A Framework for Context-Aware Privacy of Sensor Data on Mobile Systems

Supriyo Chakraborty, Kasturi Rangan Raghavan
Matthew P. Johnson, Mani B. Srivastava
University of California, Los Angeles
{supriyo, kasturir, mpjohnson, mbs}@ucla.edu

ABSTRACT

We study the competing goals of utility and privacy as they arise when a user shares personal sensor data with apps on a smartphone. On the one hand, there can be value to the user for sharing data in the form of various personalized services and recommendations; on the other hand, there is the risk of revealing behaviors to the app producers that the user would like to keep private. The current approaches to privacy, usually defined in multi-user settings, rely on anonymization to prevent such sensitive behaviors from being traced back to the user—a strategy which does not apply if user identity is already known, as is the case here.

Instead of protecting identity, we focus on the more general problem of choosing what data to share, in such a way that certain kinds of inferences—i.e., those indicating the user's sensitive behavior—cannot be drawn. The use of inference functions allows us to establish a terminology to unify prior notions of privacy as special cases of this more general problem. We identify several information disclosure regimes, each corresponding to a specific privacy-utility tradeoff, as well as privacy mechanisms designed to realize these tradeoff points. Finally, we propose *ipShield* as a privacy-aware framework which uses current user context together with a model of user behavior to quantify an adversary's knowledge regarding a sensitive inference, and obfuscate data accordingly before sharing. We conclude by describing initial work towards realizing this framework.

Keywords

Behavioral Privacy, Context-awareness, Inferences, Model-based Privacy, Android, ipShield

1. INTRODUCTION

Smartphones with onboard and externally connected body-worn sensors are capable of tracking our locations and social neighborhoods, monitoring physiological markers, and learning about our evolving social dynamics. The raw data collected are increasingly being used to *infer* our personal, social, work and urban contexts. These contexts are in turn acquired by a growing ecosystem of context-aware apps

ACM HotMobile'13, February 26–27, 2013, Jekyll Island, Georgia, USA.

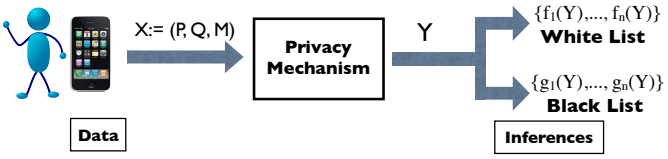


Figure 1: A simplified information flow scenario from users to app producers. The shared data is used for computing various inferences.

to provide us with *personalized* app experiences such as behavior-tailored insurance plans, mobile health (mHealth) diagnostics and customized recommendations to enrich our social and personal interactions (or targeted advertising). We refer to the benefit to the user of such personalization as *utility*. Embedded within the identity-annotated time-series of shared sensor data is the user's full behavioral footprint, to the minutest detail, including many that she may wish to keep private. Users are often unaware of the possible (mis)use of their personal information by the data-consuming untrusted apps, causing information asymmetry between information providers (users) and consumers (apps.), leading to a *lemon market* [25]: users are increasingly skeptical of app producers' privacy policies, with no way of verifying good policies, and so providers have little incentive to abide by them, leading to more user skepticism, in a negative feedback spiral.

Information flow from users to apps is summarized in Fig. 1. Broadly speaking, the shared sensor data X has: (a) a set of personal identifiers P, such as name and SSN associating it to the user; (b) a set of quasi-identifiers Q, such as age, gender, zip code, which when combined with auxiliary information sources can possibly identify the user; and (c) a set M, containing data values corresponding to the measurement. Shared data is represented by Y and is used by the apps to compute various inferences, some of which can be sensitive (i.e., are such that the user wishes them to remain private). Consider the following examples:

E1: A user shares her accelerometer data X with an mHealth app to monitor her overall *activity level* $f(Y)$ but faces the risk of revealing her exact *activity type* $g(Y)$, which is also inferable from the same data.

E2: A user desiring a *safe driver discount* $f(Y)$ on insurance rates may be willing to share her location and accelerometer data X. However, periodic location release near a place of worship may reveal *religious preferences* $g(Y)$.

E3: A user is required to share EKG and respiration data

X with an insurance company which uses the data to check for *heart and respiratory disorders* $f(Y)$, and provide discounted rates. However, the same data can be used to detect *onset of stress* $g(Y)$, a behavior the user wants to keep private.

If Y is the same as X (no privacy mechanism is used), the presence of set P implies that the inferred sensitive behaviors may now be traceable back to the user, violating her privacy.

Prior work on privacy mechanisms is centered around two design objectives: data anonymization and incomplete reconstruction. The process of anonymization includes the removal of P and the suitable *obfuscation* of Q present in X to break the association between the data and the user. In a multi-user setting where privacy of an entire database of user data is desired, measures such as *k-anonymity* [24] and *l-diversity* [17] are used to determine the level of obfuscation required to make the user anonymous or indistinguishable within a subpopulation, achieving privacy-in-numbers. However, the breakdown of anonymization in the face of auxiliary information [18, 11, 23] has prompted the design of measures such as *differential privacy* [6] which, in a multi-user setting, recommend use of structured noise to perturb aggregate query responses and protect the membership (i.e., presence or absence) of an individual within a database. The second objective is to prevent complete reconstruction of X from Y. To achieve this in addition to anonymization, the measurements in M are also adequately perturbed [22]. By preventing reconstruction, the goal is to protect against private inferences which could be made from X alone. However, it has been shown that partially reconstructed data can be used to make inferences about private behaviors [23, 11].

Now consider instead a setting in which a *single* user shares a time-series of sensor data annotated with identity information, as illustrated in $E1 - E3$. This motivates the investigation: *what are the privacy and utility goals appropriate to such a setting?* The traditional notion of protecting user identity is no longer a concern because the apps under consideration (e.g., mHealth, customized insurance plans) require user identity for providing personalized services (utility). Thus, instead of identity, a user is interested in protecting the privacy of sensitive behaviors which can be inferred from the shared data. Another consequence of this single-user setting is that privacy measures relying upon privacy-in-numbers within a subpopulation do not apply.

In this paper, we consider the general form of the privacy problem given above and make three main contributions. First, we give a very general privacy model in which two sets of inferences (the white and black lists shown in Fig. 1) constitute utility and private behaviors, respectively. These inferences are marked as $f(Y)$ and $g(Y)$ in examples $E1 - E3$. The use of inference functions allows us to establish a terminology to unify prior notions of anonymization- and reconstruction-based privacy as special cases of the more general problem. Second, we identify several information disclosure regimes, each corresponding to a specific privacy-utility tradeoff, and privacy mechanisms designed to realize these tradeoff points. These insights lead us to our third contribution: the conceptualization of *ipShield*—a privacy-aware framework which uses current user context together with a model of user behavior to quantify an adversary's knowledge regarding a given sensitive inference, and then apply an appropriate privacy mechanism on the data before sharing. We conclude by describing the initial work we have done towards realization of the framework.

2. PRIVACY PROBLEM AND CHOICES

We define an *inference function* as a classifier which takes shared data as input and performs a classification of the user as being in a particular behavior state, e.g., into one of the activity states (walking, running, still) in $E1$, one of several religions in $E2$, or the onset of stress in $E3$. Classifiers are machine learning algorithms (e.g., supervised, unsupervised, reinforcement) used to learn and then classify based on patterns in the data. For example, in supervised learning, these patterns are learned using labeled data provided by the user.

The problem of protecting the privacy of sensitive inferences is characterized by a tradeoff between the application's need to obtain information for providing *utility* to the user and the user's need to control the information shared for protecting *privacy*. As shown in Fig. 1, our privacy notion is defined in terms of what can be extracted from the shared data Y. The user specifies his privacy preferences as a *blacklist* of inferences, $\{g_1(Y), \ldots, g_n(Y)\}$, and the app provides its utility requirements as a *whitelist* of inferences, $\{f_1(Y), \ldots, f_n(Y)\}$. The privacy mechanisms are designed to ensure the app can effectively compute whitelisted inferences to some degree of accuracy, but where the app cannot draw the blacklisted inferences. Ideally, any data shared with an app should not reveal any more information than what is already known to the app about the blacklisted inferences from prior (population-scale) knowledge or side-channels. We remark that this is a general formulation of the privacy problem, and that the previously mentioned privacy mechanisms such as anonymization and protection against reconstruction attacks can be thought of as carefully chosen blacklist inferences.

2.1 Information Disclosure Regimes

The various possible tradeoff points for the utility and privacy objectives define a spectrum of information disclosure regimes that a user can operate in. At one extreme, corresponding to zero disclosure, the user shares no information at all, ensuring complete privacy but at the cost of complete loss in utility. At the other extreme, corresponding to full disclosure, all information is shared. Now the user achieves utility at the cost of complete loss of privacy. Each point in this spectrum is realizable by using an appropriately designed privacy mechanism, now we discuss some two operation points of particular interest.

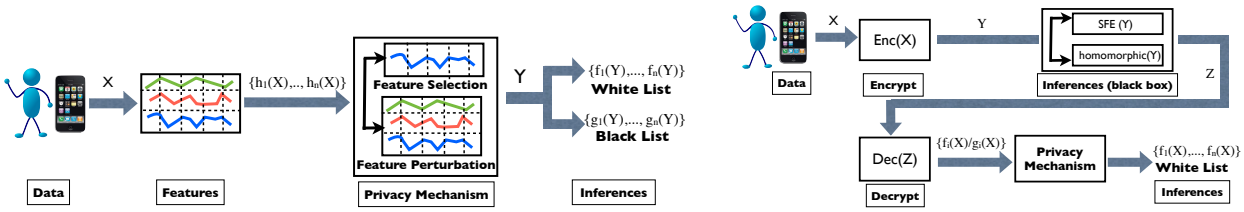
1. Maximum Privacy Disclosure (MaxP): We release information (some transformation of X) such that only the desired utility (whitelisted functions, and consequences inferable from them) can be computed from the released information. This point corresponds to targeted disclosure.

2. Maximum Utility Disclosure (MaxU): We release information which preserve all characteristics of X, except those which can be used to violate privacy (blacklisted functions). This point corresponds to targeted hiding.

2.2 Realization of the Privacy Mechanisms

We define a privacy mechanism as a two-step process: first identifying the data to be shared (e.g., the subset of features, data samples, inferences, data types, etc.) and then applying obfuscation to the data before sharing. Below, we enumerate a set of potential privacy mechanisms.

1. Feature Selection: Instead of the high-dimensional data X, from which information flow is hard to control [19, 18] we extract a set of features $F = \{h_1(X), \ldots, h_n(X)\}$ and use them to represent the data in a lower-dimensional space.



(a) Feature selection and perturbation steps for MaxP and MaxU, respectively.

(b) SFE and Homomorphic encryption for black-box computation of whitelisted functions.

Figure 2: Different realizations of privacy mechanisms.

The functions $h_i(X)$ can represent features like mean, variance, Fourier coefficients, etc., extracted from the data samples over time. Inferences typically operate in the feature space and use a subset of F to perform their classification.

To implement MaxP, we observe that by sharing features we can better control the information shared. The privacy mechanism (see Fig. 2(a)) selects a subset of features required by the whitelisted inferences but which do not contribute to the blacklisted ones. The obfuscation step either suppresses all the other features and shares only the selected features or synthesizes data X' preserving only selected features (and their consequences) and nothing else. This mechanism requires the app to share information regarding the features the inferences depend on.

2. Sharing whitelisted inferences: Another privacy mechanism which also implements MaxP is that of sharing whitelisted inferences (or suppressing blacklisted ones). The idea is to compute the inferences on the phone, obfuscate the results such that they do not reveal any information about the blacklisted inferences and share the obfuscated results instead of X. For this to work, the apps need to provide the exact implementation of the inference algorithm to the user, which may be proprietary and difficult to share. An alternate strategy for evaluating the whitelisted inference functions is to use cryptographic techniques. We suggest two such techniques (see Fig. 2(b)).

- One-sided Secure Function Evaluation (SFE) can applied (using, e.g., Yao's garbled circuit [27]) to evaluating the inference function. Both parties provide their inputs (the user provides her sensor data, and the app the inference function), and the function is evaluated. Since the protocol is one-sided, only the user obtains the result of the computation; and the app knows nothing about the user input. The user can then obfuscate the result before sharing it with the app.

- Homomorphic Encryption [8] allows computation to be carried out on the cipher text directly, yielding an encrypted result of the operations performed on the plain text. The user performs homomorphic encryption on the data and sends it to the app, which can then perform function evaluation on the encrypted data and return the encrypted result to the user, who decrypts it to obtain the result. The second step is to perform obfuscation of the result before sharing with the app.

While the above techniques allow computation of the inference functions without their disclosure, there is no way for the user to know if the results computed are for the whitelisted inferences only. Thus the privacy mechanism must use other techniques (such as zero knowledge proofs [10], random spot checks, etc.), to ensure that the correct functions are being evaluated. In addition, while feasible in theory, these techniques are extremely computationally expensive and thus energy-intensive.

3. Random Projection: Following this mechanism (see Fig. 3), we share projections of the features instead of the features themselves [16]. That is, we project the features into a lower dimensional space before sharing. To ensure that privacy is maintained, the transformation is kept private and is known only to the user.

For utility goals, the user furnishes training labels so that the app can learn a classifier, based on the projected features and associated labels, for the whitelisted inferences (and their consequences) but nothing else. In order to learn the labels in the embedded space, the key property required is that pairwise distances between points in the original feature space be preserved. Fortunately, when the transformation is derived from randomly generated basis vectors drawn from an i.i.d. normal distribution, the Johnson Lindenstrauss lemma states that this property holds with high probability when the dimensionality of the new projected feature space satisfies a certain size constraint [13, 16].

This mechanism eliminates the need to know a priori the mapping between the inferences and features as required by the feature selection approach. It places a significant burden on the app, however, which must now learn the classifier or the whitelisted inference. An advantage of using this mechanism is that we can guarantee privacy when there is no side-channel information, as only the whitelisted inference labels are shared.

4. Feature Perturbation: We use this mechanism to realize MaxU (see Fig. 2(a)). We select and transform a specific set of features, and share everything else. For example, for an audio signal we can choose pitch as the feature to transform and use perturbation to obfuscate it. While inferences such as identification of the speaker, which rely on pitch, are affected, other inferences not depending on pitch remain accurately computable. One of the drawbacks of this mechanism is that it does not protect against blacklisted inference functions, which can learn a classifier using the set of released features instead of the transformed ones.

3. DATA FLOW SCENARIOS

A privacy framework implementing (a possible subset of) the above privacy mechanisms must fit into existing state-of-the-art mobile platforms. We investigate the different placement points of such a framework by taking into account the various data flow scenarios between a user and the apps.

We use the Android OS [1] (see Fig. 4(a)) as representative of a state-of-the-art mobile system. To simplify our

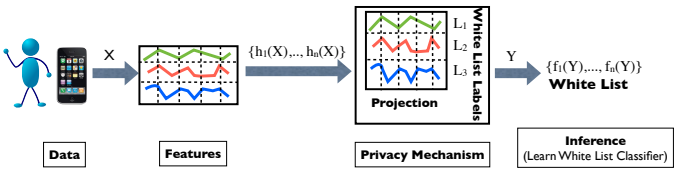


Figure 3: Random Projection for sharing whitelisted functions. We share both projection and the corresponding whitelist labels.

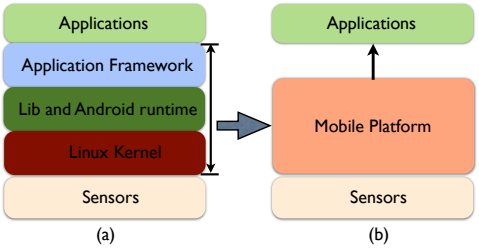


Figure 4: Abstract model of a phone.

presentation, we derive an abstract model that combines the functionalities of the middle layers into a single block and call it the mobile platform. Apps running on the phone form part of the topmost layer (see Fig. 4(b)). We assume that the platform is *trusted* to not leak user information whereas the app layer (running third-party apps) is *untrusted*. Figs. 5 (a), (b) and (c) illustrate the possible data flows. Computation local to the phone is performed within the dotted box. The implementation of the privacy mechanisms will vary depending on the placement of the privacy framework.

If a locally running app (shown in Fig. 5(a)) is completely insulated and does not communicate outside the dotted box then the user need not obfuscate data before sharing. However, static analysis of app code and information flow tracking techniques [7] have revealed the existence of side channels through which apps leak information to the cloud [9]. To prevent such attacks the framework needs to be placed between the mobile platform and the app and would require changes to the platform code. For a cloud-based app (Fig. 5(b)), the framework can be included as part of the client implementation. While this would eliminate the need to make platform changes as described for the previous case, it would involve modification of the app clients. In addition, the modified client code would need to be trusted and not leak information. Finally, with a broker-cloud-hosted app (Fig. 5(c)), the privacy framework can be pushed to the trusted server implementing the broker service.

Each placement choice corresponds to a different tradeoff in terms of implementation complexity. Choices in Figs. 5(b) and 5(c) are specific to an app client or a broker, and hence might require significant duplication of implementation effort. Also, for Figs. 5(a) and 5(b), the implementation has to be lightweight owing to resource constrained mobile platforms, whereas there is no such restriction when the framework is running on a trusted server as in Fig. 5(c). Without assuming a trusted broker, we implement our framework as part of the mobile platform. We can thus intercept and obfuscate data in all the possible scenarios.

4. ipShield: THE PRIVACY FRAMEWORK

We conceptualize `ipShield`, the inference privacy framework (see Fig. 6) and describe our implementation on an

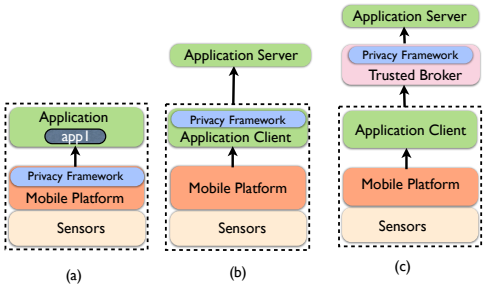


Figure 5: Data-flow paths from user to apps. Data is shared with (a) locally running apps, (b) directly with cloud-hosted apps. and (c) via trusted broker with cloud-hosted apps.

Android-based mobile platform. Prior work on privacy frameworks rely on static privacy policies, or use information flow techniques to detect potential leakage from apps and apply binary policies of complete access or no access to data at all [1, 2, 7]. In comparison, `ipShield` makes two main contributions. First, it implements context-aware privacy policies (blacklist and whitelist specification). Broadly, context refers to a combination of the current physical (e.g. walking, running, still, smoking), location (e.g. indoor, outdoor, office, home), social (with friends, in meeting), and even psychological (e.g. stress) state of a user and can be inferred from a variety of sensor measurements. There is active research towards creating an operating system service [5, 26], which would provide apps with *contexts* rather than sensor data. Context-awareness allow users to define dynamic fine-grained privacy policies depending on current context – an improvement over the current binary and static policies. Since the user wants to protect against blacklisted inferences, possibly when in certain context states, other contexts are ipso facto safe for data release. Second, `ipShield`, uses a graphical model to capture initial adversarial knowledge and its subsequent increase with each disclosure. It then uses the model to determine the level of obfuscation required before releasing the data. We present a case for model-based privacy and follow it up with the design and initial implementation of the privacy framework.

4.1 A Case for Model-Based Privacy

The degree of obfuscation required depends on the adversary's capabilities. Prior work has shown that human behavior, and thus, we can assume, behavioral inferences, exhibit significant correlation [14], which can be captured using graphical models [20, 12]. We assume a model-based adversary that maintains a belief on the blacklisted inferences based on prior knowledge and continuously updates the model parameters by observing the shared data.

4.1.1 Adversary Model

The maximum amount of information that can be extracted by an adversary is quantified by the mutual information between the whitelisted and the blacklisted inferences. To compute the mutual information, we need to learn the joint distribution of the two, which can be done using sophisticated graphical models. The mutual information gives an upper bound on the amount of information which can be possibly extracted by the adversary.

We assume that the adversarial power is captured using graphical models such as a Dynamic Bayesian Network (DBN) or a Markov Chain. The states of the model corre-

spond to the different inferences that could be made using the shared data. Prior knowledge of the adversary from auxiliary sources is expressed as the prior probability on the occurrence of a state and the transition probabilities on the edges. We track the change in adversarial knowledge by updating the probabilities with every data release.

4.1.2 Learning the Obfuscation Function

The obfuscation functions are used to transform the data so that (a) the adversary cannot make the blacklisted inferences; (b) the data utility is preserved; and (c) the obfuscated data is *plausible*, i.e. not so obfuscated that the data points become outliers which can be easily filtered by the adversary. However, the obfuscation function only provides privacy against the graphical model used for the adversary. This requires that the model be powerful and capture data dependencies effectively.

Depending on the adversary's belief we define three different types of obfuscation actions: (a) Suppression, where data is not released. However, we need to ensure that suppression itself does not increase adversarial knowledge about sensitive state; (b) Perturbation, where structured noise is added to increase uncertainty in the blacklisted inferences; and finally (c) Synthesis, where synthetic data unrelated to the actual data is generated by sampling the graphical model, to ensure plausibility of the obfuscated data.

4.2 Design of `ipShield`

The different layers of `ipShield` are shown in Fig. 6 and are explained below.

Sensors: This layer provides access to built-in sensors on the phone (e.g., accelerometers, GPS, microphones, camera), or external body-worn sensors such as a galvanic skin response sensor, EKG sensors, or other virtual sensors such as the calendar (providing event schedules), battery (providing power status), feature sensor extracting various features such as mean, variance, etc. from actual sensor measurements (e.g. probes in Funf [3]).

Context Framework: This layer collects data from the sensors and uses it to identify the current *context state* of the user. We have implemented this layer as an extension to the open-source sensor data collection and dissemination library Funf [3]. For example, an activity context can be a decision tree classifier that was trained offline. At runtime, accelerometer data is given to the classifier, which extracts features such as mean, variance, and Fourier coefficients, and uses them to classify the data into activity states such as *walking*, *running*, and *still*.

Inference Framework: The inference framework takes the current state, which recall is a set of contexts, and uses them to compute inferences. This is again done using classifiers, which are trained offline using contexts and user-provided labels as training data. For example, presence at the location of a religious place, together with time of day and day of week, could be used to make an inference about religious preferences. These inferences are part of the black list and white list specified by the user and the apps, respectively.

Privacy Firewall: The privacy firewall comprises of three different subsystems. First, the graphical model, which is as mentioned in Section 4.1. Second, the user preferences subsystem, which allows users and apps to specify the black list and white list of inferences, respectively. The third subsystem contains the rules and the obfuscation blocks. The rule block contains the set of privacy policies (similar to rules using iptables for configuring network firewalls), which specify the obfuscation action on the sensor data when a specific context state is true. These policies can be either configured by the user or derived from the white and black lists using the graphical model. There are two ways a user can configure these lists. First, we envision that similar to network firewall config files, or spam filtering, a user can obtain such config lists published by privacy experts and personalize them according to her preferences. Second option that is currently being researched is the use of crowdsourcing to understand user expectation of privacy and utility of popular apps used on the phone [15].

The obfuscation block implements the different actions. If the obfuscated data does not increase the adversarial knowledge about the sensitive inference, data is released, else, it is subjected to further iterations of obfuscation before it meets the privacy and utility requirements.

Application: The privacy firewall is the point at which data is shared with the apps. While most of the current apps require sensor data directly, there has been a steady growth in context-aware apps which take contexts as input instead of sensor data. Thus, the firewall should implement interfaces for sharing both obfuscated sensor data as well as derived contexts.

4.3 Prototype Implementation and Use Case

In [21] we provided an implementation of the changes on the Android platform required to enforce the context-aware obfuscated sensor data sharing. For model-based privacy, we created a prototype of a DBN on the Android platform [4] and used it to determine when to suppress or release data. We are currently working towards a privacy rule specification framework for user preferences.

As use case, we consider an example of selective suppression of features extracted from accelerometer data. *Activity level* of a user, is a binary decision of being active or inactive, and can be inferred via decision trees directly from features over windows of accelerometer data. The adversary model is also a decision tree that infers the *activity type*, such as walking, running, biking, from the same windows of accelerometer data (E1 in Section 1). Thus, the white list here is to perform activity level detection and the black list is to prevent the detection of specific activity type. The obfuscation we perform is selective suppression of features (we release entropy of Fourier coefficients) such that the *activity type* inference is no longer inferable via a decision tree based adversary.

5. LIMITATIONS AND FUTURE WORK

The general privacy problem of precluding a set of behavioral inferences from being made while ensuring that others *can* be made involves a complex interaction of information theory and machine learning. The problem is well defined only when the whitelisted inferences do not completely overlap the blacklisted inferences, in which case releasing even the whitelist inferences themselves would violate privacy. For well defined settings, the challenge lies in finding the right feature subset or data transformation which will have an acceptably low mutual information with the blacklisted inferences.

Our framework offers a context-aware model-based solution to the problem. The ability of the model to account for the relationship between a variety of inferences, learning

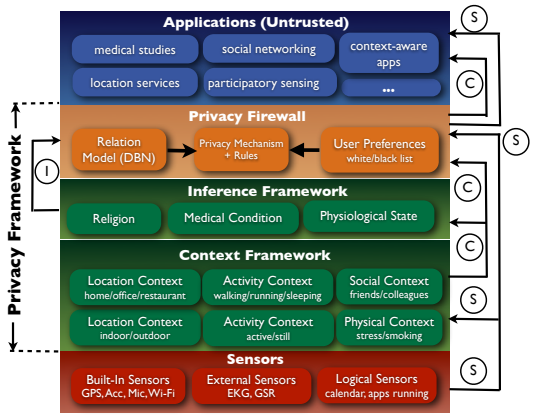


Figure 6: Inference Privacy Framework. S = sensor data, C = context state, I = inference.

accuracy based on training data, and finally the feature set corresponding to the whitelist inferences are all key to the success of the framework. In addition, apps which use data that manifest dependencies over long periods of time, maybe hard to protect against, owing to order constraints on the graphical model.

Finally, our adversary model is limited because it does not account for the relationship between the app producers. A possible enrichment to the model could be to augment it with information from social networks which model personal relationships. In the future, we aim to incorporate these into our system for a better privacy experience.

Acknowledgement

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. This material is also based upon work supported by the NSF under awards CNS-0910706 and CNS-1213140. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or represent the official policies of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defense or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. The first and second authors are supported by the Qualcomm Innovation Fellowship award for 2011-2012. We would also like to thank our shepherd James Scott for his valuable suggestions and guidance.

6. REFERENCES

[1] `http://http://developer.android.com/guide/basics/what-is-android.html`.
[2] `github.com/gsbabil/PDroid-AOSP-JellyBean`.
[3] `http://funf.org`.
[4] S. Chakraborty, K. R. Raghavan, and M. Srivastava. Poster: Model-based context privacy for personal data streams. CCS, 2012.
[5] D. Chu, A. Kansal, J. Liu, and F. Zhao. Mobile apps: it's time to move up to condos. HotOS, 2011.
[6] C. Dwork. Differential privacy: a survey of results. TAMC, 2008.
[7] W. Enck and et. al. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. OSDI, 2010.
[8] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. EUROCRYPT, 2011.
[9] C. Gibler, J. Crussell, J. Erickson, and H. Chen. Androidleaks: automatically detecting potential privacy leaks in android applications on a large scale. TRUST, 2012.
[10] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. STOC, 1985.
[11] P. Golle and K. Partridge. On the anonymity of home/work location pairs. Pervasive, 2009.
[12] M. Götz, S. Nath, and J. Gehrke. Maskit: privately releasing user context streams for personalized mobile applications. SIGMOD, 2012.
[13] K. Kenthapadi, A. Korolova, I. Mironov, and N. Mishra. Privacy via the johnson-lindenstrauss transform. *CoRR*, abs/1204.2606, 2012.
[14] E. Kim, S. Helal, and D. Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 2010.
[15] J. Lin, N. Sadeh, S. Amini, J. Lindqvist, J. I. Hong, and J. Zhang. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. UbiComp, 2012.
[16] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. on Knowl. & Data Eng.*, 2006.
[17] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 2007.
[18] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, 2008.
[19] A. Narayanan and V. Shmatikov. Myths and fallacies of "personally identifiable information". *Commun. ACM*, 2010.
[20] H.-S. Park and S.-B. Cho. Predicting user activities in the sequence of mobile context for ambient intelligence environment using dynamic bayesian network. In *ICAART*, 2010.
[21] K. R. Raghavan, S. Chakraborty, and M. Srivastava. Override: A mobile privacy framework for context-driven perturbation and synthesis of sensor data streams. PhoneSense, 2012.
[22] L. Sankar, S. Rajagopalan, and V. Poor. A theory of utility and privacy of data sources. ISIT, 2010.
[23] M. Srivatsa and M. Hicks. Deanonymizing mobility traces: Using social network as a side-channel. CCS, 2012.
[24] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 2002.
[25] T. Vila, R. Greenstadt, and D. Molnar. Why we can't be bothered to read privacy policies models of privacy economics as a lemons market. ICEC, 2003.
[26] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. MobiSys, 2012.
[27] A. C.-C. Yao. How to generate and exchange secrets. SFCS, 1986.