Privacy-Enhanced Android for Smart Cities Applications

Matthew Lepinski[1], David Levin[1], Daniel McCarthy[1], Ronald Watro[1], Michael Lack[2], Daniel Hallenbeck[2], and David Slater[2](✉)

[1] Raytheon BBN Technologies, 10 Moulton St., Cambridge, MA 02138, USA
{mlepinski, dlevin, dmccarthy, rwatro}@bbn.com
[2] Invincea Labs, 4350 North Fairfax Drive, Arlington, VA 22203, USA
{mike.lack, dan.hallenbeck, david.slater}@invincea.com

Abstract. Many Smart Cities applications will collect data from and otherwise interact with the mobile devices of individual users. In the past, it has been difficult to assure users that smart applications will protect their private data and use the data only for the application's intended purpose. The current paper describes a plan for developing Privacy-Enhanced Android, an extension of the current Android OS with new privacy features based on homomorphic and functional encryption and Secure Multiparty Computation. Our goal is to make these advances in privacy-preserving technologies available to the mobile developer community, so that they can be broadly applied and enable the impactful social utility envisioned by Smart Cities.

Keywords: Privacy · Cyber security · Encryption · Android · Smart cities

1 Introduction

A key component in the creation of Smart Cities is the deployment of appropriate applications ("apps") on mobile devices. These applications will provide data to Smart Cities systems and also allow users to reap the benefits that Smart Cities have to offer. For example, currently popular mobile phone software for traffic reporting (Waze) and navigation (Google Maps) demonstrate the utility of collecting localized sensor information from the "crowd" and suggest that the impact of future Smart City applications will be proportional to the user base.

Privacy is a fundamental issue to user acceptance of apps that deal with personal information. The details of whether privacy protections are required or optional (or perhaps even prohibited) will vary from country to country. It is our belief that widespread adoption of sensitive applications (such as Government-issued or health-related apps) will be the most successful when they are deployed on a ubiquitous mobile platform and in a manner that enables foundational privacy guarantees.

Raytheon BBN Technologies and Invincea Labs are working together on a new project to define and deploy new privacy-preserving tools on the Android OS, based on recent results in cryptography and human-data interfacing. The project goal is to develop a privacy-enhanced version of the Android OS, PE Android, emulating the

A. Leon-Garcia et al. (Eds.): Smart City 2015, LNICST 166, pp. 66–77, 2016.
DOI: 10.1007/978-3-319-33681-7_6

path of Security-Enhanced Linux (SELinux), which is now part of the Android OS [1]. The PE Android concept was created to advance the goals of the Brandeis program from the US Defense Advanced Research Project Agency DARPA [2]. The Brandeis concept is that advanced technology now exists to empower human users to much better understand and selectively control the use of their private information.

The acronym for the PE Android project is ***PEARLS***, for ***Privacy-Enhanced Android Research and Legacy System.*** As the name suggest, PEARLS will address privacy enhancements that are deployable on current and future mobile hardware. As a demonstration of the PE Android capabilities, we have designed an enhanced application called RapidGather, an emergency response application that enables rapid, targeted data gathering while ensuring user control over degree of anonymity and data privacy. The goal of PEARLS is to enable effective development of new applications like RapidGather, as well as privacy monitoring tools capable of addressing data theft, driving widespread adoption of privacy-preserving technologies across mobile devices.

2 Related Work

There are two areas of technical advances that are fundamental to the PE Android project: advanced cryptography and sophisticated privacy intent and deployment schemes for Android. We discuss each of these in turn.

2.1 Advanced Cryptography

Since the seminal work of Andrew Yao in the 1980's [3], cryptography has provided techniques for groups of individuals to make effective collective decisions without revealing their specific individual data. The paradigmatic example is the Millionaires' Problem, where Yao shows how two honest individuals can jointly determine which of them is richer without a trusted third party and without revealing any specific information about their actual wealth. This topic area, called Secure Multiple Computation (SMC), has been refined over the years to become very general and much more efficient. Just recently, deployment of SMC to smart phones has been achieved by David Evans' group at the University of Virginia [4].

In addition to SMC, there are new techniques in homomorphic [5] and functional [6] encryption that will support smart cities applications. Homomorphic encryption enables a powerful third party to receive encrypted information from mobile devices, process that information according to a set of rules, and return the updated data to the mobile devices, all without decrypting the data. Functional encryption allows a mobile user to encrypt data that can then be decrypted in multiple contexts. For example, location data can be functionally encrypted so that one set of decryption keys will provide approximate location data while another set will provide exact location.

2.2 Android Privacy Research

The PEARLS team is keenly aware that providing sophisticated privacy options on mobile device can create a complex burden on users for determining the appropriate privacy settings. The new M release of Android is a good step forward in dealing with selective permission sets for conventional apps. Do [7] provides similar capabilities by selectively removing permission-protected functionality from individual apps. Kelly [8] and Felt [9] describe how to create permission settings that are comprehensible to users; Pantel [10] discusses inferring permissions from analysis of user intent. Beyond Android app permissions, other research such as Bugiel [11], has focused on applying lower level mandatory access control over system resources to flexibly augment the permission model. None of these approaches address control over private data that was authorized by the user to be collected by the application, nor apply other privacy preserving techniques such as differential privacy. The closest existing research is Wagner [12], which addresses the privacy of data collected by instrumentation built into the Android OS. Unlike PEARLS, however, this system does not provide specific privacy preserving functionality to individual apps nor provide the user with fine-grained control over their private data.

3 Project Description

PEARLS will enhance the Android platform with a new set of privacy-preserving technologies, including operating system (OS) modifications, graphical user interfaces, and cryptographic libraries. The immediate impact of these enhancements will be to support new privacy-preserving applications, which we will demonstrate by developing RapidGather, a data gathering application for emergency response.

PE Android will extend the Android OS with new privacy constructs to enable the deployment of Smart Cities applications such as RapidGather. Development of PE Android will begin on the PEARLS Research System, a high-performance computing environment where Android virtual machines can be rapidly created, modified, tested, and updated while running on a simulated cellular, Bluetooth, or WiFi network. An Android virtual machine (VM) in the Research System may be instantiated with large amounts of compute power and storage space as needed to effectively test new cryptographic techniques. The VM will be effectively unconstrained by battery life/power consumption or network interference (cell tower congestion, RF interference for WiFi/Bluetooth), and will support emerging and idealized sensor technologies (e.g., the ability to measure human vital signs or radiation, faster refresh rates, etc.).

We will also have the capability to integrate actual mobile devices in real-time with the Research System, as part of the effort to drive PE Android to a performance point where it can be proposed as an addition to the existing Android OS on mobile devices with standard computational power, networking technologies, cell connectivity, and sensor suites (accelerometer, microphone, camera, GPS, etc.), as well as a cloud server backend. Noting that legacy cell phones are a rapidly evolving target, the platform will easily accommodate new hardware and sensors as they become available.

3.1 PE Android-R

The Research System will use a special version of PE Android called PE Android-R (for research) with additional features for the purpose of fast and transparent prototyping. These will include enabling clean state virtual snapshots instead of "factory resets," providing direct access to the system cryptographic keys and sensor outputs, replacing the internal SQLite database with new data storage technologies, and enabling non-native code to run using external virtual resources (e.g., running code written in Haskell or Figaro instead of Java or C/C++). It will also enable extensive instrumentation hooks to accurately measure system performance and metrics. Finally, these additional features will be able to be easily disabled, providing direct compatibility for mobile device deployments and integration testing.

The PEARLS Research System consists of a high-performance virtual emulation platform and simulated environment and user models, and will run PE Android-R, which has additional features for rapid and transparent prototyping.

We will showcase the PEARLS platform with RapidGather, a PE Android app that allows emergency and crisis response teams to gather information quickly and anonymously from citizens at the scene of the event. Currently, it can take days or longer to track down the requisite information, as it depends on people voluntarily coming forward or on wide net investigation by the authorities. In the Boston Marathon bombing, for instance, even though word spread very quickly through radio, television, and social media, it still took days to gather the photos and other information needed to identify the suspects. With RapidGather, the response could be immediate, even as the crisis is unfolding. Authorities could anonymously communicate with RapidGather users in an area of interest and query them for available information, such as sensor data to locate gunshots or the epicenter of an earthquake [13], while providing privacy-preserving mechanisms for user responses.

PE Android will allow developers to create privacy-preserving applications without understanding the underlying primitives. Releasing it into the development community will enable revolutionary advances across a wide range of applications, including dynamic traffic routing and light synchronization, environmental sensing, and medical research. Similar techniques can also be applied to military contexts, such as coalition blue force tracking, where coalition partners could coordinate safe meeting locations without revealing identities. Android has a majority of the smartphone market share, and its potential impact goes beyond the mobile devices themselves to emerging and future capabilities, including the "Internet of Things" and body sensor network health monitoring, since the smartphone is the current locus of control for these emerging applications.

3.2 Crypto Tool Plan

PE Android will enable application users to have unprecedented control over the use of sensitive data by applications on their mobile device. As an example of such a technology, functional encryption can ensure that a user's privacy policy is enforced even when user data resides on an external server [5]. That is, only authorized entities can access user data and such entities can only learn particular functions of the user's data – as specified by the user's privacy policy. By incorporating functional encryption, or a similar technology, into PE Android, an application such as RapidGather can safely store user data on an external server – while maintaining assurance that authorities accessing that data can only learn information about a user that is permitted by the user's privacy policy. For example, a RapidGather user might functionally encrypt geolocation data so as to allow national anti-terrorism authorities to gain access to precise geolocation, while local authorities might only be permitted to derive an imprecise location with a random noise factor applied.

Additionally, PE Android will include a secure multiparty computation (SMC) capability, as this will allow applications to perform computations on data from multiple sources (e.g., multiple user devices, or a user device and an untrusted server) without any party learning another party's input data. For example, an application such as RapidGather could use SMC to enable a set of devices to anonymously aggregate location and sensor data to produce a joint incident report for authorities. Similarly, SMC could be used to perform a joint computation between a server – possessing a message from an emergency authority – and a set of user devices so that only those user devices meeting a certain criteria learn the message from the emergency authority. In recent years, there has been significant progress towards realizing SMC for certain applications on mobile devices [4]. Providing an easily usable SMC capability for any application is an exciting direction for PE Android.

PE Android can support a number of SMC implementations, such as a fully peer-to-peer SMC technology among mobile devices supported by secure out-sourced computation [14] and wireless networking, or a client-server SMC technology that offloads some of the cryptographic computation on one or more untrusted (or partially-trusted) servers [15]. It can also accommodate differential (and other noise-added) privacy approaches through a mediated sensor interface.

PE Android could also accommodate other potential technologies for privacy protection. For example, PE-Android could support differential privacy (or other noise-added) approaches by adding a mediated sensor to the Android platform. Similarly, PE-Android could support private information retrieval (or other advanced client-server technologies) by providing an abstraction layer that allows application developers to query such sophisticated services as simply as they can query traditional application servers.

3.3 Policy Tool Plan

PE Android will enable a broad and granular range of privacy choices, including various computational mechanisms and levels of added noise. These choices, if presented to a user without mediation, would quickly overwhelm them. One of the primary gaps in current Android systems is a lack of user understanding of the Android permissions model [8], which governs what data and functionality an app has access to. In particular, few users pay attention to the permissions during installation or are able to remember what they enabled [9]. This is exacerbated by small screen real estate and limited attention span due to users often being "on-the-go" when installing and using mobile apps. New technologies will close this gap by capturing the user's intention and succinctly displaying to them how their data is being used.

Therefore, PE Android would benefit most from the ability to effectively translate minimal user input to granular privacy policy. It will accommodate both at-installation and interactive mediation of privacy policy and will free users both from the tedium of defining a long privacy policy during app installation (as in Android L) and from constant interruptions from policy pop-ups (as seen on iOS). Useful new technologies would also reason about the impact of computations over the user's data, and mediate those interactions to provide effective policy enforcement. An interactive agent-based or online machine-learning solution that can continuously answer privacy questions, especially as context changes, would be particularly desirable.

We will develop an extensible PE Android Privacy API, which will accommodate a wide variety of interface mechanisms that communicate with and capture privacy intentions of the user, as well as flexibly convey privacy implications of those data choices. For example, this could involve using probabilistically generative models based on historical or societal information to warn users or filter content [17], asking simple policy questions ("yes/no" or "low/medium/high") to develop user models, or applying computational linguistics (such as machine translation or semantic web search [10, 18, 19]) to translate from terse user privacy statements to their implications in the data space and *vice versa*.

For applications such as RapidGather, where individuals may have little prior information in an emergency context to set a precedent, it would be helpful to use information gleaned from other users as well as social norms. For this purpose, we can support systems with various architectures, such as client/server machine learning approaches, or device-hosted autonomous agents that leverage peer-to-peer communication to collectively produce more accurate predictions of user intent. However, due to the potential latency and battery drain introduced by such architectures, external components should not be required to be "always-on."

4 Technical Plan

From "Yelping" a highly-rated nearby restaurant and "Instagramming" dinner to playing Angry Birds on the Uber ride home, mobile apps have transformed society. Their success is due to ubiquitous mobile platforms (Android) tied to distribution frameworks (Google Play) that enable developers to create wide-ranging applications and effectively distribute them. Our goal for PEARLS is to leverage those catalysts so that so that revolutionary advances in privacy-preserving applications can be made via the energetic mobile developer community and applied on a broad scale, thus ensuring the requisite user base to provide impactful social utility while maintaining user privacy.

The core of our extensible PEARLS platform is a privacy-enhanced Android OS fork, PE Android, which enables application developers to use the rich new privacy technologies without needing to understand the underlying cryptographic detail. This includes modifying the Android OS to enable privacy-preserving computation and human-data-interface functionality, integrating best-of-class technologies into main

Android libraries and OS modules, and extending the API to expose those capabilities and support seamless extensibility of new privacy-preserving technology, to include new ways for end users to control what data leaves their device.

We will demonstrate the utility and accessibility of PE Android by developing an emergency response app for it, RapidGather, which enables responders to anonymously communicate with mobile users and privately gather information from them (location, photos, videos, sensor data, etc.) to provide rapid crisis response and save lives. After RapidGather has been successfully prototyped, we will test it with PE Android on current mobile devices in a real urban location.

To achieve widespread adoption and enable practical privacy-preserving computation on everyday mobile devices, we plan to integrate the efficient components of PE Android back into Google's Android Open Source Project (AOSP). Taking inspiration from the successful mainlining of *Security*-Enhanced (SE) Android into Google's AOSP, the PEARLS team will leverage our extensive experience in creating and supporting stable, custom releases of Android on the DARPA Transformative Applications Program, and work with the Android community to effectively mainline PE Android. In parallel, we will also open source PE Android-R, so that the researchers can continue innovating on our system.

Once we have developed alpha versions of PE Android and RapidGather, we will open source them and begin engaging the AOSP project and the Android app developer community to build momentum for our system and solicit feedback from the community. As our code matures, we will submit patches to the AOSP project to be considered for mainlining into the official base. Similar to SE Android, the PEARLS team will provide build support instructions to add PE Android enhancements to the base AOSP tree directly, so that early adopters can start developing new privacy-preserving applications. To mitigate adoption challenges similar to those faced by SE Android, which had very expressive but complex security policies, PE Android will utilize technologies that support the definition of expressive privacy policies without undue developer expertise or effort.

4.1 Privacy-Enhanced Android

Enabling protections for sensitive data can be done in a number of ways on mobile devices. Protecting data at the app level, or through a third-party library, are two methods that can lead to a fractured app ecosystem, with privacy-preserving apps that may be incompatible with each other. However, the PEARLS approach makes modifications at the OS layer to introduce data privacy protections. This approach bakes strong data protection assurances directly into the OS and creates an ecosystem for new applications that can leverage Brandeis data privacy protections. PE Android also integrates the creation and enforcement of privacy policies – governing a user's private, identifiable, and sensitive data, including raw sensor readings – directly into the OS.

The PEARLS team will create PE Android by modifying the OS to provide extensions to the Android runtime API. PE Android will also provide hooks for our CRT partners to easily integrate their technologies into the Android OS. When research and development of PE Android is complete, the PEARLS team will supply patches to the mainline Android OS.

There are a number of areas suitable for extending and adding hooks for Brandeis technologies. As indicated by the dotted lines in the Android Architecture diagram, Fig. 1, the PEARLS team will extend and modify the

- Android Runtime API and core libraries,
- cryptographic libraries,
- *Location* and *Sensor Manager*,
- *Activity Manager* and *Binder*,
- *Surface Manager* and/or *Material Theme*.

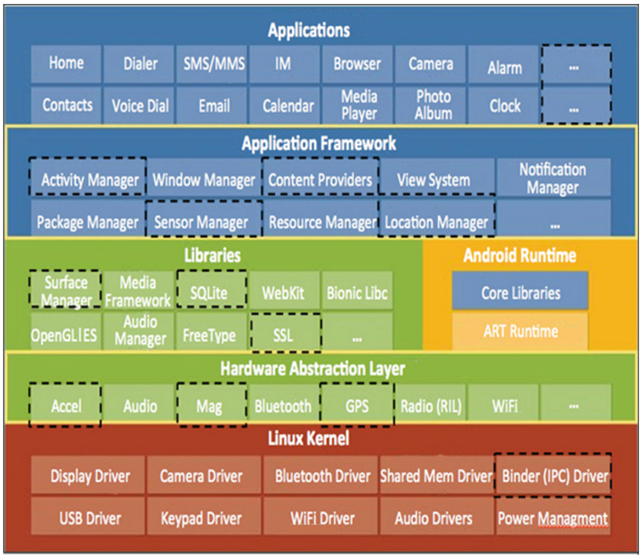

Fig. 1. Locations of PE Android Modifications

Android apps primarily make use of OS services through the Android Java Runtime and core libraries. We will modify the Android Java Runtime to expose the new privacy controls to app developers. The new PE Android Java Runtime API can be exported and used by app developers and other performers until PE Android patches are available and accepted by the Android community. In addition to the Android Java Runtime, we will modify some of the core system *Service* and *Content Provider* classes. For example, will modify the *Contacts Provider* and *Calendar Provider* to ensure data privacy policies are enforced.

Often, Android applications will store private information about the user on the Android device. On Android, the primary means to store and access information is through the use of SQLite databases. To enable existing and future apps to leverage new capabilities, PE Android will provide an interface for app developers to query data from a SQLite database with options such as added noise or cryptographic data transformations.

The Android OS supplies several cryptographic facilities for use by the system including

- the kernel crypto API provided for use by kernel modules,
- OpenSSL libraries for use in native Android binaries, and
- the BouncyCastle implementation of the Java cryptographic extensions.

Additionally, Android supplies its own keystore (either software or hardware backed depending on device) and API to access it. In Android's current state, these cryptographic facilities and keystore may not be enough to support new technologies. Where appropriate, we will add additional primitives and API extensions to PE Android to support new technologies, including extended key management functionality.

Current location and sensor information is highly sensitive yet required for applications like RapidGather. Android apps request the location and sensor information through calls to the *Location Manager* and *Sensor Manager*, which in turn request sensor data from the Android Hardware Abstraction Layer (HAL). We will extend the sensor interface in this framework so that PE Android is aware of the current privacy context, and will provide hooks for adding noise to the location and sensor information coming from the hardware. PE Android can thus enforce mediated access to the sensor information based on the privacy context and obfuscate the sensor information before it goes back to the consumer of the data.

In addition to adding privacy hooks to various Android subsystems, PE Android will allow a privacy policy to be defined, configured, and enforced. This policy can also address theft of private data, by providing users visibility into what is accessing and transferring their private data. The Android *Activity Manager* keeps track of the various running tasks and apps on the system. PE Android will add information about the data privacy context under which a task or app is running so that it can be referenced and used throughout the Android system to help enforce the privacy policy.

One such example is the Android *Binder* driver, which is an Android-specific IPC mechanism that controls most of the data flowing between processes. Applications that don't make use of *Binder* directly will often use other Android systems built on top of *Binder* to send or receive data, like Android *Intents* or *Content Providers*. Since the majority of data passed among Android processes is funneled through the *Binder* driver, it is an ideal place to enforce the privacy contexts of running applications. We will modify *Binder* so that PE Android can pass the privacy context information of the data and ensure that only tasks or applications with the appropriate privacy policy receive the data.

By tracking data throughout the system with privacy policy information, as we propose to do with *Activity Manager* and *Binder*, we can then begin to display the current privacy mode and/or private data that may be emitted from the phone to the end user. We will modify the *Surface Manager* and the *Material Theme* to provide hooks and an API for changing the user interface (UI) based on the privacy mode currently in force. This UI, coupled with privacy policy extensions, will enable application developers to integrate their technologies into the system.

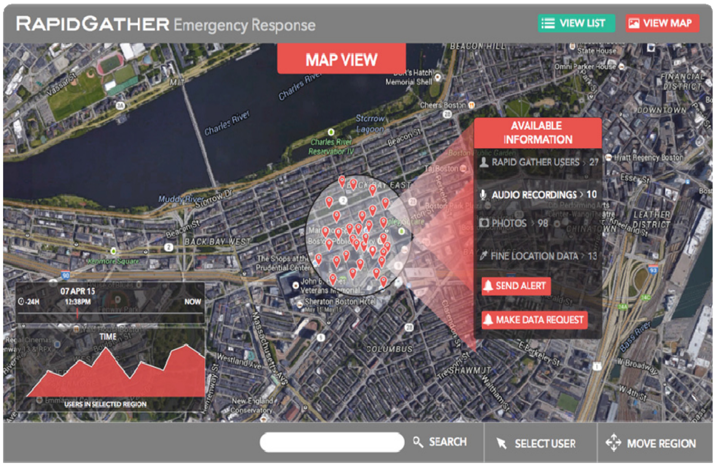

Fig. 2. RapidGather response team (server) interface shows anonymized locations of RapidGather users. Responders can select users to send messages or request information for a region of interest.

4.2 Smart City App: RapidGather

We will showcase the utility of our approach with RapidGather, a Smart City application that leverages mobile crowd sensing to provide rapid emergency response and save lives. Specifically, it enables emergency and crisis response entities (e.g., ambulance, fire, law enforcement) to communicate quickly and anonymously with, and gather data from, mobile users in the midst of unfolding events, while providing users with granular control over how their private information (location, photos, microphone, sensor data, etc.) is used. Since the effectiveness of responding to emergencies or in catching perpetrators is a function of the speed and robustness of the data gathered, RapidGather will greatly enhance the current state of crisis response techniques, and provide natural transitions for other revolutionary advances in public health and safety.

The RapidGather server component will provide a robust back-end and enable authorities to automatically contact an anonymized list of people (e.g. through pub/sub or oblivious transfer mechanisms) who are or were in the location of interest with pertinent alerts ("users in this area have an elevated health risk"), questions ("have you seen license plate XYZ-123?"), and open-ended requests for information ("who has information to share about event X?"). A mockup of the response team server interface that we will develop is shown in Fig. 2, with the icons in the circle representing users with RapidGather-enabled mobile devices at the scene of interest. At a glance, responders can determine how many people are in a chosen area and what information is available, while providing privacy-preserving mechanisms for sending alerts and allowing users to respond with variable granularity.

Data owners will maintain control over their data, with a range of options on the mobile device to elect what information they are willing to share, and the granularity with which it is reported. For example, they could share their general location with varying noise added, depending on population density and their proximity to ground zero for the crisis, or they could allow only some specific function of their location to be securely computed (such as gunshot localization). A mockup of the RapidGather client interface is shown in Fig. 3.

The RapidGather application will leverage PE Android to provide a rich landscape to navigate the range of user privacy options (e.g., translating high-level user concerns to granular privacy policy or using past user behavior to predict intent). It will also support a variety of architectures and mechanisms, such as peer-to-peer secure multiparty computation (e.g., localizing shots fired or aggregating sensor data), functional searches over encrypted data, or outsourced mobile computation through homomorphic encryption, enabling exploration of utility/privacy tradeoffs.

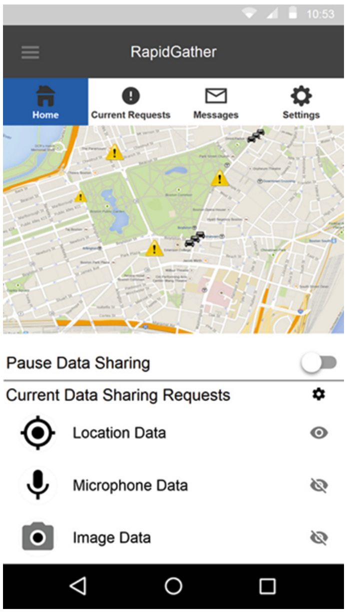

Fig. 3. RapidGather user/client interface

5 Summary

This paper has described the plan for a new extension to Android OS, called PE Android, which includes new data privacy protection and management tools aimed at supporting Smart Cities applications. Our goal for this project is to make the advances in privacy-preserving technologies available to the mobile developer community, so that they can be broadly applied and provide the impactful social utility envisioned by Smart Cities, while maintaining user privacy.

References

1. http://source.android.com/devices/tech/security/selinux/. Retrieved 30 June 2015
2. US Defense Advanced Research Projects Agency: Brandeis, http://www.darpa.mil/program/brandeis. Retrieved 30 June 2015
3. Yao, A.C.: Protocols for secure computations. In: FOCS, 23rd Annual Symposium on Foundations of Computer Science (FOCS 1982), pp. 160–164. doi:10.1109/SFCS.1982.88
4. Huang, Y., Chapman, P., Evans, D.: Privacy-preserving applications on smartphones. In: 6th USENIX Workshop on Hot Topics in Security (HotSec 2011), San Francisco, August 2011
5. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Symposium on the Theory of Computing (STOC), pp. 169–178 (2009)
6. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)

7. Do, Q., Martini, B., Choo, K.-K.R.: Enhancing user privacy on Android mobile devices via permissions removal. In: 2014 47th Hawaii International Conference on System Sciences (HICSS). IEEE (2014)
8. Kelley, P.G., Consolvo, S., Cranor, L.F., Jung, J., Sadeh, N., Wetherall, D.: A conundrum of permissions: installing applications on an android smartphone. In: Blyth, J., Dietrich, S., Camp, L. (eds.) FC 2012. LNCS, vol. 7398, pp. 68–79. Springer, Heidelberg (2012)
9. Felt, A.P. et al.: Android permissions: user attention, comprehension, and behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security. ACM (2012)
10. Pantel, P., Lin, T., Gamon, M.: Mining entity types from query logs via user intent modeling. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers, vol. 1. Association for Computational Linguistics (2012)
11. Bugiel, S., Heuser, S., Sadeghi, A.-R.: Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In: Usenix security (2013)
12. Wagner, D.T. et al.: Device analyzer: a privacy-aware platform to support research on the Android ecosystem. In: Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks. ACM (2015)
13. Minson, S.E. et al.: Crowdsourced earthquake early warning. Science Advances **1**(3), 10 April 2015. http://advances.sciencemag.org/content/1/3/e1500036
14. Carter, H., Amrutkar, C., Dacosta, I., Traynor, P.: For your phone only: custom protocols for efficient secure function evaluation on mobile devices. Secur. Comm. Netw. **7**, 1165–1176. doi:10.1002/sec.851
15. Carter, H., Mood, B., Traynor, P., Butler, K.: Secure Outsourced Garbled Circuit Evaluation for Mobile Devices. In: Proceedings of the 22nd USENIX Security Symposium, August 2013, Washington, D.C. (2013)
16. Bogetoft, P., et al.: Secure multiparty computation goes live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
17. Peng, H. et al.: Using probabilistic generative models for ranking risks of android apps. In: Proceedings of the 2012 ACM Conference on Computer and Communications security. ACM (2012)
18. Roy, R.S. et al.: Discovering and Understanding Word Level User Intent in Web Search Queries. Web Semantics: Science, Services and Agents on the World Wide Web (2014)
19. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**, 195–197 (1981)