SoSharP: Recommending Sharing Policies in Multiuser Privacy Scenarios

Users often share information about others; sometimes this inadvertently violates others' privacy. Thus, here the authors propose SoSharP, an agent-based approach to help users maintain their own and others' privacy by guiding a selection of sharing policies in multiuser scenarios. SoSharP learns incrementally and asks for users' input only when required, reducing users' effort.

Ricard L. Fogues
Universitat Politecnica de Valencia

Pradeep K. Murukannaiah
Rochester Institute of Technology

Jose M. Such
King's College London

Munindar P. Singh
North Carolina State University

Social network services (SNSs) enable users to share information and link with others through facilities such as tagging and mentioning. However, linking information presents a privacy risk for the linked users. Suppose Alice uploads a photo from a party in which she and her friend Bob appear, and tags Bob. Bob might find that the photo Alice uploaded is sensitive. But Bob has no control over that photo being uploaded, and Alice's action threatens Bob's privacy. We term such a situation a *multiuser privacy scenario* or, for short, a *multiuser scenario*.

Currently, SNSs leave the responsibility of setting an appropriate sharing policy for information being shared to the uploader. The uploader's choice might not be appropriate for other users, who might cope with the problem via strategies[1] such as self-censorship and tag-approval (offered by Facebook). However, these strategies might be ineffective in a multiuser scenario. First, by the time a user realizes that an inappropriate photo involving him was shared and untags himself, the sensitive information might have been disclosed. Second, even if the uploader wants to find a policy that respects each user's preference, doing so is tedious and nontrivial.

Multiuser scenarios are tailor-made for personal agents that make effective recommendations about sharing based on relevant features. Such agents help reduce the cognitive burden and social stress on users in making decisions that promote privacy.

SoSharP

We envision each SNS user as employing a *personal agent* to share information in a multiuser scenario. A personal agent interacts with its user and with personal agents of other users in the scenario, and recommends a sharing policy to its user.

Published by the IEEE Computer Society
1089-7801/17/$33.00 © 2017 IEEE

To realize this vision, we make two contributions.

SoSharP. This personal agent acts as a recommender for sharing policies. SoSharP addresses multiuser scenarios by doing the following: automatically learning to recommend a sharing policy from features about context, users, groups, and preferences; providing recommendations despite incomplete information; and improving recommendations by receiving user ratings, incrementally.

Bootstrapping SoSharP via crowdsourcing. A *cold start* — producing recommendations before obtaining information — is a major challenge for recommenders, including SoSharP. Because of the variety of multiuser scenarios, collecting sufficient data from users would be time consuming. Instead, we bootstrap SoSharP by *crowdsourcing* a dataset that includes diverse multiuser scenarios, each associated with a suitable sharing policy. We employ this dataset to train SoSharP so it can make recommendations right away, even before it has obtained data for a particular user.

Features

Given a multiuser scenario, SoSharP recommends a sharing policy — for example, *share with common friends* — suitable for all users involved. To do so, it exploits four types of features: contextual and preference-based features from our prior work,[2] and user and group-based features introduced here. These features are motivated by the privacy literature, information available in existing SNSs, and our intuition.

Context

The context of sharing influences users' privacy attitudes[3] and, consequently, sharing decisions in a multiuser scenario. SoSharP incorporates three key contextual factors in the recommendation process: First, it incorporates *relationships* among the individuals (for example, privacy attitudes might differ for family versus friends). Second, it incorporates the *sensitivity* of the information (for example, users might share less sensitive information more freely). Third, it incorporates the *sentiment* of the information (expressing emotions and eliciting emotional responses from others influences how users build social capital).

User Characteristics

SoSharP exploits research on privacy behavior on social media that shows that demographics, social media practices, and sharing behaviors influence how information is disclosed.

Age and gender. In general, younger users are freer than older users and men are freer than women in sharing information on social media.[4]

Education level. This is commonly provided by users to receive recommendations of friends and relationships.

Use frequency. Active users tend to generate more content and share frequently.

Sharing experience. Experienced users would have learned the implications of their privacy settings.

Conflict experience. Having dealt with conflicting multiuser scenarios can influence how users attempt to resolve sharing conflicts.

Sharing Preferences

SoSharP employs the following features, representing different combinations of sharing preferences that a multiuser scenario might present.

Most restrictive policy. This is the policy that shares the least. For instance, if the policies are *share with common friends* and *share with all*, the most restrictive policy is *share with common friends*.

Least restrictive policy. This is the opposite of *most restrictive policy*.

Majority policy. This is the policy (or policies in case of a tie) preferred by a majority of users.

Group Characteristics

SoSharP employs aggregate measures, specifically, the *mean*, *maximum*, *minimum*, *standard deviation*, and *range* (= *maximum* − *minimum*) of the individuals' characteristics, as group characteristics. We compute these measures for all characteristics except gender, which takes one of four values: all or majority of males or females.

Recommenders

SoSharP provides personalized recommendations without requiring extensive user input, thus reducing the burden on users. As Figure 1

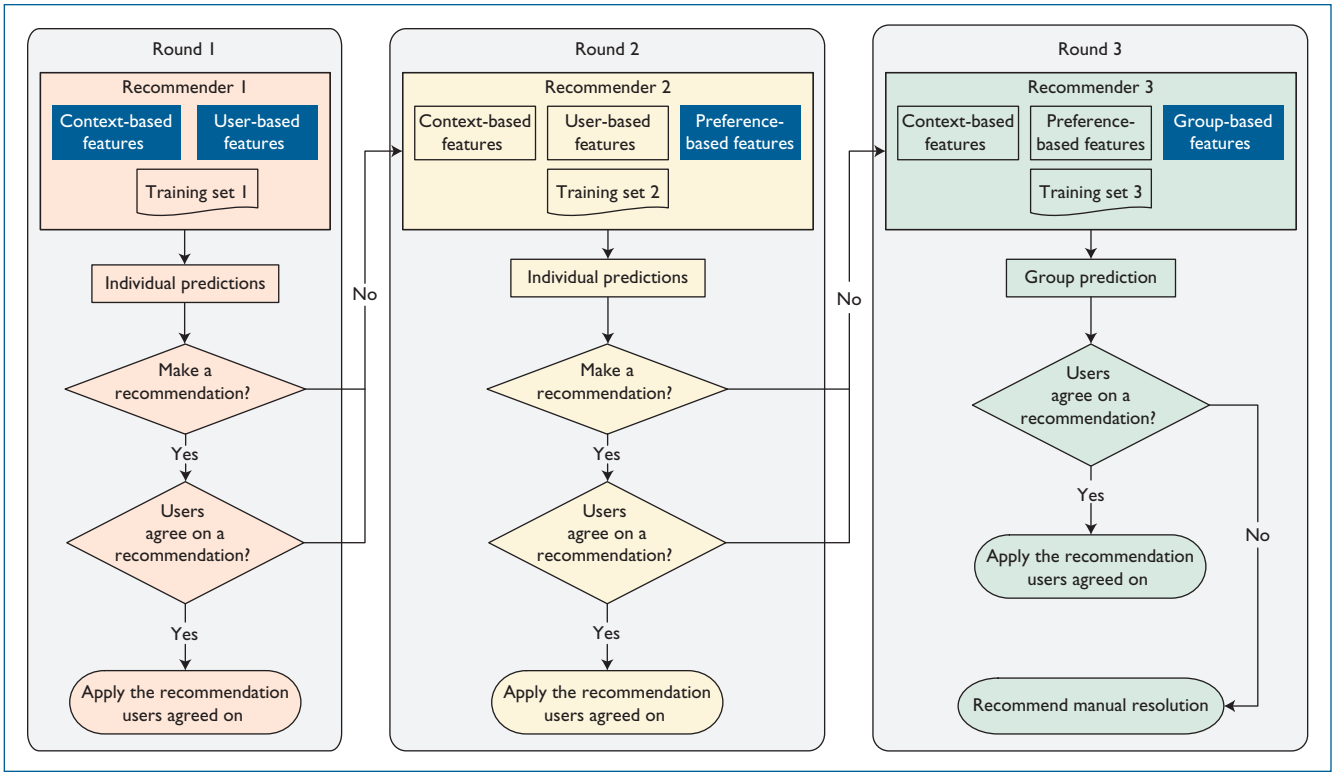

Figure 1. SoSharP's incremental recommendation approach, highlighting the information added in each round.

shows, SoSharP operates in up to three rounds, proceeding to the next round only if necessary to obtain additional information.

First round. In the first round, SoSharP employs only context and user features that can be obtained from an SNS (fully automatically), thereby avoiding user input. For example, age is usually required in SNSs. Additionally, an SNS can record how many conflicts were previously resolved through SoSharP — to determine the conflict experience of each party. Importantly, in this round, SoSharP generates recommendations even when some of the contextual features are unknown.

SoSharP makes potentially distinct recommendations to the users. Users might accept SoSharP's recommendations or choose another policy. Regardless, if all users agree on a sharing policy, SoSharP stops; otherwise, it moves to the next round.

Second round. In the second round, in addition to context, SoSharP employs users' preferred sharing policies, which can be the same as SoSharP's Round 1 recommendations, or different if a user didn't like the recommendation. From the individual users' preferred policies, we compute representative features of *most restrictive*, *least restrictive*, and *majority policy*. As before, SoSharP makes individual recommendations. It stops if all users agree on a policy or moves to the last round, otherwise.

Third round. In the third round, SoSharP generates a single recommendation for the whole group. To this end, SoSharP considers not individual, but group features. The recommendation becomes final if it's accepted unanimously; otherwise, SoSharP stops and lets the users proceed manually.

Adaptation. For each multiuser scenario, when SoSharP stops (irrespective of the round), it records the users' final decision and the features for that scenario. From such data instances, SoSharP learns a model on how groups of users make sharing decisions in disparate multiuser scenarios. It employs the learned model to make recommendations for the users in subsequent scenarios.

Bootstrapping SoSharP

SoSharP requires historical data about a user's multiuser scenarios to make recommendations for that user in a new scenario. Instead of waiting to accumulate historical data, which takes time and user effort, we bootstrap SoSharP from a training dataset acquired via crowdsourcing data. However, SoSharP continually enhances the training dataset with user-specific scenarios and decisions, because of which its recommendations can adapt to users over time.

Crowdsourcing provides an opportunity to collect data from a number of users of diverse backgrounds quickly. It enables SoSharP to offer recommendations that a majority of people are likely to find as appropriate, when user-specific data aren't available.

To acquire the crowdsourced training dataset, first, we present information about two or more individuals in a specific *scenario*: a combination of context and preferences. Then, we ask participants to choose a suitable sharing policy for that scenario. We ask participants to identify themselves with the people involved in the scenario, which is similar to the methodology employed in related work,[5] but for a multiuser scenario.

To create scenarios, we reuse the picture survey instrument from prior work,[2] where we employed the collected survey data to assess the influence of a feature on the final policy in a multiuser scenario. Here, we employ the collected data to bootstrap SoSharP.

Picture Survey

To generate picture surveys (Table 1 is an example), we combine context and sharing policies.

Context. First, we consider how the individuals in a picture are related via one of three predefined relationships: friends, family, or colleagues; second, we consider whether the pictures are sensitive or nonsensitive; and third, we consider whether the picture conveys a positive or negative sentiment. Doing so yields 12 contexts. We select a representative picture for each context. However, as Table 1 shows, we ask participants to identify the relationships among the people involved in the scenario, and rate sensitivity (using the Likert scale where 1 = not sensitive at all, and 5 = very sensitive), and sentiment (1 = extremely positive, and 5 = extremely negative). We build the training dataset considering participants' responses.

Sharing policy. A policy can imply no sharing, sharing publicly, or anything in between. Furthermore, sharing policies depend on the number of a user's contacts and their relationships. The space of possibilities is large. For simplicity, we bootstrap SoSharP considering only three disclosure levels (these three levels matched the Facebook's basic privacy settings at the time of our study):

1. *Share with all*. Anyone on the SNS can access the information — this is the least restrictive policy.
2. *Share among themselves*. Only those directly connected with the information can access it. Because the scenarios in our study always include individuals who are members of a group picture, this policy equals a policy to not share, and hence is the most restrictive.
3. *Share with common friends*. Only common friends of the individuals in the scenario can access the information. We assume that this policy is a reasonable intermediate between the previous two.

We limit multiuser scenarios in our study to involve three individuals so that a majority policy without ties is possible. Although some pictures show more than three individuals, our scenarios discuss the preferences of only three. To train SoSharP, we require scenarios with conflict. Thus, we make sure that not all three individuals in a scenario use the same preference.

Consequently, we generate 216 scenarios: 12 pictures based on context, three policy preferences for the first two individuals, and two preferences for the last (see the aforementioned restriction).

Following the picture, its description, and context identification, we ask participants to answer two questionnaires (Q1 and Q2), sequentially. Each of the two questionnaires tells participants that one of the individuals in the scenario wants to upload the picture to a social media account and asks what sharing policy they prefer. The participants choose a policy (*share with all*, *share with common friends*, or *share among themselves*). In Q1, participants know only the contextual attributes, but not the sharing preferences of the individuals in the

Table 1. Shortened example of a picture survey.[2]

Picture	
Description	Maria, Bonita, and Felipe, three junior employees in a company, attend a business lunch in which they meet their seniors.
Rating	Identify the relationship between Maria, Bonita, and Felipe and rate the sensitivity and sentiment of the picture.
Context (Q1)	Consider that Maria wants to upload this picture to her social media account. What sharing policy should she apply for this picture?
Preferences (Q2)	Next, consider users' preferences as follows: **Bonita** *share among ourselves* **Felipe** *share among ourselves* **Maria** *share with all* Considering the context and users' preferences, what sharing policy should Maria apply for this picture?

scenario. This case is similar to a real scenario where a user wants to share information without asking others potentially concerned with the information. Q2 introduces all the users' preferences.

Participants

We recruited participants from Amazon MTurk to answer picture surveys. We directed each participant to an external website, asking to complete a demographics presurvey, five picture surveys (with distinct pictures), and a postsurvey about the participant's general opinions about dealing with multiuser scenarios. The picture survey scenarios were randomized to counter ordering bias. Our study was approved by an institutional review board (IRB).

Quality Control

We required participants to complete at least 50 tasks on MTurk and to have a success rate of at least 90 percent. We included an attention check question[6] in the ratings section of each picture survey, asking how many people were present in the picture, because responding requires counting from the picture. If a participant incorrectly answered the attention check question in a picture survey, we excluded that survey from analysis (but paid the participant, anyway).

Datasets

We built three training datasets, one for each round of SoSharP (see Figure 1), from the MTurk study data.

- The *Round 1* training dataset uses participants' responses to Q1. Each response is an observation, whose features are the ratings of contextual factors and the characteristics of the participant providing that response.
- The *Round 2* training dataset uses participants' responses to Q2. Each observation includes features analogous to those in Round 1 and features computed from the preferences presented in the corresponding Q2 scenario. For example, if the scenario of a given response presented two users preferring *share with all* and one preferring *share with themselves*, the feature *majority policy* would be *share with all*.
- The *Round 3* training dataset employs each possible triplet of responses provided for Q2. To form a triplet, the three responses must share the contextual and preference features. We consider each triplet as an observation with the following features: contextual, preferences, and group characteristics.

The class (target) variable's value for each observation in the Rounds 1 and 2 datasets is the sharing policy chosen by the corresponding participant. For the Round 3 dataset, it's the majority policy in the response triplet or *share with common friends* in case of a tie. For example, if a triplet is formed by two responses that chose *share with all* and one that chose *share with common friends*, the class of the triplet is *share with all*.

Classifiers

From each of the datasets, we train a three-class (each sharing policy is a class) machine learning classifier.

In each round, SoSharP employs the classifier for the corresponding round to recommend a sharing policy.

We choose specific classifiers and tune their parameters based on empirical evaluations over data. Specifically, for Round 1, we train a *random forest* classifier with 150 trees; for Round 2, we train a *logistic regression* classifier; and for Round 3, we train a random forest classifier with 200 trees.

Evaluation

We gathered 3,767 valid picture survey responses, of which we use 70 percent (2,637) to build training datasets, and the other 30 percent (1,130) for testing. We form triplets from the 1,130 responses. All responses in a triplet must share the same contextual and preference features.

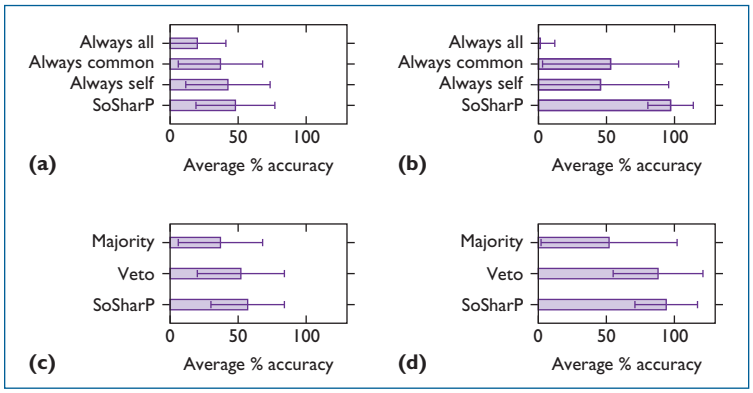

Figure 2. Accuracy comparisons between SoSharP, baseline, and preference-aggregation recommenders. (a) Baseline (all scenarios). (b) Baselines (scenarios with nonconflicting ground truth). (c) Preference-aggregation recommenders (all scenarios). (d) Preference-aggregation recommenders (scenarios with nonconflicting ground truth).

To calculate the accuracy of SoSharP's recommendations, we run each testing triplet through SoSharP as if it was an actual multiuser scenario. After each round, if a recommendation is generated, we compare the recommended sharing policy with the actual sharing policy of each response in the triplet. If both are equal, SoSharP recommended a sharing policy correctly; otherwise, it failed.

Recommendations

First, we compare SoSharP with three baselines: *Always Self*, *Always Common*, and *Always All*, each of which always recommends the policy corresponding to its name; and second, we compare it with two recommenders based on preference aggregation (adapted from other work[7,8]): *Veto* recommends the most restrictive policy among users' preferred policies and *Majority* recommends the policy that the majority of users prefer.

Figure 2a shows the accuracies of SoSharP and the baseline recommenders over 510 triplets we form from 1,130 observations in the testing set.

The main goal of our study was to collect data for bootstrapping SoSharP. Thus, we didn't evaluate user interface questions such as how users could accommodate SoSharP's

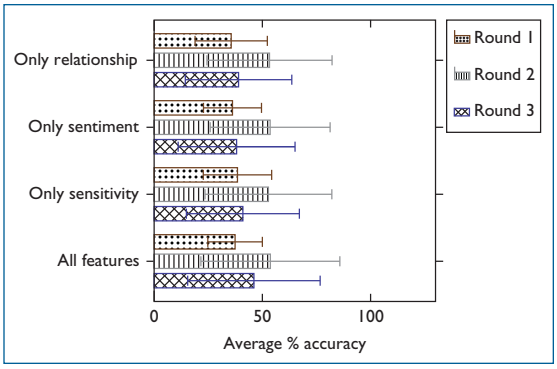

Figure 3. Average accuracy per round.

recommendations or reach an agreement about the optimal sharing policy. Instead, we identified scenarios where the ground truth is in conflict — for example, a scenario where two participants chose *share with all* and the third chose *share with common friends*. From the test data, we generated 71 triplets with a nonconflicting ground truth (that is, all participants chose the same sharing policy). Figure 2b shows the accuracies of SoSharP and baseline recommenders for these 71 triplets. SoSharP yields an accuracy close to 1, indicating that, nearly all of the time if all users agree on a sharing policy, SoSharP recommends that policy.

Because *Veto* and *Majority* depend on users' preferences, they can't provide any recommendation in the first round of SoSharP. To compare SoSharP with these two recommenders, we employ a modified version of SoSharP that never provides a recommendation in Round 1. Figure 2c shows the results of this comparison. As with the baseline recommenders, Figure 2d shows the results considering only scenarios with a nonconflicting ground truth.

In Figure 2, all differences are statistically significant ($p < 0.05$), except for the *Always Self* versus *Always Common* comparison in Figure 2b; both yield low accuracy with high variance.

Incremental Improvement

To evaluate whether adding features as SoSharP proceeds from one round to the next increases its accuracy, we measure the accuracy in each round separately. That is, when SoSharP generates a recommendation, we credit that accuracy score to the round that created the recommendation.

SoSharP provides recommendations even when some contextual features are unspecified. For example, a personal agent might infer the relationship among the parties but fail to infer sentiment and sensitivity. In that case, SoSharP would provide a recommendation employing a relationship as the sole contextual feature.

Figure 3 shows the average accuracy achieved at each round. Further, to simulate the effects of unknown contextual features, in this evaluation, SoSharP also generates recommendations employing only one contextual feature at a time. Round 2 is the most accurate overall. Although the accuracy in Round 1 reduces if only one contextual feature is specified, accuracy in Rounds 2 and 3 is affected only slightly. All the differences are statistically significant ($p < 0.05$) except for the *Only Sentiment* versus *Only Relationship* comparison in Round 3; both have medians that aren't statistically different.

Discussion

SoSharP focuses on multiuser scenarios. However, a variety of techniques seek to reduce the burden of privacy management on SNSs from an individual user's perspective. Özgür Kafalı and colleagues[9] detect privacy violations on SNSs. Nadin Kökciyan and Pınar Yolum[10] expand on that approach and employ an agent-based representation of an SNS, reasoning about commitments between agents. Lujun Fang[11] proposes a tool that recommends sharing policies based on contact similarity. Anna Cinzia Squicciarini and colleagues[5] propose A3P, which suggests a photo-sharing policy based on context, content, and metadata.

A few works focus on multiuser privacy. For instance, Anna Cinzia Squicciarini and colleagues[12] propose an auction-based framework to help users reach an agreement. Other approaches elicit sharing policies based on fixed[8] or variable[7] preference aggregation methods. Hongxin Hu and colleagues[13] describe a game-theoretic mechanism for multiparty access control. These works base their recommendations on the users' sharing preferences and require users to specify all the information. SoSharP, instead, takes into account elements such as the context, users' characteristics, and the relationship among them, and can generate a recommendation even when some information is unknown.

Other works focus on applying individual privacy settings to specific elements in a

photo. Jun Yu and colleagues[14] present iPrivacy, an approach that automatically identifies privacy-sensitive elements in a photo and blurs them to protect the user's privacy. Similarly, Panagiotis Ilia and colleagues[15] suggest blurring users' faces depicted in a picture based on each user's preference. These approaches can, however, limit the value derived from the information.

SoSharP aims to enhance the user's experience by recommending an appropriate sharing policy and respecting all parties' preferences. Because SoSharP can work even when some of the scenario information is unknown, it can proactively recommend sharing policies without user input.

Our evaluation demonstrates the validity of bootstrapping SoSharP and shows that SoSharP performs better than other baseline methods. A limitation of our approach is that participants provided responses to scenarios where they weren't directly involved. Thus, our methodology doesn't support dynamic feedback and whether users would concede ground regarding their preferences. For example, a user preferring *share with all* could consider the *share with common friends* recommendation as acceptable.[1] Data collected from users employing SoSharP for a long period of time would enable us to evaluate the acceptability of SoSharP's suggestions.

SoSharP doesn't address all the information inferences that might occur in a multiuser scenario. Suppose Alice knows that Bob and Charlie were together last Friday, but she doesn't know where. Bob doesn't want Alice to know that he was at a party that day. Charlie wants to share a cool picture (in which Bob doesn't appear) from the party. SoSharP doesn't consider Bob's preference and recommends sharing with all. However, this violates Bob's privacy because if Alice sees Charlie's picture, she can infer that Bob was at the party.

Additional directions for future work include engineering additional features during recommendation — for example, incorporating personality traits as user-based features (Round 1); adaptive agents that learn how flexible their users' preferences are; and agents that negotiate and persuade each other to identify policies acceptable to all users.

Acknowledgments

We thank the US Department of Defense (the Science of Security Lablet), Engineering and Physical Sciences Research Council (EP/M027805/1), Ministerio de Economía y Competitividad (TIN2014-55206-R), and the North Carolina State University College of Engineering for partial support.

References

1. P. Wisniewski, H. Lipford, and D. Wilson, "Fighting for My Space: Coping Mechanisms for SNS Boundary Regulation," *Proc. Computer-Human Interaction*, 2012, pp. 609–618.
2. R. Fogues et al., "Understanding Sharing Policies in Multiparty Scenarios: Incorporating Context, Preferences, and Arguments into Decision Making," *ACM Trans. Computer-Human Interaction*, vol. 24, no. 1, 2017, pp. 1–29.
3. C. Dong, H. Jin, and B. Knijnenburg, "Predicting Privacy Behavior on Online Social Networks," *Proc. 9th Int'l AAAI Conf. Web and Social Media*, 2015, pp. 91–100.
4. F. Stutzman and J. Kramer-Duffield, "Friends Only: Examining a Privacy-Enhancing Behavior in Facebook," *Proc. Computer-Human Interaction*, 2010, pp. 1553–1562.
5. A. Squicciarini et al., "Privacy Policy Inference of User-Uploaded Images on Content Sharing Sites," *IEEE Trans. Knowledge and Data Eng.*, vol. 27, no. 1, 2015, pp. 193–206.
6. U. Gadiraju et al., "Understanding Malicious Behavior in Crowdsourcing Platforms: The Case of Online Surveys," *Proc. Computer-Human Interaction*, 2015, pp. 1631–1640.
7. H. Hu, G. Ahn, and J. Jorgensen, "Multiparty Access Control for Online Social Networks: Model and Mechanisms," *IEEE Trans. Knowledge and Data Eng.*, vol. 25, no. 7, 2013, pp. 1614–1627.
8. K. Thomas, C. Grier, and D. Nicol, "unFriendly: Multi-Party Privacy Risks in Social Networks," *Proc. Privacy Enhancing Technology Symp.* 2010, pp. 236–252.
9. Ö. Kafalı, A. Günay, and P. Yolum, "PROTOSS: A Run Time Tool for Detecting Privacy Violations in Online Social Networks," *Proc. Advances in Social Networks Analysis and Mining*, 2012, pp. 429–433.
10. N. Kökciyan and P. Yolum. "PriGuard: A Semantic Approach to Detect Privacy Violations in Online Social Networks," *IEEE Trans. Knowledge and Data Eng.*, vol. 28, no. 10, 2016, pp. 2724–2737.
11. L. Fang and K. LeFevre, "Privacy Wizards for Social Networking Sites," *Proc. World Wide Web*, 2010, pp. 351–360.
12. A. Squicciarini, M. Shehab, and F. Paci, "Collective Privacy Management in Social Networks," *Proc. World Wide Web*, 2009, pp. 521–530.

13. H. Hu et al., "Game Theoretic Analysis of Multiparty Access Control in Online Social Networks," *Proc. Symp. Access Control Models and Technologies*, 2014, pp. 93–102.
14. J. Yu et al., "iPrivacy: Image Privacy Protection by Identifying Sensitive Objects via Deep Multi-Task Learning," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 5, 2017, pp. 1005–1016.
15. P. Ilia et al., "Face/Off: Preventing Privacy Leakage from Photos in Social Networks," *Proc. Sigsac Conf. Computer and Comm. Security*, 2015, pp. 781–792.

Ricard L. Fogues has a PhD in computer science from Universitat Politecnica de Valencia. His research interests include social media with an emphasis on artificial intelligence, human-computer interaction, and interpersonal privacy. Contact him at rilopez@dsic.upv.es.

Pradeep K. Murukannaiah is an assistant professor in software engineering at the Rochester Institute of Technology. His research interests include engineering socially intelligent and privacy-aware personal agents. Murukannaiah has a PhD in computer science from North Carolina State University. Contact him at pkmvse@rit.edu.

Jose M. Such is a senior lecturer (associate professor) in computer science at King's College London. His research interests are at the intersection between cybersecurity, artificial intelligence, and human-computer interaction, with a strong focus on privacy, intelligent access control, and co-owned data in socio-technical and cyber-physical systems. Such has a PhD in computer science from the Universitat Politecnica de Valencia. Contact him at jose.such@kcl.ac.uk.

Munindar P. Singh is a computer science professor and the co-director of the Science of Security Lablet at North Carolina State University. His research interests include the governance of sociotechnical systems, with an emphasis on security and privacy. Singh is a Fellow of IEEE and the AAAI, a former Editor-in-Chief of *IEEE Internet Computing*, and the current Editor-in-Chief of *ACM Transactions on Internet Technology*. Contact him at mpsingh@ncsu.edu.