscr{C}(C) = \mathscr{C}(C) \cup \{C_j\}$;
  return $\mathscr{C}(C)$;
end

Example 1. Consider the privacy context similarity graph shown in figure 1. Suppose a user *Alice* wants to find the closest contexts of the privacy context {Log-in, Sign-in}. The *relatedContext* set for this privacy context is {{Browse, Surf}, {Register, Registration, Sign-up}}. The graph shows that *sim*({Log-in, Sign-in}, {Browse, Surf}) = 0.2 and *sim*({Log-in, Sign-in}, {Register, Registration, Sign-up}) = 0.6. Therefore, $\mathscr{C}$({Log-in, Sign-in}) is found to be the context {Register, Registration, Sign-up} as it has highest similarity degree with the privacy context {Log-in, Sign-in}.

If the privacy context similarity graph *PCSG* has n nodes, then the node corresponding to the privacy context C can be related to at most $n-1$ nodes in the graph. Therefore, at most $n-1$ edges can be in the set $relatedContext(C)$, from which the closest privacy contexts are determined. Thus the algorithm has complexity $O(n)$,

where n is the number of nodes in the privacy context similarity graph *PCSG*. However, note, if C was not present in *PCSG*, then the truster needs to update the existing *PCSG* by including a node corresponding to C and determining the weighted edges between the new node and the existing nodes. This updated *PCSG* is then used in the above algorithm 2 to find $\mathscr{C}(C)$.

Extrapolating Privacy Preferences from Similar Privacy Contexts

When a user A does not have privacy preferences in a particular privacy context C, we show how she can select one such preference rule-set using one or more similar privacy contexts. Suppose the user A encounters a privacy context C with an entity B and A does not have a privacy preference rule-set for C in the repository. A finds the set of closest privacy context $\mathscr{C}(C)$ using the algorithm 2. If $\mathscr{C}(C)$ is a singleton set, say $\{C'\}$, then the preference rule-set corresponding to C' is retrieved from the repository and set for context C. Now, suppose $\mathscr{C}(C) = \{C_1, C_2, \ldots, C_k\}$ i.e., $\mathscr{C}(C)$ is not a singleton set. Suppose for all $i = 1, \ldots, k$, $\mathscr{R}_{C_i}$ is the privacy preference rule-set corresponding to privacy context C_i. The user A has two choices in this case to set the rule-set for C. She can choose an $\mathscr{R}_{C_i}$ arbitrarily from the available $\mathscr{R}_{C_i}$s. Alternatively, she constructs the rule-set by taking union of all available $\mathscr{R}_{C_i}$s. Algorithm 3 describes the method.

Algorithm 3 *Extrapolate privacy preference rule-set for a new privacy context*
Input: *(i) C – the privacy context for which preference rule-set needs to be set (ii) PCSG – the privacy context similarity graph in which C is a context (iii) The privacy preference rule-set repository* $\mathscr{R}$
Output: $\mathscr{R}_C$ *– privacy preference rule-set for privacy context C*

Procedure $ConstructPreferenceRules(C, PCSG, \mathscr{R})$
begin
  $\mathscr{R}_C = \{\}$; $\mathscr{C}(C) = FindClosestContext(C, PCSG)$;
  if $\mathscr{C}(C) = \{C'\}$
    if $\mathscr{R}_{C'} \in \mathscr{R}$
      $\mathscr{R}_C = \mathscr{R}_{C'}$;
    else *exit;*
  if $\mathscr{C}(C) = \{C_1, C_2, \ldots, C_k\}$
    Case 1: $\mathscr{R}_C = \mathscr{R}_{C_j}$ *for an arbitrary* j *such that* $1 \leq j \leq k$ *and* $\mathscr{R}_{C_j} \in \mathscr{R}$;
    Case 2: $\mathscr{R}_C = \bigcup \mathscr{R}_{C_j}$ *for all* $1 \leq j \leq k$ *such that* $\mathscr{R}_{C_j} \in \mathscr{R}$;
  return $\mathscr{R}_C$;
end

Extrapolating Trust from Similar Privacy Contexts

As mentioned earlier, if a truster A does not have a trust about a trustee B in a privacy context C in her trust repository, then she calculates the initial trust about B in C using the similar privacy contexts of C. For this evaluation, we discuss two scenarios:

Scenario 1: $\mathscr{C}(C) = \{C'\}$ i.e., the closest privacy context set is a singleton set.

In this case A retrieves the normalized trust vector $(A \xrightarrow{C'} B)^N$ with B in privacy context C' and assigns the value $sim(C, C') \times v(A \xrightarrow{C'} B)^N$ as the initial value for the trust relationship $(A \xrightarrow{C} B)^N$. If $v(A \xrightarrow{C'} B)^N = \perp$, i.e., A has no information about trust on B in privacy context C', then she needs to extrapolate $(A \xrightarrow{C'} B)^N$.

Therefore, this extrapolation can be a recursive process.

Scenario 2: $\mathscr{C}(C) = \{C_1, C_2, \ldots, C_k\}$.

A retrieves all the normalized trust vectors $(A \xrightarrow{C_i} B)^N, \quad i = 1, 2, \ldots, k$. The initial value for the trust relationship $(A \xrightarrow{C} B)^N$ is calculated as $v(A \xrightarrow{C} B)^N = \frac{1}{k}\sum_{i=1}^{k}[sim(C, C_i) \times v(A \xrightarrow{C_i} B)^N]$. To illustrate the above, we continue with our earlier example.

Example 2. Let *Alice* now want to extrapolate her trust on www.Books.com in the privacy context {Log-in, Sign-in}. In our earlier example, *Alice* finds the closest privacy context of the privacy context {Log-in, Sign-in} as {Register, Registration, Sign-up}. For the sake of brevity, let us denote the privacy context {Log-in, Sign-in} by *Log-in* and the privacy context {Register, Registration, Sign-up} by *Register*. Suppose *Alice* has a trust relationship $(Alice \xrightarrow{Register} www.Books.com)^N$ with the server www.Books.com (trustee) in her trust repository. Suppose $v(Alice \xrightarrow{Register} www.Books.com)^N = 0.8$. Then the initial value of the trust relationship $(Alice \xrightarrow{Log-in} www.Books.com)^N$ is evaluated as $0.6 \times 0.8 = 0.48$.

6 Conclusion and Future Work

In this paper, we introduce a framework that helps the Internet user make reasoned decision regarding release of her private information. Using the framework, a user can make sound choice regarding to whom she should release, why to release, and to what extent, during a transaction in specific privacy context. The framework uses a trust model which is build upon our earlier proposed trust model. In this model trust is expressed as a numeric value within the range $[-1, 1] \cup \{\perp\}$. We identify parameters that influence this trust and propose methods to evaluate them. A mechanism to define relative importance of parameters is also proposed. We posit that a user switch privacy contexts during online sessions and is likely to have different privacy preferences for a trustee in different contexts. We also argue that it is possible that the user does not have a privacy preference as well as a trust in some privacy context. To solve this issue we define an ontology on privacy context. The ontology defines relationship between privacy contexts. This helps a user to extrapolate trust and privacy preferences for a new privacy context from existing privacy contexts.

We plan to extend this work in future. We are currently investigating how to define an operator to combine two privacy context similarity graphs. It will be useful for group of users, working collaboratively, to make privacy related decisions. We need to investigate the possibility of other relationships between contexts, for example generalization/specialization or composition, in the privacy context ontology. We also have plan to evaluate the model by implementing it and then analyzing its performance for privacy protection.

Acknowledgements This work was partially supported by the U.S. Air Force Office of Scientific Research under contract FA 9550-07-10042. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Air Force or other federal government agencies.

References

1. Ackerman, M., Cranor, L., Reagle, J.: Privacy in E-Commerce: Examining User Scenarios and Privacy Preferences. Communications of the ACM (1999)
2. Bonatti, P., Kraus, S.: Foundations on Secure Deductive Databases. IEEE Transactions on Knowledge and Data Engineering **7**(3), 406–422 (1995)
3. Corp., A.: Privacy Bird Project. http://www.privacybird.org
4. Cranor, L., et al.: The Platform for Privacy Preferences 1.1 (P3P 1.1). Tech. rep., World Wide Web Consortium (2004)
5. Domingo-Ferrer, J.: Inference Control in Statistical Databases from Theory to Practice. Volume 2316. Springer (2002)
6. Goecks, J., Mynatt, E.: Enabling Privacy Management in Ubiquitous Computing Environments Through Trust and Reputation. In: Proceedings of CSCW 2002 Workshop on Privacy in Digital Environments. New Orleans, LA (2002)
7. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition **5**(2), 199–220 (1993)
8. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data **1**(1), Article 3 (2007)
9. Nguyen, D., Mynatt, E.: Privacy Mirrors: Understanding and Shaping Socio-technical Ubiquitous Computing Systems. Technical Report GIT-GVU-02-16, Georgia Institute of Technology (2002)
10. Ray, I., Chakraborty, S.: A Vector Model of Trust for Developing Trustworthy Systems. In: Proceedings of the 9th European Symposium on Research in Computer Security, pp. 260–275. Sophia Antipolis, France (2004)
11. Samarati, P.: Protecting Respondents' Identities in Microdata Release. IEEE Transactions on Knowledge and Data Engineering (TKDE) **13**(6), 1010–1027 (2001)
12. Samarati, P., Sweeney, L.: Generalizing Data to Provide Anonymity When Disclosing Information. In: Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. Seattle, WA (1998)
13. Shand, B., Dimmock, N., Bacon, J.: Trust for Ubiquitous, Transparent Collaboration. In: Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications. Dallas, TX (2003)
14. Sweeney, L.: K-Anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuziness and Knowledge-based Systems **10**(5), 557–570 (2002)
15. Uschold, M., Grüninger, M.: Ontologies: Principles, Methods, and Applications. Knowledge Engineering Review **11**(2), 93–155 (1996)