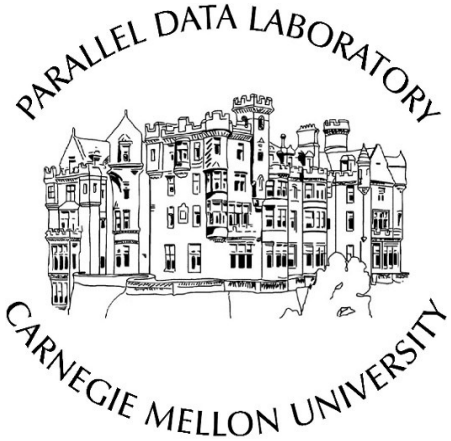

Challenges in Security and Privacy for Mobile Edge-Clouds

Jiaqi Tan, Rajeev Gandhi, Priya Narasimhan

CMU-PDL-13-113
Oct 2013

Parallel Data Laboratory
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Mobile devices such as smartphones and tablets are ubiquitous today, and many of them possess significant computation power, powerful sensors such as high-resolution cameras and GPS sensors, and a wealth of sensor data such as photos, videos, and location information. Collections of mobile devices in close geographical proximity present both opportunities and challenges for mobile applications: the enormous collection of mobile devices presents an extremely rich source of user-generated content as well as collective computation power, but these devices are mutually distrusting, and security and privacy concerns, amongst many other obstacles, prevent users from cooperating with other distrusting entities to exploit both the available computation power and data. In this paper, we articulate and describe some of the security and privacy challenges which currently prevent us from leveraging the collective data and computational power available in collections of mobile devices belonging to mutually distrusting users. By addressing these security and privacy challenges, we envision that a new class of applications can be developed which leverage the collective mobile devices available in close geographical proximity: mutually distrusting users would be willing to participate in such public computations with sufficient security and privacy safeguards, enabling novel applications.

Acknowledgements: We would like to acknowledge Utsav Drolia and Rolando Martins for the insightful discussions regarding the ideas in this paper.

Keywords: Mobile Computing, Edge-Clouds, Security, Privacy, Hyrax, MapReduce

1 Introduction

The ubiquity of mobile personal devices such as smartphones and tablets today, with their growing computational and storage resources, together with their high-fidelity sensors (e.g. high-resolution cameras, fine-grained location sensing) available on them, has created unique opportunities and challenges for novel applications. As mobile device owners carry their devices everywhere with them, and use them in many different scenarios, mobile devices today capture a wealth of data about their owners and their surroundings, as well as rich contextual information accompanying the data, such as time and location. Collectively, the mobile devices carried by users in close geographical proximity presents a pool of untapped resources, in terms of processing and storage capacity, as well as rich data in the form of contextually-enhanced sensor data. We propose the notion of mobile edge-clouds [15], which are clouds comprising mobile nodes with high mobility, such as smartphones and tablets, as their compute nodes, as well as the source of data. However, the security and privacy of users' mobile devices and their personal data on these devices presents a significant obstacle to users from allowing their mobile devices to participate in such an mobile edge-cloud, especially in public venues when mobile edge-clouds comprise mutually distrusting nodes belonging to unrelated persons. In this paper, we explore the challenges in security and privacy for mobile edge-clouds that must be tackled to give users of mobile devices sufficient assurance to allow their devices to be used in such edge-clouds.

1.1 Mobile Edge-Clouds

We propose the notion of mobile edge-clouds, which are clouds comprising mobile nodes with high mobility, such as smartphones and tablets, as their compute nodes, as well as the source of data, operating in close geographical proximity. The mobile devices carried by users in close geographical proximity, as a collection, provides both processing and storage capacity, and rich, contextually-enhanced sensor data. First, the processing and storage resources of mobile devices in a local physical area can be pooled into a single computing resource to form a mobile compute (and storage) cloud. This is particularly pertinent in environments where high bandwidth, low latency connections to the Internet may be limited, for instance in massive crowds, or in a disaster response scenario where cellular networks may have failed. Second, the rich sensor data available on all the mobile devices can be combined in meaningful ways to enable novel applications to be developed on top of the data. Hence, the main functional benefits to using an mobile edge-cloud are that (i) *it provides additional processing and storage resources beyond that available on a single node* during times of limited or absent cellular communications, and (ii) *it provides users with rich data available on nearby nodes not available on a node itself*.

The users of an mobile edge-cloud would be owners of devices who are undergoing a similar experience in close physical proximity for a sustained amount of time, who would have similar computational goals. For instance, spectators at a large sporting event may want to capture photos or videos of the best views of the game, but they may not have the best vantage point to capture the best photos or videos, or they may want to have photos or videos from vantage points other than their own. Likewise, spectators may also want to perform additional processing on their photos and videos, such as perform video stabilization, and this task may be computationally-intensive. Hence, spectators would be incentivized to participate in such an edge-cloud: by volunteering their data, they can obtain additional computational power from neighboring devices, and by volunteering the computational resources of their devices, they can obtain useful data. It may not be feasible for users to upload all their data to a central location due to infrastructural limitations at such large-scale public events [16]. In other scenarios, communications infrastructure may not even be available, e.g. in a disaster-response scenario, and users will have only their mobile devices to rely on.

Hence, the goal of an mobile edge-cloud would be to enable participants in the cloud to cooperate for a short period of time to exchange data and computation in order to complete a computational task

whose result the participants are interested in, without requiring a central processing system. We envision that edge-clouds can enable users to efficiently perform computation tasks by obtaining data from other edge-cloud nodes which may own data required for the computation, and by sharing the load with other edge-cloud nodes which are also interested in the results. The key differences between edge-clouds and traditional compute clouds are that: (i) the latter comprise stationary servers, often located in data-centers, with excellent network connectivity, whereas edge-clouds have highly mobile nodes, and may have poor or even intermittent network connectivity, and (ii) traditional clouds provide storage and compute capacity, and users provide their own data. In mobile edge-clouds, besides providing storage and compute capacity, nodes also provide the data for the computation, for instance when smartphone users contribute photos, videos, or other forms of data captured on their smartphone, for the computation.

1.2 Security and Privacy Challenges

Security and privacy concerns of users are a key obstacle deterring users from allowing their mobile devices to be participants in an edge-cloud. In this paper, we describe some of the security and privacy goals that we believe must be met for users to be convinced to participate in an edge-cloud, providing data, storage and computation on their personal mobile devices.

The first key security challenge is that all users of an mobile edge-cloud are mutually distrusting. In providing computation resources to other users, participants of an mobile edge-cloud must execute untrusted foreign code received from other users. Allowing code from other edge-cloud participants to run on a smartphone poses a greater security risk than running third-party applications, as third-party applications downloaded through official vendor App Stores are typically subject to a vetting process to screen for malware and malicious apps, whereas mobile edge-clouds are unable to provide a central App Store nor vetting process due to the transient nature of mobile edge-clouds.

The second key challenge is that for mobile edge-cloud applications to be useful, users must be able to contribute data which they own on their mobile devices to the application. However, as mobile devices contain potentially privacy-sensitive personal data, such as contact information, photos, videos, and location information, users would be concerned about: (i) whether the data they share with the edge-cloud is privacy-sensitive, and (ii) whether the edge-cloud application is able to access data on the owner's device which the owner did not intend to share with the edge-cloud application.

The third key challenge is that of identity. Given that the nodes of edge-clouds are mobile devices whose owners are in close physical proximity for a short duration (e.g. for several hours at a sporting event), these mobile devices participating in the edge-cloud are likely to have never interacted with each other. In a fully mobile setting with no central processing site, it would be impossible to utilize any security mechanism which requires pre-arranged roots of trust, such as a public key cryptosystem. Hence, it would not be possible to establish identities of nodes beforehand, and it would not be possible to make use of identity-based solutions, such as message-signing or even code-signing, to gain trust in foreign code being executed.

In summary, the key security and privacy challenges facing mobile edge-clouds, are that each mobile device owner participating in the edge-cloud is an untrusted principal, there are data privacy concerns as participating devices store privacy-sensitive data, and securing communications among nodes is extremely challenging as pre-establishing identities is not possible given the transient nature of mobile edge-clouds.

2 Functional Goals

We begin by describing the high-level features an mobile edge-cloud would aim to provide, and the accompanying security and privacy concerns users might have regarding these features when considering whether

to allow their personal smartphones to be part of an mobile edge-cloud. Then, we outline the types of security and privacy guarantees users would find desirable and/or sufficient. We describe these desired features in ascending order of sophistication, and we begin with the simplest features that an mobile edge-cloud would provide.

2.1 Remote Data Access

In its simplest form, an mobile edge-cloud can allow nodes to upload their data to a central location for some computation to process it, before each node is provided with a result aggregating the data from all the nodes in the edge-cloud. This would require nodes to either submit their own data, or allow the mobile edge-cloud to remotely access some (or all) of the data on the node. Both options would require smartphone owners to give access to some or all of the potentially private data on the node to the untrusted edge-cloud.

2.1.1 Privacy of Data

One concern edge-cloud users would have with both sending data on the node to the edge-cloud, and allowing the edge-cloud to remotely access data on the node, is that this data should not be privacy sensitive. For instance, spectators at a ballgame may be willing to share photos and videos of the game with the edge-cloud, but they not be willing to share these photos and videos if they contain images of the users themselves in the photos and videos. Hence, users should have mechanisms available to help them decide if data being shared with an edge-cloud is privacy-sensitive.

2.1.2 Isolation of Data

Another concern edge-cloud users would have with sending or allowing remote access to data on their nodes, is whether the edge-cloud can access only data that the user intends to share with the edge-cloud, or whether the edge-cloud can also access all other potentially private data on the node, and exfiltrate the data, leading to exposure. The edge-cloud should be able to provide users with mechanisms to ensure that only the data they intend to share with the edge-cloud is exposed, and that no other data is exfiltrated from the node. It would also be desirable for these data isolation mechanisms to be themselves trustworthy, to increase confidence in the security of the edge-cloud.

2.2 Remote Computation

In an mobile edge-cloud environment, network bandwidth usage can be reduced by allowing remote computation, so that small pieces of code can be sent to each node to operate on the node's data, thus eliminating the need to send all data from each node to the edge-cloud. This has two effects on the security and privacy of the edge-cloud: with remote computation, each node's data does not have to leave the node, thus improving data privacy; however, since nodes in an edge-cloud are mutually distrusting, code from other nodes is also untrusted, and running untrusted code on a node can pose security challenges.

2.2.1 Securely Executing Untrusted Code

To allow remote computation, an mobile edge-cloud must allow untrusted code from other nodes to be executed securely on the node. At a high-level, the untrusted code must not cause any harm to the node, and the untrusted code should only have behaviors necessary for completing its task.

2.2.2 Verifiable Execution

At the same time, the untrusted code in an mobile edge-cloud is also executing on a potentially hostile node. Nodes can return bogus results to avoid executing the remote code, for instance to save energy while still appearing to participate in the computation. Hence, an mobile edge-cloud should be able to verify that the remotely executed code did indeed execute correctly, and that it produced the correct results.

2.3 Context-aware Computing

Finally, an mobile edge-cloud can make use of the data on a node for not just computing results, but also for making scheduling and other systems decisions as part of its execution. For instance, a computation can use the location data of a node to help aggregate results, by sending a computation to nodes in the edge-cloud to instruct nodes close to each other to query each other to select the highest quality photo in that physical location. In such cases, users of nodes should be given control over whether to allow particular sensors or other data sources on the node for computation decisions. The mobile edge-cloud should also be able to identify whether contextual data on a node is being used as data in a computation, or whether it is being used as contextual data to assist the edge-cloud in making systems decisions.

2.3.1 Context Privacy

Nodes in an mobile edge-cloud should be able to provide users with controls over context data, such as location information. Users should be able to set policies on acceptable uses for this data, and the mobile edge-cloud system should then respect these policies and provide various options to users for policy violations, such as replacing context information with incorrect or less precise versions, or completely disallowing the computation.

2.4 Communication Substrate

Finally, the mobile edge-cloud will also require a number of auxiliary features which support the operation of the mobile edge-cloud itself. It would be challenging to implement and deploy traditional cryptographic systems in an mobile edge-cloud, such as a public-key cryptosystem. Nonetheless, an mobile edge-cloud should provide features for establishing and managing the identities of participants, and for authenticating nodes. However, it is likely that the notion of identity and authentication would be different in a mobile edge-cloud with mutually distrusting participants and no pre-established cryptographic material. In addition, the mobile edge-cloud is likely to harness various protocols beneath the application-level logic for providing the mobile edge-cloud functionality. These protocols and network communications also need to be secured against malicious attackers and adversaries who may passively or actively attempt to eavesdrop on and subvert communications between edge-cloud nodes for various goals.

3 Problem Statement

The goal of this paper is to identify challenges in security and privacy for an mobile edge-cloud comprising of mobile devices such as smartphones and tablets as its nodes. These measures would *convince* mobile users that (i) their mobile devices would not be adversely affected and that (ii) their privacy and personal data would not be compromised by the execution of the untrusted code, while they enjoy the benefits of using an mobile edge-cloud, hence convincing them to allow their mobile devices to serve as nodes in an mobile edge-cloud. We hypothesize that an mobile edge-cloud must provide these security properties to convince mutually distrusting users to participate in the edge-cloud, allowing untrusted code to run on the edge-cloud, and submitting their tasks to untrusted nodes in the edge-cloud.

Stakeholder	Category	Property	Sub-properties
Provider	Isolation	Process Isolation	Execution isolation Memory isolation Filesystem/Data isolation
Provider/User	Isolation	Data Access Management	
User	Computation	Quantifiable inaccuracy/correctness Trustworthy/Verifiable Computation	
Provider/User	Lack of Permanence		

Table 1: Desirable security properties of an mobile edge-cloud.

3.1 Threat Model

We consider all mobile users in an edge-cloud to be mutually distrusting. Mobile users can submit arbitrary processing tasks: these tasks are potentially malicious, and may try to exfiltrate private user data from the host smartphone on which they execute, or may try to corrupt or render their hosts unusable. In addition, mobile nodes have arbitrary behavior, and may not correctly or faithfully execute the submitted tasks: nodes may falsify results to try to convince users that the tasks have been executed although they have not. However, we exclude network threats from our scope (e.g. attempting to impersonate other nodes, falsify traffic, or deny service at the network layer to other nodes), as we focus on software security.

3.2 System Environment and Model

We envision an mobile edge-cloud comprising of mobile devices such as smartphones and tablets in geographical proximity with each other as its nodes, with local network connectivity among the nodes e.g. through WiFi, but with poor or no cellular connectivity and hence limited connectivity to the Internet. Users of the mobile edge-cloud submit processing tasks to the edge-cloud, and these tasks make use of the processing resources, and potentially the local data on each node, for its computation. The edge-cloud allows any user to submit processing tasks to be run on a subset of, or all, nodes in the edge-cloud, and each task may make use of the data present on each smartphone (subject to limitations to protect privacy). Each smartphone, or node, in the edge-cloud, acts both as a **provider**, when it allows submitted tasks to run on its own processor and use its data, and as a **user**, when it submits tasks to be run on other nodes in the edge-cloud.

Further, we assume that mobile devices in the mobile edge-cloud run on a commodity operating system, such as Google's Android or Apple iOS, and that user-submitted tasks run as unprivileged code in user-space, either as a mobile application (*app*), or as part of an app which runs the submitted task. The tasks submitted by nodes in an mobile edge-cloud can be custom user-created code, and these tasks, unlike installed Apps on commodity mobile operating systems, do not have a vetting process such as Google's Play Store for Android or Apple's App Store for iOS, due to the dynamic nature of these tasks. Submitted tasks in our mobile edge-cloud must be accompanied by a description of the functionality of the task, a description of the processing done by the task, and the local data on each provider that the task would like to use in its computation.

4 Security Properties

There are two types of stakeholders in an mobile edge-cloud: (1) the owners of each mobile device and of the (potentially private) data on the smartphone, whom we call the **providers**, and (2) the **users** of the mobile edge-cloud who submit jobs to be processed on mobile nodes, who are interested in the results of the computation. We consider the security properties that would be desirable to both the providers and users

of an mobile edge-cloud; it should be noted that each mobile device in an mobile edge-cloud acts as both provider, when it is executing a task on behalf of the user, and a user, when it is submitting a task to or receiving results from other nodes. Hence, *providers* and *users* can be thought of as roles in our system.

Next, we present a hypothesized list of security properties desirable to providers and users. Abstractly, the key security guarantees that providers would require, is that the providers' mobile devices are **isolated** from any possible malicious effects of code in submitted user tasks, while users would require that the **trustworthiness** of the computation of their submitted tasks can be managed. We summarize the desirable security properties in Table 1.

4.1 Providers

Providers of the mobile edge-cloud, or mobile device owners, would run tasks submitted by users. As the code in user-submitted tasks is untrusted, the execution of its code must be *isolated* from the rest of the provider smartphone to protect it from malicious task code. In other words, the submitted task must not be able to access the provider smartphone's data nor affect its behavior in unauthorized ways. In addition, these submitted tasks can potentially use require the use of data on each provider's smartphone, which we will call *provider data*. This provider data required by the tasks may contain private data relating to the owner of the provider smartphone. Specifically, the task code must **not** be able to:

- Read from or write to privileged files owned by the provider mobile device's OS;
- Read from or write to files and other persistent storage owned by other applications on the provider mobile device;
- Affect or alter the behavior of the provider mobile device's OS in unauthorized ways (i.e. in ways that are not part of the submitted task's functionality);
- Affect or alter the behavior of other apps on the provider mobile device;
- Read from or write to memory belonging to other apps on the provider mobile device, or to the OS on the provider mobile device;
- Access or modify data on the provider mobile device other than the authorized data.

4.2 Users

Users of the mobile edge-cloud would submit tasks to the edge-cloud, to be executed by mobile devices acting as providers to the edge-cloud by allowing tasks to be run on their mobile devices. We call these tasks *user tasks*. Users are interested in the results of the computation in the submitted user tasks. Providers have arbitrary behavior, and may or may not faithfully execute the user tasks, and may or may not provide accurate values for the requested provider data. At a high-level, we would like the edge-cloud to enable the execution of user tasks to be managed. This should allow providers to provide less-than-accurate results, and communicate the amount of inaccuracy injected for the purposes of preventing privacy loss of the provider. We would also like the edge-cloud to be able to measure the degree to which each provider's results can be trusted, independently of each provider's results. Hence, the goal of the edge-cloud's execution of user tasks is to provide the following guarantees:

- Either the results from the computation of the user tasks are correct, or the degree of uncertainty in the results can be quantified or qualified;
- Either the results from the computation of the user tasks are trustworthy, or the degree of untrustworthiness can be quantified.

We consider uncertainty in the results of user tasks to be Mathematical uncertainty, such as in the notion of ε-differential privacy [17]. We defer the discussion of what it means for the results of user tasks to be trustworthy to a survey of techniques for trustworthy computation.

4.3 Privacy: Lack of Permanence

Users of the edge-cloud would find it desirable that the execution of their submitted user tasks on each provider node leaves no trace of its execution behind on the provider after the execution has been completed, and after the results have been sent back to the user node. This would protect the privacy of the user, as it would prevent subsequent users, or even the provider, from learning about the user task that was executed.

4.4 Security Properties for Functional Requirements

In addition to the key security properties of **isolation** and **trustworthiness** in the face of mutually distrusting providers and users, additional functionality is needed for an edge-cloud, and these mechanisms must also be secured from unauthorized actions by both users and providers.

4.4.1 Authenticated Addressing and Messaging

In an edge-cloud, without a central coordinating authority, participants in the edge-cloud must be able to locate and address each other, and must be able to transport messages among themselves. Users and providers must also be able to establish and verify the authenticity of messages sent/received to/from each other to prevent spoofing and impersonation.

Users must also be able to securely dispatch computing tasks to providers for execution, and securely receive results from task execution. Mechanisms for authenticated message transmission will provide the necessary guarantee that providers know the true user which submitted a task, and that users will know the true provider which returned a particular result.

4.4.2 Node Entry and Exit

In an edge-cloud, as nodes are made up of smartphones owned by mobile users, the geographical location of nodes can be expected to be continuously changing. As we consider the use of limited-range wireless communication (i.e. WiFi networking), there is likely to be considerable numbers of nodes entering and leaving the edge-cloud. Hence, the processing of user tasks in the edge-cloud must be able to tolerate the continuous entry and departure of nodes to the edge-cloud. Specifically, the edge-cloud should be able to provide *progress* in the execution of user tasks in the face of a given rate of node entry/exit.

4.5 Summary

Hence, here is a short summary of the properties that we desire of our secure mobile edge-cloud system:
Isolation: Provider mobile devices must be isolated from any potentially malicious actions of user code.
Trustworthiness: Users must be able to ascertain the trustworthiness of the results from the execution of user tasks independently of providers.
Lack of Permanence: The execution of user tasks on providers must leave no trace of their execution behind on provider mobile devices.
Authenticated Addressing and Messaging: Nodes must be able to authenticate each other, authenticate messages sent/received to/from each other, and locate and transport messages to/from each other.
Tolerant of Node Churn: The edge-cloud must be able to make progress in executing user tasks in the face of a given rate of node churn.

5 Related Work

First, we review recent work on systems for collective computation and storage on collections of mobile devices. Then, we review some of the related work in security and privacy that could provide guidance for implementing our proposed security and privacy goals in mobile edge-clouds.

5.1 Collective Computing on Mobile Devices

Currently, the United States Army has proposed the notion of Content-Based Mobile Edge Networking (CBMEN) [3], which provides distributed storage on the smartphones carried by soldiers in a battlefield. However, CBMEN does not provide computation, unlike the mobile edge-clouds which we propose. In addition, CBMEN assumes that smartphones can be pre-set to install cryptographic material such as keys, thus allowing secure communications. However, in our mobile edge-cloud scenarios, it is not possible to carry out any coordination or management tasks such as key loading prior to processing.

A number of efforts have built distributed processing and storage frameworks for mobile devices. Hyrax [35, 40], which was developed by our group, was the first system to propose the idea of distributed cloud-computing on mobile devices, and ported the Hadoop open-source implementation of the MapReduce distributed parallel computing framework [12] to the Android mobile OS. However, Hyrax did not provide security guarantees for code executing on providers, nor privacy protection for data on providers. Lightweight MapReduce (LWMR) [13] is another implementation of the MapReduce framework for Android, while Misco [14] implemented the MapReduce framework, but for resource-constrained devices. Both LWMR and Misco, like Hyrax, did not address security considerations for executing untrusted mobile code as part of a distributed computation, nor the privacy considerations for using data from provider mobile devices.

5.2 Smartphone and Mobile Device Security

Google's Android OS for smartphones is one of the major operating systems for smartphones, with almost 80% market share for newly shipped smartphones for the second quarter of 2013 [1]. Coupled with the open-source nature of Android, much research has focused on security for Android smartphones.

The literature on smartphone security has examined various aspects of security on mobile platforms and smartphones. They have analyzed and proposed new permissions systems (e.g. finer-grained access control [11], formally modeled [5, 36, 39] and verified [22] access control mechanisms), statically analyzed mobile apps for poor security practices [10, 19, 23, 26, 21, 32, 9], proposed dynamic runtime analysis of mobile apps [18, 44, 42, 20, 25, 30, 29], and proposed various sandboxing and execution isolation techniques [7, 43].

Much of the work on Android security has looked at statically analyzing the bytecode of Android apps to try to determine if there are security vulnerabilities and insecure behavior. Some of this work applied static-analysis techniques to the bytecode of Android apps to study security-relevant aspects of their behavior, such as their use of permissions and Intents [10, 21]. ComDroid [10] used static-analysis to extract function calls which send and receive Intents to find instances of insecure Intent usage, such as unauthorized hijacking of broadcast Intents and Intent spoofing. ScanDroid [23] applied the WALA Java analysis toolkit to Android bytecode to identify information leakage via sensitive exfiltration functions (i.e. Inter-Component Communications/Intent sending functions). Woodpecker [26] used control-flow analysis to construct models of Android app execution, and used symbolic simulation to check the resulting control-flow graphs (CFGs) for instances of "confused deputy" attacks.

Dynamic analysis of Android apps during their execution has also been proposed to help check for malicious behavior. Dynamic analysis is generally more precise than static-analysis, as it does not suffer from false-positives, although it can only check actual executions, and it incurs overheads, which may be

too costly in terms of energy on battery-powered smartphones. TaintDroid [18] provides system-wide taint-tracking. TaintDroid taints data received from privacy-sensitive sources, such as the IMEI number, and transitively applies taint labels as the data moves through the system. DroidScope [44] builds on the QEMU emulator to enable VM introspection into Android at the ARM level, the Linux level, and at the Dalvik VM level to enable analysis of Android malware by security analysts. ProfileDroid [42] provides coarse-grained runtime monitoring and profiling of Android apps. ProfileDroid records all user interactions with an app, and collects traces of the execution of an Android app. Similarly to DroidScope, ProfileDroid is aimed at security analysts studying the behavior of a given app. AppFence [29] uses a combination of dynamic taint-analysis and a modified Dalvik VM to identify the movement of sensitive data in program code during execution, and to either block the exfiltration of sensitive data, or perform "data-shadowing", where sensitive data is replaced with non-sensitive or anonymized versions.

While Android isolates the execution of apps from each other, and it isolates the execution of apps from the OS, recent work has gone further to provide more restrictive sandboxing and isolation techniques to improve the security of Android. TrustDroid [7] implements a lightweight version of domain isolation first proposed as Trust Virtual Domains (TVD) [8, 27]. Aurasium [43] provides app-level sandboxing by intercepting system calls to dynamically enforce security policies.

5.3 Security and Privacy for Cloud-Computing

Techniques for establishing security and privacy in cloud-computing systems can provide guidance for security and privacy in mobile edge-clouds. Recent work has examined ways to build accountable systems, such as identity binding using cryptographic signatures [46], building tamper-evident logs [28], and logging cloud interactions for later verification [31]. However, these techniques require either a trusted verifier [28, 31], or public-key cryptography infrastructure, both of which are unlikely to be available in an mobile edge-cloud setting.

Specific security and privacy techniques have also been developed for MapReduce [12], which is particularly relevant as many mobile distributed computation systems have been modeled after MapReduce. SecureMR [41] uses task replication to defeat untrustworthy slave nodes that collude to return false results, while Airavat [37] deploys SELinux on Hadoop [2] hosts to provide Mandatory Access Control (MAC) to prevent information leakage, and provides differential privacy by restricting the computational model of submitted Maps and Reduces. Sedic [47] modifies MapReduce to allow it to run computations on private data on the private part of a hybrid cloud, while allowing only public data to be processed on untrusted cloud nodes; however, in our proposed mobile edge-cloud, all nodes are untrusted.

Techniques for secure cloud storage have also been proposed, to provide integrity for data stored on the cloud by auditing [4], and by using sentinel values in files to create proofs of retrievability (POR) [33].

5.4 Securely Executing Untrusted Code

Isolating execution attempts to provide security by restricting the effects of executing programs in one trust domain from programs in another trust domain. This will prevent any harmful effects of one program from affecting the data and/or execution of programs in another trust domain. The Xen hypervisor [6] introduced the concept of paravirtualization, which passes through most calls to the underlying hardware instead of providing full virtualization of all processor instructions, thus achieving a high degree of efficiency which made virtualization feasible for real-world applications. One of the key security benefits which virtualization provides is isolation: multiple instances of operating systems running in different virtual machines are isolated from each other and (in principle) cannot harm each other, nor learn anything about each other. Virtualization has also been applied to mobile platforms to provide isolation for security-critical applications. L4Android [34] uses virtualization to run security-critical applications for smartphones in a separate, iso-

lated virtual machine so that it cannot be tampered with by other potentially malicious applications running in a different virtual machine.

5.5 Verifiable Execution

Techniques for trustworthy computation provide guarantees that outsourced computations have been faithfully executed. Such techniques enable clients to verify that computations are correct, at a fraction of the cost of executing the computations themselves, thus allowing them to benefit from the gains of outsourcing the computation. Some of the techniques that have been used are fully homomorphic encryption [24], garbled circuit evaluation [45], and probabilistically checkable proofs (PCP) [38]. Hence, the key guarantees that current techniques for verifiable computation have provided are that computations as submitted by clients have indeed been carried out faithfully, and that the returned results are not falsified. These guarantees target execution of code on untrusted servers. However, these techniques do not provide any guarantees about the system-level interactions of the submitted code on the servers. Orthogonal techniques are needed for providing isolation guarantees on the execution of untrusted code. In addition, current techniques for verifiable computation are limited to integer operations and loops with finite number of iterations that can be unrolled, and they do not support completely arbitrary computation, particularly floating-point operations and system operations.

6 Future Work and Conclusion

In this paper, we have described our proposal of mobile edge-clouds [15], comprising mobile devices such as smartphones and tablets as their compute and storage nodes for distributed, *in situ* processing of the rich sensor and contextually-enhanced data stored on each user's device, and we have proposed and discussed the security and privacy challenges facing such mobile edge-clouds. We have identified the different functional goals of such mobile edge-clouds, and the security and privacy requirements of these goals of remote data access, remote computation, context-aware computing, and underlying communications. We also proposed a threat model for mobile edge-clouds, and we proposed security properties desirable for mobile edge-clouds. Finally, we provided a review of contemporary work in related areas in security and privacy in security for mobile devices, cloud-computing systems, and techniques for secure and verifiable execution.

In future, we plan to design implement the security and privacy safeguards necessary for mobile edge-clouds. The key challenges which we believe we must overcome are in the inherent limitations of mobile platforms, such as limited battery power, storage, and processing capability, as well as the novel trust model of having mutually untrusted nodes in a distributed computation setting, with no realistic opportunities for initializing or bootstrapping trust in the mobile edge-cloud (e.g. pre-set encryption keys) due to the transient nature of strangers cooperating in an mobile edge-cloud.

To conclude, this paper represents a first step in implementing the security and privacy safeguards necessary for enabling an mobile edge-cloud. These safeguards would enable mobile edge-clouds by assuring users of such edge-clouds that their mobile devices, as well as their personal, potentially privacy-sensitive data on these devices, would be safe from harm and safe from undesired exposure. We plan to concretely explore appropriate techniques for mobile devices to ensure the security and privacy of data and execution on provider nodes in mobile edge-clouds.

References

[1] Android Nears 80% Market Share In Global Smartphone Shipments, As iOS And BlackBerry Share Slides, Per IDC. http://tcrn.ch/1bccUj4.

[2] Apache Hadoop. http://hadoop.apache.org/.

[3] Content-Based Mobile Edge Networking (CBMEN). http://www.darpa.mil/Our_Work/STO/Programs/Content-Based_Mobile_Edge_Networking_(CBMEN).aspx.

[4] E. Aguiar, Y. Zhang, and M. Blanton. *High Performance Cloud Auditing and Applications*, chapter An Overview of Issues and Recent Developments in Cloud Computing and Storage Security. Springer, 2013.

[5] A. Armando, G. Costa, and A. Merlo. Formal Modeling and Reasoning about the Android Security Framework. In *International Symposium on Trustworthy Global Computing (TGC)*, 2012.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[7] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A. Sadeghi, and B. Shastry. Practical and Lightweight Domain Isolation on Android. In *ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)*, 2011.

[8] L. Catuogno, A¿ Dmitrienko, K. Eriksson, D. Kuhlmann, G. Ramunno, A. Sadeghi, S. Schulz, M. Schunter, M. Winandy, and J. Zhan. Trusted Virtual Domains Design, Implementation and Lessons Learned. In *International Conference on Trusted Systems (INTRUST)*, 2009.

[9] A. Chaudhuri. Language-Based Security on Android. In *ACM SIGPLAN Fourth Workshop on Programming Languages and Analysis for Security (PLAS)*, 2009.

[10] E. Chin, A. Felt, K. Greenwood, and D. Wagner. Analyzing Inter-Application Communication in Android. In *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2011.

[11] M. Conti, V. Nguyen, and B. Crispo. CRePE: Context-Related Policy Enforcement for Android. In *Information Security Conference (ISC)*, 2010.

[12] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *USENIX Symposium on Operating System Design and Implementation (OSDI)*, Oct 2004.

[13] D. Diaz-Sanchez, A. Lopez, F. Almenares, and R. Sanchez. Flexible Computing for Personal Electronic Devices. In *IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2013.

[14] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, and V. Tuulos. Misco: a MapReduce framework for mobile systems. In *ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA)*, Jun 2010.

[15] U. Drolia, R. Gandhi, and P. Narasimhan. Enabling Edge-clouds. Technical Report CMU-PDL-13-114, Carnegie Mellon University Parallel Data Laboratory, Oct 2013.

[16] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, and P. Narasimhan. Motivating Mobile Edge Clouds. In *IEEE International Conference on Ubiquitous Intelligence and Computing (UIC)*, Dec 2013.

[17] C. Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming (ICALP)*, Jul 2006.

[18] W. Enck, P. Gilbert, B. Chun, L. Cox, J. Jung, P. McDaniel, and A. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *USENIX Symposium on Operating System Design and Implementation (OSDI)*, 2010.

[19] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A Study of Android Application Security. In *USENIX Security Symposium*, 2011.

[20] Y. Falcone, S. Currea, and M. Jaber. Runtime Verification and Enforcement for Android Applications with RV-Droid. In *International Conference on Runtime Verification*, 2012.

[21] A. Felt, H. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission Re-Delegation: Attacks and Defenses. In *USENIX Security Symposium*, 2011.

[22] E. Fragkaki, L. Bauer, L. Jia, and D. Swasey. Modeling and enhancing Android's permission system. In *European Symposium on Research in Computer Security (ESORICS)*, 2012.

[23] A. Fuchs, A. Chaudhuri, and J. Foster. SCanDroid: Automated Security Certification of Android Applications. In *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[24] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC)*, 2009.

[25] P. Gilbert, B. Chun, L. Cox, and J. Jung. Vision: Automated Security Validation of Mobile Apps at App Markets. In *International Workshop on Mobile Cloud Computing and Services (MCS)*, 2011.

[26] M. Grace, Y. Zhou, Z. Wang, and X. Jiang. Systematic Detection of Capability Leaks in Stock Android Smartphones. In *Network and Distributed System Security Symposium (NDSS)*, 2012.

[27] J. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn, and R. Caceres. Trusted Virtual Domains: Toward secure distributed services. In *Workshop on Hot Topics in Dependable Systems (HotDep)*, 2005.

[28] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: practical accountability for distributed systems. In *ACM Symposium on Operating Systems Principles (SOSP)*, Dec 2007.

[29] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These Arent the Droids Youre Looking For: Retrofitting Android to Protect Data from Imperious Applications. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[30] S. Hussein, P. Meredith, and G. Rosu. Security-Policy Monitoring and Enforcement with JavaMOP. In *ACM SIGPLAN Fourth Workshop on Programming Languages and Analysis for Security (PLAS)*, 2012.

[31] S. Jana and V. Shmatikov. EVE: Verifying Correct Execution of Cloud-Hosted Web Applications. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, Jun 2011.

[32] J. Jeon, K. Micinski, and J. Foster. SymDroid: Symbolic Execution for Dalvik Bytecode. In *Submitted to Science of Computer Programming*, 2012.

[33] A. Juels and B. Kaliski. PORs: Proofs of Retrievability for Large Files. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[34] M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter. L4Android: A Generic Operating System Framework for Secure Smartphones. In *ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)*, 2011.

[35] E. Marinelli. Hyrax: Cloud Computing on mobile devices using MapReduce. Technical Report CMU-CS-09-164, Carnegie Mellon University, School of Computer Science, Sep 2009.

[36] M. Nauman, S. Khan, and X. Zhang. Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints. In *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*, 2010.

[37] I. Roy, S. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and Privacy for MapReduce. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Apr 2010.

[38] S. Setty, R. McPherson, A. Blumberg, and M. Walfish. Making argument systems for outsourced computation practical (sometimes). In *Network and Distributed System Security Symposium (NDSS)*, 2012.

[39] W. Shin, S. Kiyomoto, K. Fukushima, and T. Tanaka. A Formal Model to Analyze the Permission Authorization and Enforcement in the Android Framework. In *International Symposium on Secure Computing (SecureCom-10)*, 2010.

[40] V. Teo. Hyrax: Crowdsourcing Mobile Devices to Develop Proximity-Based Mobile Clouds. Technical Report CMU-CS-12-131, Carnegie Mellon University, School of Computer Science, Aug 2012.

[41] W. Wei, J. Du, T. Yu, and X. Gu. SecureMR: A Service Integrity Assurance Framework for MapReduce. In *Annual Computer Security Applications Conference (ACSAC)*, 2009.

[42] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos. ProfileDroid: Multi-layer Profiling of Android Applications. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2012.

[43] R. Xu, H. Saidi, and R. Anderson. Aurasium: Practical Policy Enforcement for Android Applications. In *USENIX Security Symposium*, 2012.

[44] L. Yan and H. Yin. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. In *USENIX Security Symposium*, 2012.

[45] A. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (SFCS)*, 1986.

[46] A. Yumerefendi and J. Chase. The Role of Accountability in Dependable Distributed Systems. In *Workshop on Hot Topics in Dependable Systems (HotDep)*, Jun 2005.

[47] K. Zhang, X. Zhou, Y. Chen, X. Wang, and Y. Ruan. Sedic: Privacy-Aware Data-Intensive Computing on Hybrid Clouds. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.