SoSharP: Recommending Sharing Policies in Multiuser Privacy Scenarios

Ricard L. Fogues, Pradeep K. Murukannaiah, Jose M. Such, and Munindar P. Singh

Abstract

Users often share information about others and may inadvertently violate their privacy. We propose SoSharP, an agent-based approach to help users maintain their own and others' privacy by guiding selection of sharing policies in multiuser scenarios. SoSharP learns incrementally and asks for user input only when required, reducing user effort.

1 Introduction

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 may find that the photo Alice uploaded is sensitive. However, Bob has no control over uploading that photo, 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 may not be appropriate for the other users, who may cope with the problem via strategies [14] such as self-censorship and tag-approval (offered by Facebook). However, these strategies may 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 may 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.

2 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.

To realize this vision, we make two contributions.

SoSharP: A recommender for sharing policies. SoSharP addresses multiuser scenarios by (1) automatically learning to recommend a sharing policy from features about context, users, groups, and preferences; (2) providing recommendations despite incomplete information; and (3) improving recommendations by receiving user ratings, incrementally.

Bootstrapping SoSharP via crowdsourcing: *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.

3 Features

Given a multiuser scenario, SoSharP recommends a sharing policy, e.g., *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 [3], and user and group based features introduced here. These features are motivated by the privacy literature, information available in existing SNSs, and our intuitions.

3.1 Context

The context of sharing influences users' privacy attitudes [1] and, consequently, sharing decisions in a multiuser scenario. SoSharP incorporates three key contextual factors in the recommendation process.

Relationships among the individuals (e.g., privacy attitudes may differ for family versus friends).

Sensitivity of the information (e.g., users may share less sensitive information more freely).

Sentiment of the information (expressing emotions and eliciting emotional responses from others influences how users build social capital).

3.2 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 [12].

Education level: 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.

3.3 Sharing Preferences

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

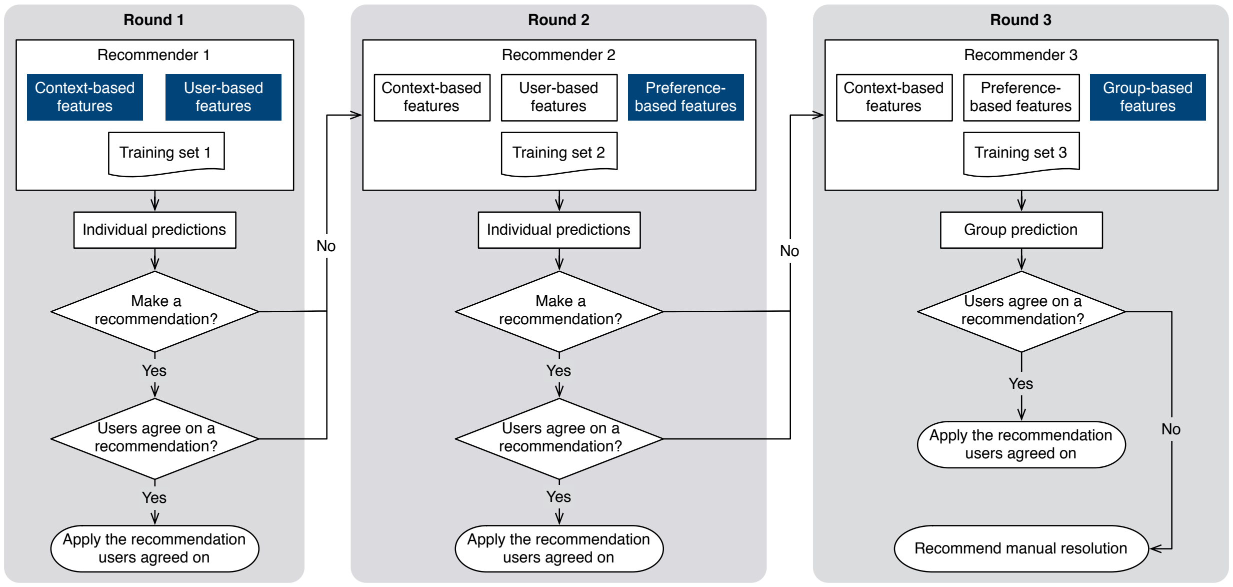


Figure 1: SoSharP's incremental recommendation approach, highlighting the information added in each round.

Most restrictive policy 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 is opposite of *most restrictive policy*.

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

3.4 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.

4 Recommenders

SoSharP provides personalized recommendations without requiring extensive user input, thus reducing the burden on users. As Figure 1 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 may 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 same as SoSharP's Round 1 recommendations or different if a user did not 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 is 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.

5 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 take 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 is not 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 [10] but for a multiuser scenario.

To create scenarios, we reuse the picture survey instrument from prior work [3], 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.

5.1 Picture Survey

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

Context: We consider that (1) the individuals in a picture are related via one of three predefined relationships: *friends*, *family*, or *colleagues*; (2) the pictures is sensitive or nonsensitive; and (3) it conveys a positive or a negative sentiment. Doing so yields 12 contexts. We select representative picture for each context. However, as Table 1 shows, we ask participants to identify the relationship among the people involved in the scenario, and rate sensitivity (Likert scale 1 = not sensitive at all, 5 = very sensitive), and

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?

Table 1: Shortened example of a picture survey [3].

sentiment (1 = extremely positive, 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. Further, 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. Since 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 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 restriction above).

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 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 preferences of all users.

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 post-survey about the participant's general opinions about dealing with multiuser scenarios. The picture survey scenarios were randomized to counter ordering bias. Our study was IRB approved.

Quality Control

We required participants to have completed at least 50 tasks on MTurk and to have had a success rate of at least 90%. We included an attention check question [4] in the ratings section of each picture survey, asking how many people were present in the picture, answering which 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).

5.2 Datasets

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

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.

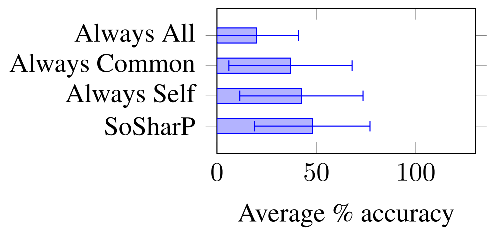
Round 2 training dataset uses participants' responses to Q2. Each observation includes features analogous to those in Round 1 and features computed from the preferences presented in the corresponding Q2 scenario. For example, if the scenario of a given response presented two users preferring *share with all* and one preferring *share with themselves*, the feature *majority policy* would be *share with all*.

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.

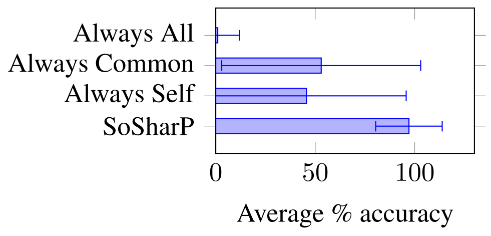
The class (target) variable's value for each observation in Rounds 1 and 2 datasets is the sharing policy chosen by the corresponding participant. For Round 3 dataset, it is the majority policy in the response triplet or *share with common friends* in case of a tie. For example, if a triplet is formed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

5.3 Classifiers

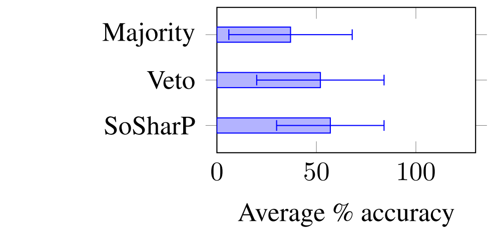
From each dataset above, we train a three-class (each sharing policy is a class) machine learning classifier. In each round, SoSharP employs the classifier for the corresponding round to recommend sharing policy.



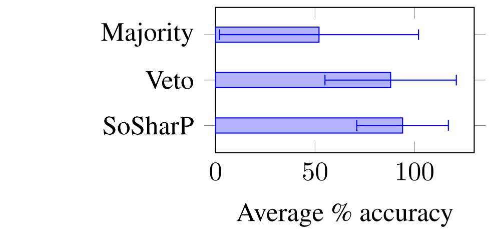
(a) Baseline (all scenarios)



(b) Baselines (scenarios with nonconflicting ground truth)



(c) Preference-aggregation recommenders (all scenarios)



(d) Preference-aggregation recommenders (scenarios with nonconflicting ground truth)

Figure 2: Accuracy comparisons between SoSharP, baseline, and preference-aggregation recommenders

We chose specific classifiers and tune their parameters based on empirical evaluation 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.

6 Evaluation

We gathered 3,767 valid picture survey responses, of which we use 70% (2,637) to build training datasets, and the other 30% (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.

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.

6.1 Recommendations

We compare SoSharP with (1) three baselines: *Always Self*, *Always Common*, and *Always All*, each of which always recommends the policy corresponding to its name; and (2) two recommenders based on preference aggregation (adapted from [5, 13]): *Veto* recommends the most restrictive policy among users' preferred policies and *Majority* recommends the policy the majority of the users prefer.

Figure 2a shows the accuracies of SoSharP and 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. Therefore, we did not evaluate user interface questions such as how users could accommodate SoSharP's recommendations or reach an agreement about the optimal sharing policy. Instead, we identified scenarios where the ground truth is in conflict, e.g., 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 nonconflicting ground truth (i.e., 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 cannot 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 nonconflicting ground truth.

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

6.2 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 may infer the relationship among the parties but fail to infer sentiment and sensitivity. In that case, SoSharP would provide a recommendation employing 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 *Only Sentiment* vs. *Only Relationship* comparison in Round 3 which have medians that aren't statistically different.

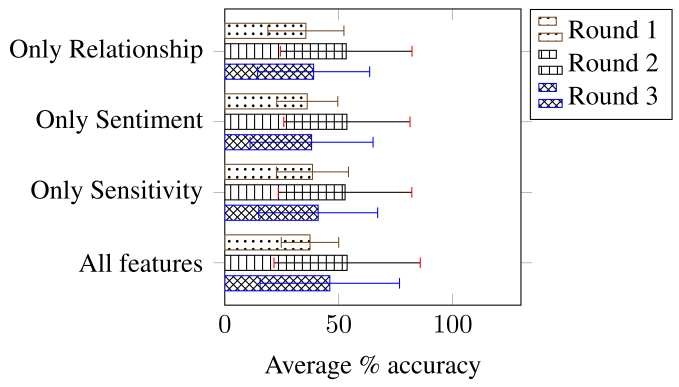


Figure 3: Accuracy per round.

7 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. Kafalı et al. [8] detect privacy violations on SNSs. Kökciyan and Yolum [9] expand on that approach and employ an agent-based representation of an SNS, reasoning about commitments between agents. Fang [2] propose a tool that recommends sharing policies based on contact similarity. Squicciarini et al. [10] propose A3P, which suggests a photo sharing policy based on context, content, and metadata.

A few works focus on multiuser privacy. Squicciarini et al. [11] propose an auction-based framework to help users reach an agreement. Other approaches elicit sharing policies based on fixed [13] or variable [5] preference aggregation methods. Hu et al. [6] describe a game-theoretic mechanism for multiparty access control. These works base their recommendations on sharing preferences of the users and require users specifying all the information. SoSharP, instead, takes into account elements such as the context, characteristics of users, and the relationship among them, and can generate a recommendation even when some information is unknown.

Other works focus on applying individually privacy settings to specific elements in a photo. Yu et al. [15] present *iPrivacy*, an approach that automatically identifies privacy-sensitive elements in a photo and blurs them to protect user's privacy. Similarly, Ilia et al. [7] suggest blurring the faces of the users depicted in a picture based on each user's preference. These approaches can, however, limit the value derived from the information.

SoSharP aims at enhancing user experience by recommending an appropriate sharing policy, respecting all parties' preferences. Since SoSharP can work even when not all the scenario information is known, 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 were not 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[14]. Data collected from users employing SoSharP for a long period of time would enable us to evaluate how acceptable SoSharP's suggestions are.

SoSharP doesn't address information inferences a multiuser scenario may lead to. 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 (1) additional features during recommendation, e.g., personality traits can be incorporated as user-based features (Round 1); (2) adaptive agents that learn how flexible their users' preferences are; and (3) 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), EPSRC (EP/M027805/1), Ministerio de Economía y Competitividad (TIN2014-55206-R), and the NCSU College of Engineering for partial support.

Author Bios

Ricard L. Fogues ⟨rilopez@dsic.upv.es⟩ is a PhD student in Computer Science at Universitat Politecnica de Valencia. His research interests include social media with an emphasis on artificial intelligence, human-computer interaction, and interpersonal privacy.

Pradeep K. Murukannaiah ⟨pkmvse@rit.edu⟩ is an Assistant Professor in Software Engineering at Rochester Institute of Technology. His research interests include engineering socially intelligent and privacy-aware personal agents.

Jose M. Such ⟨jose.such@kcl.ac.uk⟩ is Senior Lecturer (Associate Professor) in Computer Science at King's College London. His research interests are at the intersection between cyber security, 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.

Munindar P. Singh ⟨singh@ncsu.edu⟩ is a Professor in Computer Science and a co-director of the Science of Security Lablet at NC State University. His research interests include the governance of sociotechnical systems with an emphasis of security and privacy. Singh is an IEEE Fellow, a AAAI Fellow, a former Editor-in-Chief of *IEEE Internet Computing*, and the current Editor-in-Chief of *ACM Transactions on Internet Technology*.

References

[1] C. Dong, H. Jin, and B. Knijnenburg. Predicting privacy behavior on online social networks. In *Proc. ICWSM*, 2015.

[2] L. Fang and K. LeFevre. Privacy wizards for social networking sites. In *Proc. WWW*, pages 351–360, 2010.

[3] R. Fogues, P. Murukannaiah, J. Such, and M. Singh. Understanding sharing policies in multiparty scenarios: Incorporating context, preferences, and arguments into decision making. *ACM TOCHI*, 24(1):1–29, 2017.

[4] U. Gadiraju, R. Kawase, S. Dietze, and G. Demartini. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proc. CHI*, pages 1631–1640, 2015.

[5] H. Hu, G. Ahn, and J. Jorgensen. Multiparty access control for online social networks: Model and mechanisms. *IEEE TKDE*, 25(7):1614–1627, 2013.

[6] H. Hu, GJ. Ahn, Z. Zhao, and D. Yang. Game theoretic analysis of multiparty access control in online social networks. In *Proc. SACMAT*, pages 93–102, 2014.

[7] P. Ilia, I. Polakis, E. Athanasopoulos, F. Maggi, and S. Ioannidis. Face/Off: Preventing privacy leakage from photos in social networks. In *Proc. SIGSAC*, pages 781–792, 2015.

[8] Ö. Kafalı, A. Günay, and P. Yolum. PROTOSS: A run time tool for detecting privacy violations in online social networks. In *Proc. ASONAM*, pages 429–433, 2012.

[9] N. Kökciyan and P. Yolum. PriGuard: A semantic approach to detect privacy violations in online social networks. *IEEE TKDE*, 28(10):2724–2737, 2016.

[10] A. Squicciarini, D. Lin, S. Sundareswaran, and J. Wede. Privacy policy inference of user-uploaded images on content sharing sites. *IEEE TKDE*, 27(1):193–206, 2015.

[11] A. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *Proc. WWW*, pages 521–530, 2009.

[12] F. Stutzman and J. Kramer-Duffield. Friends only: Examining a privacy-enhancing behavior in Facebook. In *Proc. CHI*, pages 1553–1562, 2010.

[13] K. Thomas, C. Grier, and D. Nicol. unFriendly: Multi-party privacy risks in social networks. In *Proc. PETS*, pages 236–252, 2010.

[14] P. Wisniewski, H. Lipford, and D. Wilson. Fighting for my space: Coping mechanisms for SNS boundary regulation. In *Proc. CHI*, pages 609–618, 2012.

[15] J. Yu, B. Zhang, Z. Kuang, D. Lin, and J. Fan. iprivacy: Image privacy protection by identifying sensitive objects via deep multi-task learning. *IEEE TIFS*, 12(5):1005–1016, 2017.