Journal of Computer and Communications, 2017, 5, 1-31
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Survey on Three Components of Mobile Cloud Computing: Offloading, Distribution and Privacy

Anirudh Paranjothi, Mohammad S. Khan, Mais Nijim

Department of Electrical Engineering and Computer Science, Texas A&M University, Kingsville, TX, USA
Email: adhoc.khan@gmail.com

How to cite this paper: Paranjothi, A., Khan, M.S. and Nijim, M. (2017) Survey on Three Components of Mobile Cloud Computing: Offloading, Distribution and Privacy. *Journal of Computer and Communications*, 5, 1-31.
https://doi.org/10.4236/jcc.2017.56001

Received: January 18, 2017
Accepted: April 3, 2017
Published: April 6, 2017

Abstract

Mobile Cloud Computing (MCC) brings rich computational resource to mobile users, network operators, and cloud computing providers. It can be represented in many ways, and the ultimate goal of MCC is to enable execution of rich mobile application with rich user experience. Mobility is one of the main characteristics of MCC environment where user can be able to continue their work regardless of movement. This literature review paper presents the state-of-the-art survey of MCC. Also, we provide the communication architecture of MCC and taxonomy of mobile cloud in which specifically concentrates on offloading, mobile distribution computing, and privacy. Through an extensive literature review, we found that MCC is a technologically beneficial and expedient paradigm for virtual environments in terms of virtual servers in a distributed environment, multi-tenant architecture and data storing in a cloud. We further identified the drawbacks in offloading, mobile distribution computing, privacy of MCC and how this technology can be used in an effective way.

Keywords

Cloud Computing, Mobile Cloud Computing, Offloading, Distribution and Privacy

1. Introduction

Smartphones are becoming popular and its users are increasing rapidly every year. Features of smart phones include touch screen interface, Wi-Fi, high speed processors, GPS, etc. Popularity of smartphones allows developers to develop mobile applications in various domains like sports, games, finance, education, etc. [1]. Still these devices are suffered from issues like limited storage space, limited bandwidth and energy due to development of complex mobile applications. To solve these issues cloud computing techniques were introduced. Some of the popular cloud service providers are Amazon, Windows, Google, etc. Heavy Reading [2] and ABI research [3] suggested that revenue of MCC market $68 billion in 2017.

MCC is the combination of cloud computing, mobile computing and wireless networks. Advantages of MCC over cloud computing are:

1) Flexibility: Due to flexibility, users can access the data using their devices from any part of the world. But the user should have proper internet connectivity.

2) Data availability: Data availability allows the user to access their data at any time. It also provides the facility of multiple users accessing the same data simultaneously.

3) Multiple platforms: MCC also provides support for multiple platforms. It allows users to access their data in cloud irrespective of any platform.

Current cloud computing provides following facilities to the user: 1) Executing operations on cloud, 2) Large storage capacity, 3) Backup, 4) Traffic counting, 5) Ability to choose datacenters. Cloud providers mainly concentrate on areas like throughput, memory, availability of server, storage, etc. Also, they are providing three basic services for MCC:

1) Platform service: Platform as a Service (PaaS) provides hardware and software for the user to create, modify, run their applications. Main advantage of Paas is that it allows user to execute and complete their tasks without having appropriate software or hardware.

2) Application services: Application services are also known as Software as a Service (Saas). It provides software application to the user whenever they need it over the internet. It gained more popularity in software market due to software on demand. The main advantage of using Saas is: 1) Cost savings, 2) Efficiency. It also eliminates the issue of individual user license and thereby reduces the expense of an organization.

3) Context-rich services: Mobile applications are becoming popular and providing context aware services to its users. To support this, MCC providers are providing context rich services to the users. It includes congestion detection, discovering parking space, etc.

Papers [4] [5] [6] [7] have not discussed in detail about various techniques involved in MCC. This paper gives the overall idea about offloading, mobile distribution, and privacy in the cloud. Further this paper gives information about various factors affecting MCC and future of cloud computing environment.

Rest of the paper is organized as follows: Section 2 discusses about current mobile cloud architecture and programming model; Section 3 discusses about Offloading mobile applications in cloud; Section 4 focuses on Mobile distribution computing and cloud; Section 5 discusses about Privacy in cloud and user authentication. Finally, we presented the conclusion and future work in Section 6.

2. Current Mobile Cloud Architecture and Programming Models

Mobile cloud architecture:

Current mobile cloud computing architecture includes following components: 1) Regional Data center (RDC), 2) Wireless core, 3) Base stations. It is represented in Figure 1. MCC architecture allows users to offload their operations on cloud [8]. Example: Global Positioning System [GPS], multiplayer games, etc. [9]. But, it is most suitable for heterogeneous environments [10].

RDC: It is used in home computer systems and its associated elements like storage and telecommunication systems. RDC consists of various security devices, power supplies, environment controls, etc. Cloud data centers are distributed in different locations around the world [11].

Wireless core network: Routing the telephone calls across PST is the main function of wireless core network. Also, it provides various services to users who are connected in a network.

Programming models:

Existing programming models in MCC are: 1) Clone cloud, 2) MAUI, 3) Odessa, 4) Orleans, 5) RESTFUL [9]. These programming models are briefly discussed below. Table 1 illustrates comparison of programming model based on blocking state, cloud state and remote execution unit.

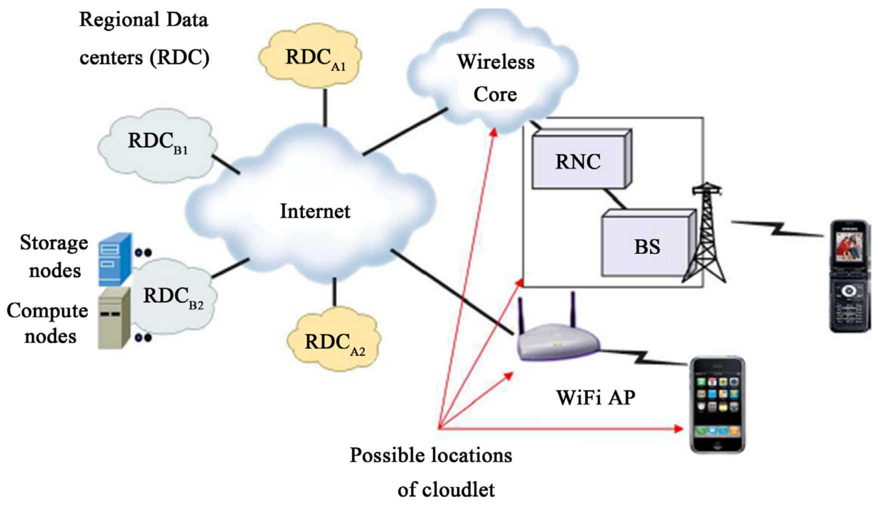

Figure 1. Mobile cloud architecture.

Table 1. Programming model comparison.

Models	Blocking	Cloud state	Remote exec. Unit
Clone Cloud	Yes	Full thread	Thread
MAUI	Yes	Partial	Method
Odessa	Yes	Partial	App task
Orleans	No	Partial	Grains
RESTful	No	No	Cloud task

1) Clone cloud: Clone cloud allows its users to have own copy of their cloud. By providing this facility, user will have full control over their clouds. Clone cloud consists of solver, profiler and analyzer. Solver in clone cloud is responsible for offloading the data on cloud. It will be based on dynamic profiler and static analyzer.

2) MAUI: This programming model is based on Microsoft.NET framework. Profiler in MAUI framework makes remotable decisions. Resource demanding process can be accessed with the help of Remote Procedure Call (RPC). This model is platform and language independent.

3) Odessa: It is a parallel processing framework where developers have to arrange their applications in the form of data flow graph. In graph, vertices are called as stages and edges are called as connectors. In Odessa, connectors give information about data dependency between stages. This programming model is mostly suitable for media applications. Existing applications cannot be accessed in Odessa framework.

4) Orleans: It is the reliable framework for establishing scalable, elastic applications on cloud. Orleans consists of grains, which uses asynchronous messages for communication. Application developer in Orleans mainly concentrates on logic since it provides scalability, reliability and availability during its runtime. It is one of the promising programming models in MCC environment.

5) RESTFUL: This programming model is developed due to media processing applications often requires components for gesture recognition, face recognition, etc. In this model, appropriate functions can be invoked whenever they are needed. It can be done by using http or https protocol.

3. Offloading Mobile Applications in Cloud

In recent days, cloud computing research has been moved towards how to make offloading decisions rather than concentrating on making offloading feasible. Analytical model helps in making these decisions [12] [13]. Offloading and parallelism are the two main factors that impact the system performance. In this section we illustrated the existing frameworks suitable for offloading in mobile application environment.

Offloading: Transferring computations to servers available on the cloud are called offloading [8]. Offloading decisions can be done in two ways 1) Manually by the developers [14] 2) Automatically using tools [15].

3.1. Odessa Framework

Odessa is a lightweight framework, designed for mobile applications [17]. Odessa makes offloading more flexible. Mobile application has three main requirements: 1) Crisp response, 2) Continuous data processing 3) Algorithms should be computed intensive. This framework provides three major contributions: 1) Odessa contributes to offloading and parallelism decisions. 2) Odessa designs a light weight mobile interactive perception applications. 3) It works well across variety of execution environments. The authors used three different applications to measure their system performance. The applications are described below in detail.

Interactive Perception Applications

Face Recognition:

Face recognition application is represented in Figure 2(a). Face detector and classifier are the main components involved in it. Face detector is used to detect faces using OpenCV [18], Haar classifier and face classifier will classify the faces detected by face detector using a dedicated algorithm.

Object and Pose Recognition:

Object and Pose recognition application is represented in Figure 2(b). Four

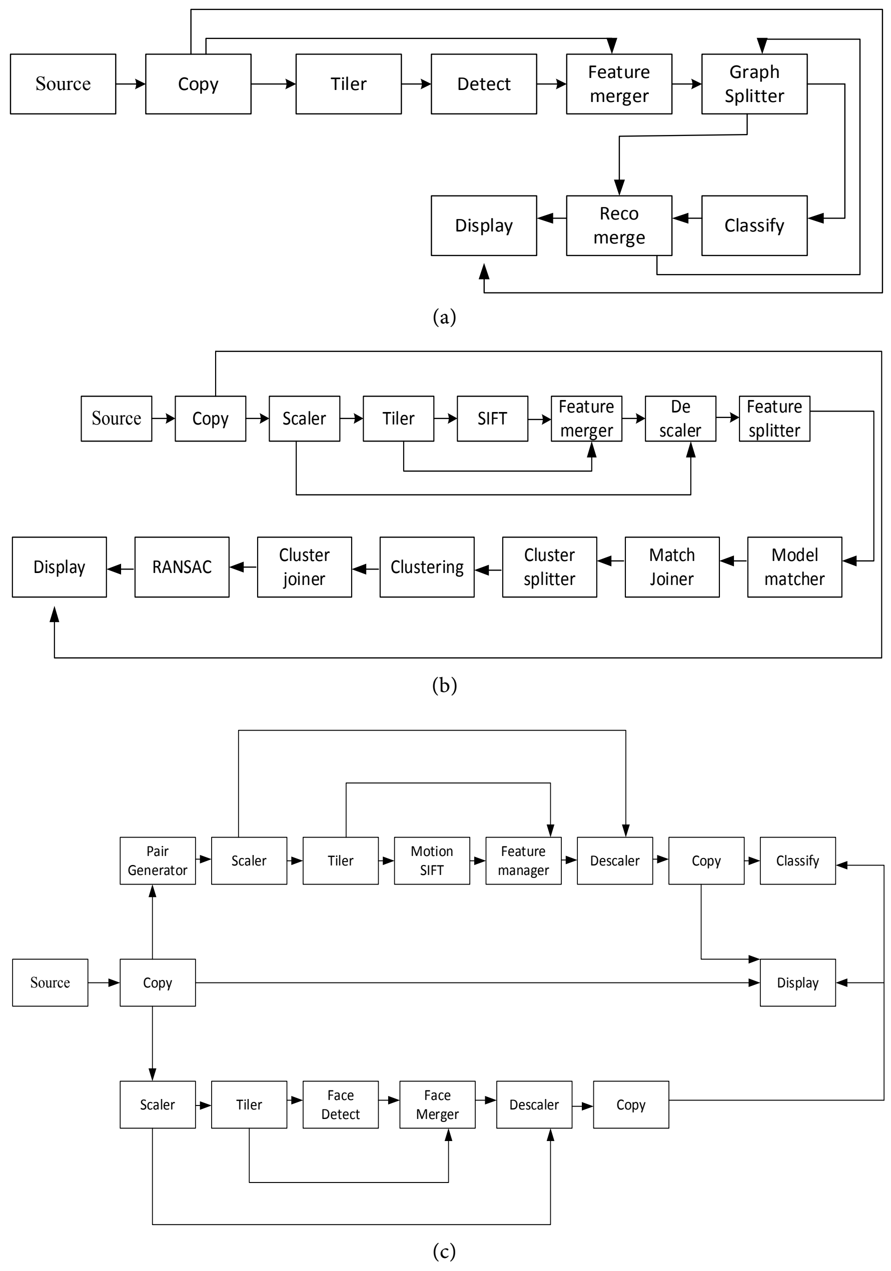

Figure 2. (a) Face recognition; (b) Object recognition; (c) Gesture recognition.

main components are involved in it. First, the image will go through a downscale to extract SIFT features [20]. Second, the extracted SIFT features will be compared with previously constructed 3D models. Object features are clustered by position to isolate different occurrences. RANdom SAmple Consensus (RANSAC) algorithm identifies each occurrence of image with estimated 6D posture.

Recognition of Gesture:

Gesture recognition application is represented in **Figure 2(c)**. Face detection and motion extraction are the major components involved in it. It extracts SIFT features to encode optical flow. These features are then filtered by positions to compare with previously generated histograms. The histograms are used as an input for machine identifies the control gestures.

3.2. Sprout

Sprout is a distributed system used for stream processing [21]. It is used to create and execute parallel processing applications [17]. The main goal of sprout is to support processing of high streaming data. Two important features of sprout are: 1) Automated data transfer, 2) Parallelism support. Also, sprout provides the mechanism of adjusting applications dynamically at run time, changes the degree of parallelism, migrating processing stages between machines.

3.3. Odessa Design

Odessa uses the concepts of offloading, pipelining and data parallelism to improve its performance and accuracy. Odessa has three main goals they are

1) To satisfy the need of mobile applications Odessa should accomplish low make span and high throughput.

2) It should concentrate on input complexity change, device capability and network conditions.

3) It should have low communication and computation overhead.

3.4. ThinkAir Framework

ThinkAir is one of the simplest frameworks in Mobile Cloud Computing (MCC) [22], represented in **Figure 3**. It allows developers to migrate their software to the cloud. Smart phone virtualization and method-level computation offloading are two main concepts adopted by ThinkAir. Offloading in ThinkAir removes the restrictions caused by CloneCloud during the process of offloading [22]. It also on-demand resource allocation for efficient performance of an application. Parallelism is attained by dynamically creating, destroying virtual machines in the cloud.

3.5. ThinkAir Design

ThinkAir framework is designed based on following parameters: 1) Mobile broad band connectivity and speeds are increasing continuously, 2) Capabilities of smart phones are increasing, 3) Cloud computing is becoming more popular, and provide resources to users at low cost. The design goals of ThinkAir are:

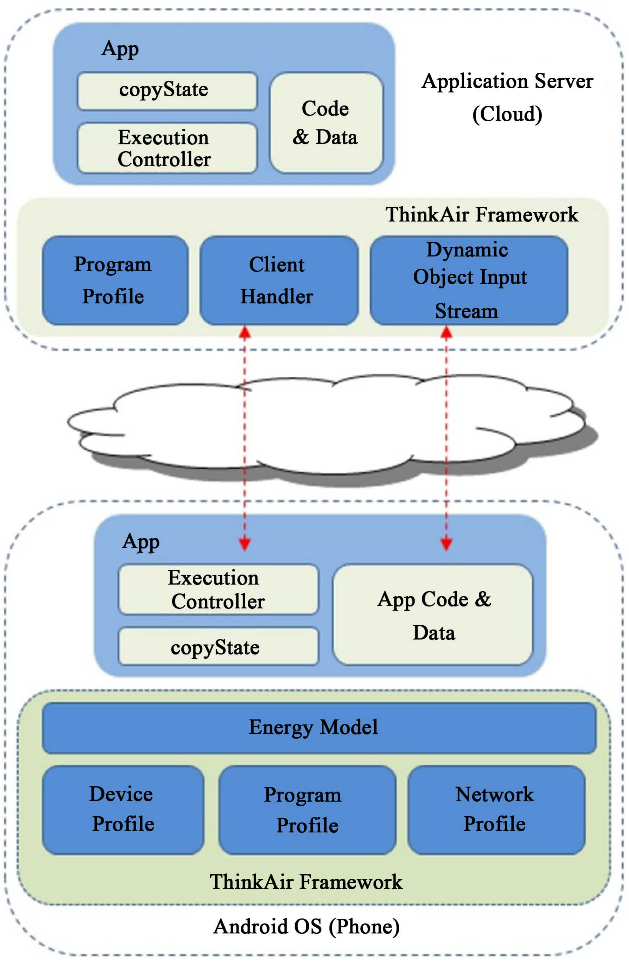

Figure 3. ThinkAir framework.

1) Adaptation

ThinkAir framework easily adapts according to environment and it also avoids interference of correct executing software.

2) Ease of Use

ThinkAir framework provides simple interface for developers to avoid the issue of misusing framework [22] and it increased competition among developers.

3) Performance improvement

ThinkAir framework improves performance and efficiency of mobile devices by binding smartphones to the cloud.

4) Dynamic scaling

ThinkAir provides the feature of calculating computational power dynamically at server side. It also provides parallel executions to improve the performance.

This framework has three major components: 1) Execution Environment, 2) Application servers, 3) Profilers.

3.6. Compilation and Execution

Compilation and Execution section of ThinkAir deals with three major areas: 1) Programmer API, 2) Compiler, 3) Execution controller.

1) Programmer API

ThinkAir contains library with compiler since the developer can access execution environment indirectly. Method considered for offloading is commented with @Remote. ThinkAir code generator generates necessary remote able method with utility functions by taking source file as input and execution controller is used for method invocation and detects the given method is suitable for offloading or not.

2) Compiler

ThinkAir Compiler consists of two parts 1) Remoteable Code Generator, 2) Native Development Kit (NDK). Remoteable code generator used for annotated code translation and Native Development Kit (NDK) used for native code support in cloud.

3) Execution Controller

Execution controller executes remotable methods and makes offloading decisions. Offloading decision depends on the data collected during past execution, the current environment and user's policies. There are four such policies combine with execution time, energy and cost. The four policies are, 1) Execution Time, 2) Energy, 3) Execution time and energy, 4) Execution time, energy, and cost.

4) Execution Flow

Execution Controller starts with profiler to provide data for future invocations and it decides this invocation is suitable for offloading or not. If it is suitable for offloading, then it can be migrated to the cloud using java reflection technique. If it is not suitable for offloading or if connection fails, then execution will back to local execution environment (*i.e.*, smart phone) by eliminating the data collected by the profiler.

4. Mobile Distribution Computing and Cloud

Mobile distribution computing will provide access to widely distributed resources. Distribution computing has the advantages of scalability, fault tolerance, and load balancing. In many situations processing tasks needs to be distributed. However, in distribution computing there is chance of communication failure because it could fail at any time. This section gives the information about various distribution techniques used in the cloud.

4.1. Clone 2 Clone (C2C)

Clone 2 Clone (C2C) [23] provides distributed peer to peer platform for smartphones. Performance measurement of C2C in private and public clouds shows that it is possible to implement C2C in distributed environment with 3 times lesser cellular traffic. In addition, it also saves 99%, 80% and 30% of the battery respectively.

4.2. C2C: Architecture Design

C2C platform needs a mechanism to enable peer to peer networking, to notify others about the presence of others. In C2C, CloneDS (*i.e.*, Clone Directory Services) maps its users to clones and clones to IPs. To establish a connection in C2C platform, request a clone with public IP and key pair. Key pair can be public key or private key. C2C architecture is represented in **Figure 4** and it consists of five basic steps: 1) DS register, 2) DS lookup, 3) C2C Connect, 4) User lookup, 5) User clone connection.

DS register: Clone id, public key, IP address and device id will be send to clone DS from the clone.

DS lookup: Clone A obtains a list signed by CloneDS. The list contains the details of signed clones with their IP address and public keys.

C2C connect: Peer to peer connection will be established with other clones from clone A.

User lookup: User A can always get her clone's IP through a CloneDS lookup.

User clone connection: It establish a connection with user through Public IP.

4.3. C2C and Security

In C2C, communication between the user and clone is secured by the shared symmetric key. This architecture provides trust to the users through CloneDS but some destructive cloud providers use this opportunity by connecting user to harmful ones.

4.4. CloneDoc Framework

CloneDoc and SPORC [24] provide more complexity to the system, but the main advantage of using such system is that it will improve the battery performance by reducing its usage. CloneDoc receives the operations from user's smart phones and keeps the device updated. The clone maintains two states 1) pending queue, 2) committed queue [25]. The clone in C2C delivers operation received from device to server and again send backs the result to appropriate devices. To handle these tasks, it maintains a queue. It should be managed in such a way that

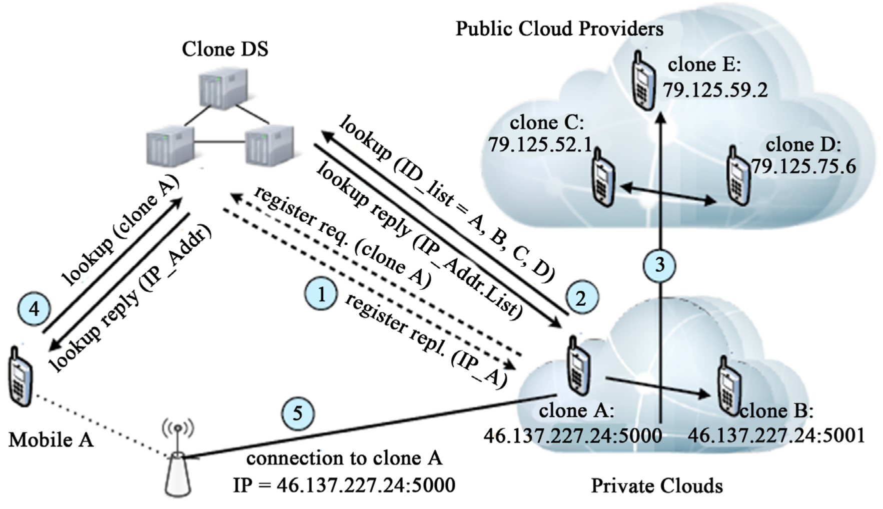

Figure 4. C2C architecture and networking.

delay should be minimum. CloneDoc contains a protocol to solve this problem. It is commonly known as clone user consistency protocol. In CloneDoc, clone is also responsible for detecting server malfunction. Detecting server malfunction contains checking of encryption and decryption operation, sequence numbers, etc.

4.5. Code in the Air

Code In The Air (CITA) [26] is a system which simplifies the rapid development tasking applications. It can be handled by both expert users, non-expert users. In CITA non expert users specify their tasks easily over phone and expert users specify their tasks by writing server side scripts. Current approaches have two major problems: 1) Poor abstraction, 2) Poor programming support. CITA helps developers as well as end users.

4.6. CITA Architecture

CITA architecture has following 3 components, 1) Tasking framework, 2) Activity layer, 3) Push communication and it is represented in **Figure 5**.

Tasking framework: It allows developers to write and compile scripts. Compilation of scripts can be done in server side. CITA also provides JavaScript interface for its developers to manipulate different devices in single program. Backend of CITA deals with device coordination and efficient execution of code on different devices

Activity layer: Activity layer in CITA provides an extensive abstraction to high level activities and also it provides facility of energy efficient recognition of an activity.

Push communication: it improves on the energy and load shortcomings of existing systems.

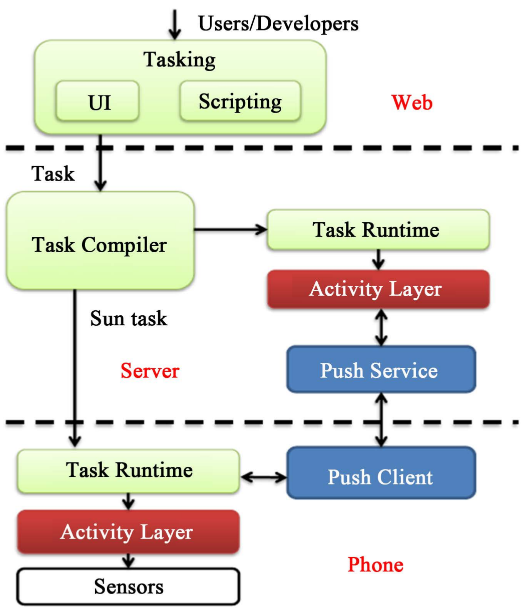

Figure 5. Code in the air architecture.

4.7. CITA Activity Layer

CITA contains an activity layer. The main purpose of activity layer is to express conditions.

4.7.1. Place Hierarchies

CITA uses a built-in location hierarchy, it identifies three types of locations: 1) Room level, Floor level 3) Building level. These hierarchies are having different implementations.

Room level location hierarchy: It is used to match a named location if Wi-Fi signal strength is good.

Floor level location hierarchy: It is used during overlapping of Wi-Fi signals.

Building level location hierarchy: It is used to refer the buildings or large bounding box on a map.

CITA contains two activity detectors: 1) enterPlace, 2) leavePlace. It will be called when a user enters and leaves a location respectively [26].

4.7.2. Activity Composition

CITA allows developers and users to create high level activities using logical predicates [26]. This is one of the advantages of CITA because it provides reusability. Developers and end users can reuse activity modules created by other developers to write their own activities. CITA supports AND, OR, NOT, WITHIN, FOR, NEXT primitives.

4.8. Dial to Deliver Push Service

CITA provides an asynchronous message delivery service to mobile devices from CITA server [26] but it has three major problems.

1) The information to be delivered is very less.

2) TCP connection in mobile devices leads to timeout due to long waiting time.

3) Current push notifications limits notifications of specific types.

CITA uses standard telephone service. It contains the registered users phone numbers. To verify the user, CITA server initiates a voice call on the other end CITA client verify the phone number. If the number matches it wakes up the client. The main disadvantage of using this service is, load on the network will be increased.

4.9. WhereStore

WhereStore provides location based storage for mobile devices. It uses the technique of filtered replication to distribute location history among mobile devices [27]. WhereStore reduces energy consumption by exchanging data in clouds. Location specific applications are the one differentiates computer from mobile phones [28]. The following applications are benefitted from WhereStore framework: 1) Web applications, 2) Media player, 3) Live traffic and sensing applications.

4.10. Where Store Background

WhereStore is designed using two techniques: 1) location prediction, 2 Replication system [27].

4.10.1. Replication

Replication system uses collection as its major technique. It is used to maintain data synchronization between peers. Collection in a replication system has user data and meta data. It will be represented as separate items. In a filtered system, filter identifies the subset where the data is stored exactly. Consider the example of separating the JPEG images according to geographical region. It can be done by identify the images according to the geotag attached in it. There are two major goals involved in filtered replication system:

1) Each clone stores exactly matching item in its filter.
2) In each clone version of items should be same.

4.10.2. Predicting Location

GPS in mobile devices provides location based services to the user. Location based services provides the advantages of tracking the user [29] [30]. The main idea of using location prediction in WhereStore is to predict the future location of the user by matching past location history of the user with present location [31].

4.11. WhereStore System Framework

WhereStore provides dynamic data storage for its user's. It will be based on two parameters: 1) past location of the user, 2) present location of the user. The conceptual view of WhereStore is represented in Figure 6. WhereStore provides complete control to its user. Here, data will be grouped according to the graphical regions where the is likely to stay. It ensures availability of data in user's current location. WhereStore provides complete transparent mechanism for data placement when compared to other frameworks.

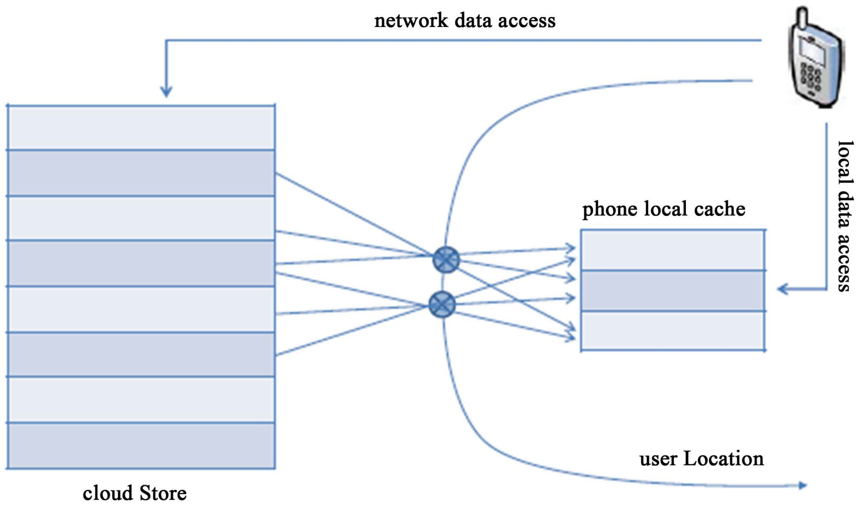

Figure 6. WhereStore conceptual view.

WhereStore will be located top of the replication system and location service. In which, replication layer consists of various collections. Each application in WhereStore has separate collections. The replication layer creates clones (*i.e.*, replicas) for mobile devices and cloud. Each clone has its own storage capacity and filter. Storage capacity of each clone specifies the maximum number of bytes. WhereStore semantics are same as cache (*i.e.*,) the cloud can be accessed only when the item is not available in a local environment. The filter located in each clone will be adjusted according to the current location of user.

4.11.1. Types of Data

WhereStore acts on groups, regions and items. Each item can be identified using its key and priority associated to it. Data can be divided into several items. Groups are set of items and regions gives information about different geographical area. WhereStore has a separate interface to create application, maintain regions and groups. According to the geographical areas regions will be created and each region will be associated with multiple groups.

4.11.2. Filters

Filters in WhereStore specify the items stored in a given location. It has set of filters each possible future location. Future locations will be identified using current location based on the location prediction system. Consider (l_1, l_2, l_3, … l_n) be the future location with probability p_i. WhereStore create new filters (f_i) for each possible future location. When the devices location changes new set of filters which is computed and updated recently will be passed into the replication. Each replication system has its own probability p_i and maximum storage capacity. The rank of items will be based on occurrences. If a particular item is located in more filters, rank will be high.

4.11.3. Cloud Synchronization

Data exchange is performed in replication platform using synchronization established between cloud and smartphones. It provides the advantage to the smartphone by having only items which matches exactly with filter. Smartphones suffers due to limited storage capacity. To utilize the space in a proper manner, smartphones filter will be evaluated in a cloud. Each filter will be associated with a cloud calculates the set of items matches with filter and rank will be provided to each item. Storage capacity of each item calculated according to the rank.

4.12. Implementation of WhereStore

WhereStore implementation is based on clientserver architecture. Here, cloud act as server and user's mobile device act as client. The client has two major components 1) location, 2) Replication. Location specifies the information about future smartphone location and contains information of local cache memory on cloud. Architecture of WhereStore is represented in Figure 7. Whenever application interacts with WhereStore configuration file needs to be provided. Configuration file specifies replication at a particular location. Filters will be updated

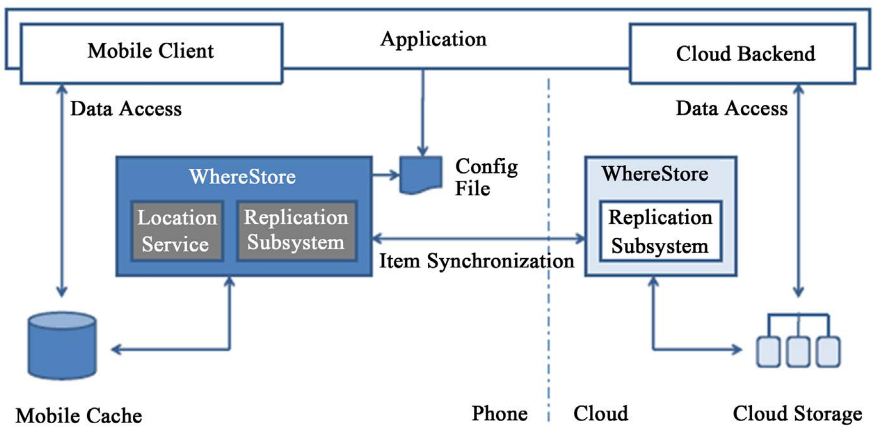

Figure 7. Architecture of WhereStore.

based on input given by the configuration file. Later the updated filter will be used by the replication system.

4.12.1. Mobile Phone Data Access

WhereStore use existing data storage for accessing data. There are numerous existing applications stores the data in their own way. WhereStores uses the concept of Cimbiosys as replication system [32]. It uses callback mechanism for accessing the data and implemented whenever needed. Cimbiosys determines the data to be broadcasted during the synchronization process. Wherestore is responsible for creating metadata whenever new item added into it.

4.12.2. Synchronization of Cache

In WhereStore, Cimbiosys synchronization exchange messages based on a technique called pull style exchange. It will be done in one way. Here, target clone will establish a synchronization with source clone. The connection will be established by sending a request message. Once the connection gets established, source clone starts checking any of its tem is are not admitted by target clone. If any of such items exist, source clone will return the corresponding item to target clone. The current Cimbiosys model can be expressed in two ways: 1) modification of sync request from mobile device to cloud, 2) modify the filter based on Cimbiosys.

4.12.3. Location Prediction

WhereStore uses location prediction technique to predict its user's possible future location. It can be achieved by using StarTrack framework [33]. In StarTrack framework, location of a user will be captured periodically using smartphones. The captured location will be forwarded to cloud where StarTrack servers will be located. StarTrack server will convert the location it is received from smartphones into tracks. It provides a API (Application Program Interface) to perform operations on tracks. In order to identify where the user will be located in a future, StarTrack uses a technique called place transition graph. This technique will be created based on the tracks generated by StarTrack framework. It also has the detail of places that are visited by the users frequently. Latitude and Longitude pair will be used to create this FrequentPlaces. It will be created based on the tracks. Place transition graph will be constructed based on FrequentPlaces. Usually it will be represented in the form adjacency matrix. First, all the element in the matrix will be initialized as zero. Then, set the corresponding value of each item by combining start and end points. Finally, normalize the value in each row according to the probability. This normalization can be done by adding the trip frequencies of each row and divide each frequency with sum of row.

4.13. Virtual Machine Synchronization (VMsync)

The utility of the mobile device will increase if the user wants to switch from one device to another device. For example, user is able to continue the operation in the second mobile device absolutely where the user left in first mobile device without any delay. VMsync [34] used to synchronize virtual machines (VMs). This synchronization will take place among mobile devices. In device switching, VMs encapsulates computation state and data for a complete operating system and applications associated with it. In VMs application state also getting synchronized along with mobile devices. System level VMs are used in now days to provide improved security and manageability. In order to reduce the delay and make image consistent, VMsync is used. It will transfer the changes made in active VM to other mobile devices. The most important component in VMsync architecture is known as daemon. The main purpose of daemon is to monitor the memory and filesystem of VM. If any changes made, it will report the changes to the server. Server will be located on the cloud, and it will send the changes to devices.

4.14. Preliminary Design of VMsync

The main function of VMsync is to handle VM images across various mobile devices and reducing time between switching devices. It uses method called Switch Penalty to perform device migration. The disadvantage of using this method is, data transfer cost is high. VMsync architecture is represented in **Figure 8**. It contains multiple hosts and provides following facilities to the user: 1) Virtualization support, 2) Resource rich server, 3) Synchronization between devices. Initially, VMsync had only one active device. This device is used to update server regarding memory, file system changes over a periodic time. It can be done only when the device is active. This process is known as checkpoint.

VMs other than active VM are known as standby VM. These VMs are getting updated with the help of synchronization server periodically. But, during the updating process device should be connected to network. Synchronization daemon used to monitor this process. VMs should be designed in a way that it can balance data transfer and computational overhead. Modern mobile operating systems like Windows, Android, iOS, provides support for different hardware manufactured by various companies. This functionality can provide the facility adapting changes in hardware during runtime in future.

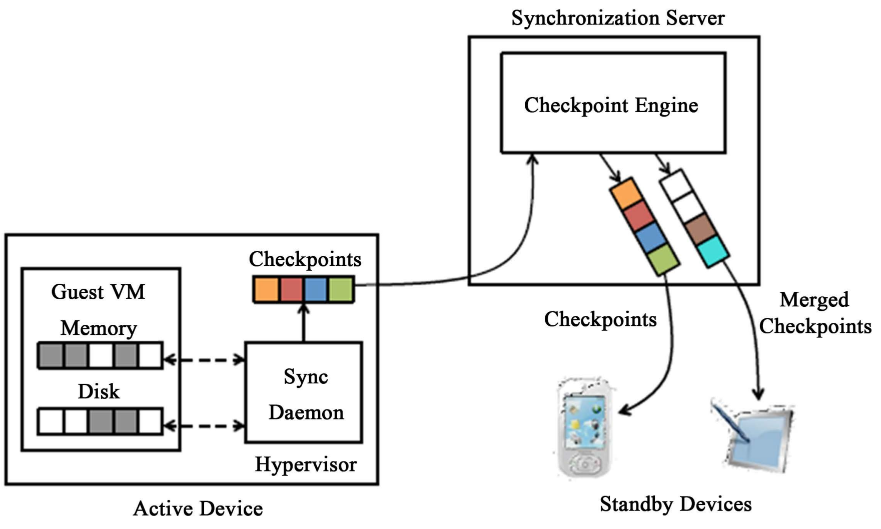

Figure 8. VMsync architecture.

4.15. Wireless Mesh Networks (WMN)

Wireless Mesh Networks (WMN) provides a low cost, next generation wireless networking, and also it provides a high speed internet access. Wireless Mesh Network supports wide range of mobile applications. Wireless Mesh Networking with Mobile Cloud Computing (WM-MCC) is considered to be a best solution for large scale big data applications [35]. In Wireless Mesh Networks mobile client is connected to a Base Transceiver Station (BTS) and it access the mesh network via mesh router. While mesh routers will be connected to each other and it will communicate with cloud through internet. The cloud service platform in wireless mesh networks provides data query services.

5. Privacy in Cloud

Privacy is a major component in MCC. The user needs to understand the standards and procedures provided by the cloud provider to protect their data from threats. The number of businesses and individuals that are moving their data and performing computation on cloud is increasing. Although the cloud computing provides numerous benefits, security remains as one of the major challenge when data and computation are utilized by untrusted third parties [36]. The following section provides the information about different security approaches used in the cloud to protect user data.

5.1. Secure Outsourcing of Collective Sensing and Systematic Applications to the Cloud (p-Cloud)

Two main approaches were proposed to provide security. i) StreamForce, ii) CloudMine.

Streamforce: It is an access control system for sharing of data over malicious and untrusted clouds. This approach is designed with three goals:

1. To provide support to specify and impose fine grained access control policies.
2. To outsource data to cloud if access control methods are enforced.

3. When handling most expensive computations system will be efficient.

CloudMine: It is an on-demand and cloud-based service with which different data owners achieve secure analysis over their collective data. This approach supports three essential functions: 1) sum, 2) set union and intersection, 3) scalar product. CloudMine attains three security promises,

1. It provides data confidentiality in contrast to colluding, semi-honest data owners and semi-honest clouds.
2. Protection is provided to outputs of joint computation against semi-honest clouds.
3. Data owners can accurately identify if the cloud has been lazy.

5.2. System Model for p-Cloud Approach

Figure 9 represents our system model for deploying collective applications to un-trusted clouds. It includes two entities: 1) client, 2) cloud. On the cloud, a collective task that consists of joint data and computation from different clients is performed. The attacker cloud consists of the un-trusted cloud and clients. There are three levels of un-trustworthiness:

1. Curious but Honest
2. Curious and Lazy
3. Fully Malicious

Curious and lazy model allows the attackers to compromise while operating carrying out outsourced undertakings. Particularly, the cloud attempts to learn sensitive information and does not effectively corrupts the computation, but rather it tries to do as limited as possible while charging the customers for the same. This model is legitimized by the economic incentives to overcharge clients without being distinguished, however, there are three security properties relating to this framework.

1. The outsourced data should be protected for input privacy.
2. The outputs should be protected for output privacy.
3. Three parameters are included in integrity. Namely: a) Correctness, b) Completeness and c) Freshness.

Streamforce approach accomplishes input and output privacy as well as correctness in the curious but honesty model. CloudMine accomplishes similar properties yet in the curious and lazy adversary model.

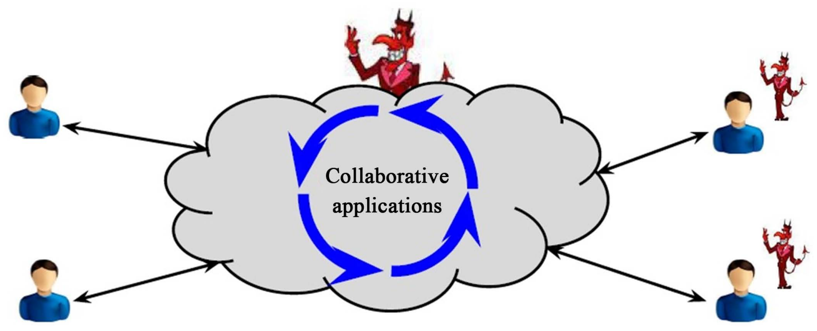

Figure 9. Collaborative applications on the un-trusted cloud.

5.3. Streamforce Approach

Streamforce approach utilizes a fine grained access control system to share data in un-trusted clouds. Implement fine-grained access control in collective applications that are out-sourced to an un-trusted cloud. There are two roles assigned to client. Namely a) Data Users, b) Data Owners. An access agreement P is given to user and the owner is provided with private data x along with related attribute I. When the data attribute fulfills the strategy, *i.e.* P(I) = True, then an authorized client can get access to x. First, the owner sends c = f(x;I) to cloud by using a encoding function f. Later, the cloud changes the encoded data as t = π(c). Lastly, the client assesses a function g(t). In this setting, input privacy infers that the cloud cannot learn x from c. From output privacy and correctness, it is implied that the access control methodology is secure, that is the unauthorized access is not permitted: g(t) = x ↔P(I) = True.

Figure 10 illustrates the design space for access control implementation on a cloud domain. It is described into three measurement strategy. Namely: a) fine-graininess, b) cloud reliability and c) cloud/customer work proportion. Trusted cloud can accomplish best fine-graininess. It supports an extensive variety of approaches and accomplishes best work proportion. Streamforce is particularly intended for stream data. It is outlined with three goals:

1) It supports specification and enforcement of fine-grained access control policies.

2) Access control policies are enforced when data is outsourced to the cloud.

3) System is efficient when handling most expensive computations.

Streamforce security depends on three main encryption strategies. Namely: a) Deterministic ε_d, b) proxy attribute-based ε_p and c) sliding window ε_w.

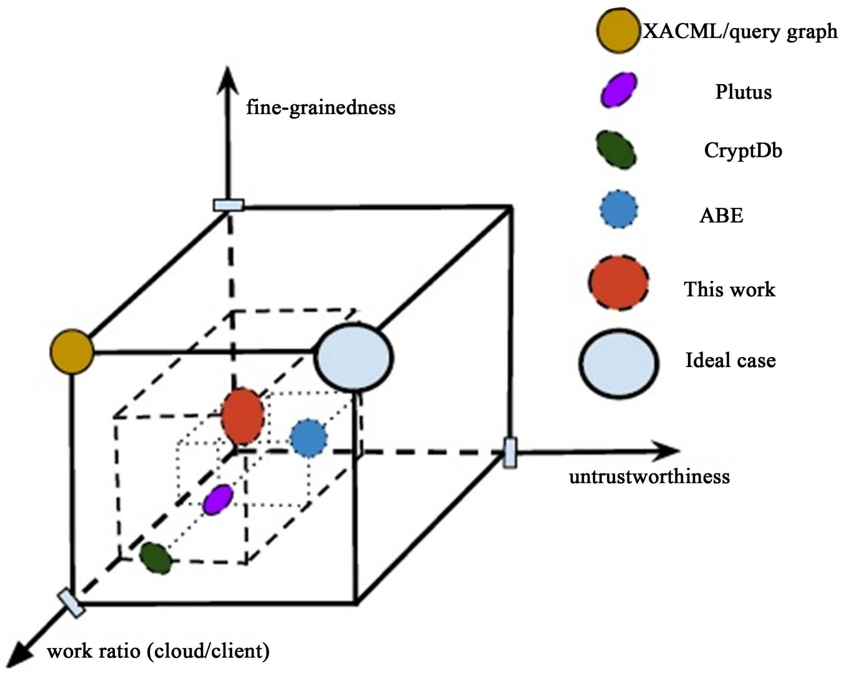

Figure 10. State of the art in outsourced access control.

Deterministic Encryption technique (DET): It is a private-key strategy that is semantically secure while encoding multiple plaintexts. ε_d = (Gen, EncDec), ε_d·Enc(m) = ε_d. $Enc(m') \leftrightarrow m = m'$. The Proxy Attribute Based Encryption technique (PABE) broadens the idea of Key Policy Attribute Based Encryption (KP-ABE). ε_p = (Gen, KeyGen, Enc; Trans, Dec) [37]. Specifically, a master key MK is generated by Gen(.), a transformation key TK is generated by KeyGen (MK, P) and a predicate P is given by decryption key SK. By utilizing the attributes A, Enc (m, A) encrypts m. Trans (TK; CT) partly decodes the cipher-text, which is later decrypted by Dec(SK,CT').

Decryptions and transformation are effective if P(A) = True. Streamforce utilize the strategy provided in [38]. The Sliding Window Encryption technique (WE) permits a client to decrypt just the aggregate of window of ciphertexts but not the individual ciphertexts, ε_w = (Gen, Enc, Dec). Assume that p(M, ws)[i] and s(M, ws)[i] are the product and sum of i^{th} window sliding windows upon a sequence M. The general public parameters and the private keys are created by Gen(k). M is encrypted by Enc (M = (m_0,m_1, ….,m_{n-1}),W) by utilizing a set of window sizes W, whose outcome is CT = (c_0, c_1, c_2,…). CT is decrypted by Dec (ws,CT, SK_{ws}) for the window size ws by utilizing the private key SK_{ws}. s(M; ws) [i] for all I is the outcome, which is the aggregate of the sliding window.

Secure query administrator

Encryption is determined as a strategy that is used to secure data confidentiality in contrast to the cloud and unauthorized client access. However, straightly presenting encryption details to system entities is not considered as a perfect reflection for access control. Rather, Streamforce models and authorizes access control strategies by means of an arrangement of secure inquiry administrators like: 1) secure Map, 2) Filter, 3) Join and 4) Aggregate.

Evaluation

Execute a model of Streamforce over Epser (An open source stream processing engine proficient of handling millions of data items every second). Make a benchmark dataset similar to stock market data and containing one million tuples that belong to 100 streams[7]. Throughput and latency of Streamforce are examined through the experiments conducted on Amazon EC2 with 6 multiple policies (T1-T6). The throughputs for various strategies on a single cloud server are demonstrated in **Figure 11**. The maximum throughput is observed for simple strategy using Map operator, which is 250 tuples/sec. This analyzes ineffectively against Esper's execution on plaintext data. In addition, experiment is also carried with different cloud servers and the outcomes demonstrated scalability for both latency and throughput. We shared the workload in- two ways when more cloud servers are added. They are: a) Simple: based on stream, b) Balanced: based on computation load.

5.4. Practical Confidentiality in Preserving Big Data Analysis

Cloud Computing provides support to Big Data Analysis [39] via data flow languages known as Pig Latin [40]. It is of great value to manage sensitive data only

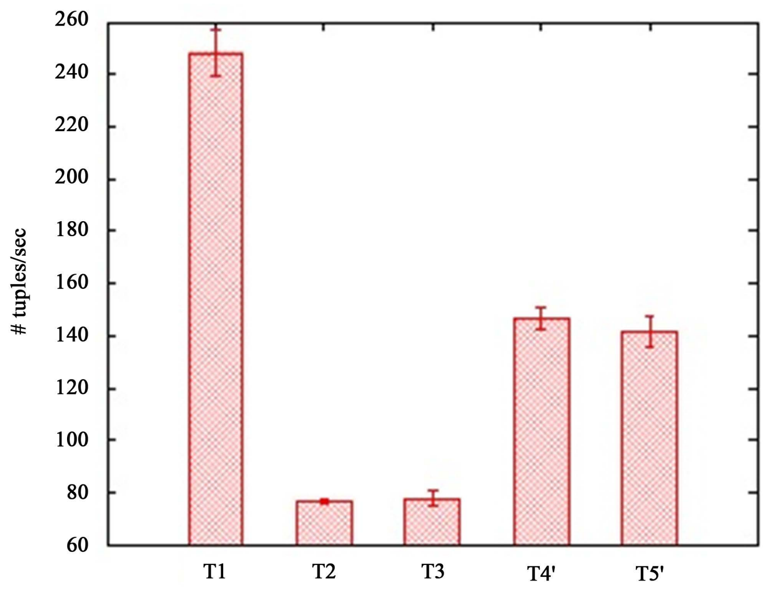

Figure 11. Throughput on single server.

in an encrypted from in the cloud and to perform reasonable data analysis. Crypsis, is a runtime system for Pig Latin which allows corresponding scripts to be executed efficiently by utilizing cloud resources however without exposing input data in the same form. Crypsis can broaden the scope of encryption empowered big data analysis depending on the following point of view:

i) Perspective of Extended Program

Multiple opportunities to operate in encrypted mode are identified by Crypsis by evaluating entire data flow programs.

ii) Perspective of Extended System

Cloud resources can be performed by Crypsis instead of giving up and driving users to run the entire data flow programs on their end. This can be done by considering the chance of performing small computations on user end.

Three main Contributions for preserving big data analysis

1) Without sacrificing confidentiality propose an architecture for executing Pig Latin scripts

2) Outline a novel-field sensitive program to study and transform to Pig Latin scripts that can distinguish operation with effects.

3) Current fundamental assessment results for implementing the solution depending on run time Pig Latin scripts obtained from an open source Apache PigLatin.

5.5. Background: PigLatin

Apache Pig is a data examination platform. It incorporates the Pig runtime framework for high-level data flow language Pig Latin [40]. Pig permits data experts to query big data without the complication of writing MapReduce programs. No fixed schema is required to be operated by Pig. All these properties of Pig Latin and in addition its wide reception made it to be chosen as the data flow language for Crypsis.

Data types and Statements

Pig Latin includes simple types (e.g., **int**, **long**), and complex types (e.g., **bag**, **tuple**, **map**). In addition, field can be a data item like a **tuple**, **bag**, **map**. Pig Latin statements also work with relations; relations are simply a bag of tuples.

Expressions and Operators

Relations are established by loading an input file or by applying relational operators to different relations. Examples of relational operators are **JOIN**, **GROUPBY**, **FOREACH**. etc. Operators in Pig Latin can likewise incorporate casts, arithmetic operators (e.g., +, −, \, *), comparisons, as well as **LOAD** and **STORE** operators.

Functions

Pig Latin incorporates built-in functions (e.g., **ABS**, **COS AVG**) and allows users to define their own user defined functions (UDFs) if needed.

5.6. Architecture and System Overview

Crypsis is having an adversary capable of fully manipulating the cloud infrastructure. The adversary can see encrypted data and Pig Latin scripts that operate on the data and it can control the computation software and control the cloud infrastructure. Crypsis ensures confidentiality in the presence of adversary. Figure 12 illustrates the architecture of Crypsis prototype.

1. Transformation of program

The client presents a source Pig Latin script that works on unencrypted information. This is evaluated by Crypsis in order to find the suitable encryption scheme through which the input data must be encrypted. Calls to Crypsis UDFs which implement operations on encrypted data are used to replace the operators in source script. The constants are supplanted using their encrypted values in order to create an objective script that can be executed on encrypted data.

2. Encryption techniques missing in cloud

The parts of input data that are encrypted previously and stored in cloud are tracked by an encryption service which contains an input data encryption sche-

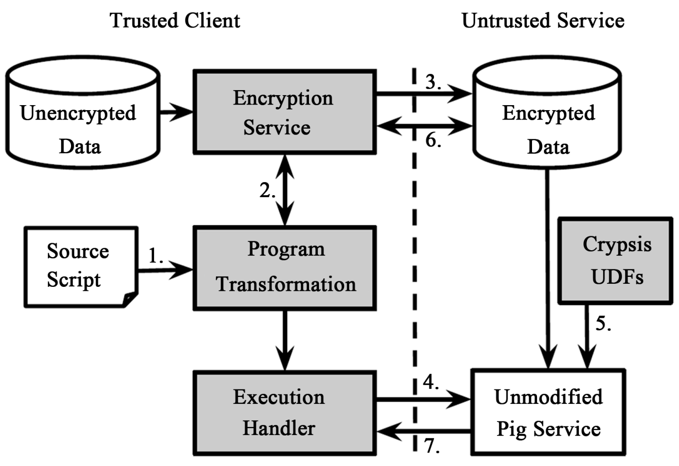

Figure 12. Architecture of Crypsis.

ma. Depending on the input data encryption schema as well as the recommended encryption schemes assumed in previous step, the encryption service determines the encryption schemas lacking in the cloud.

3. Encryption, Sending data to cloud

Various encryption schemes which are enabled through diverse cryptosystems are used by Crypsis.

1) Randomized encryption (RAN) is the main encryption scheme which does not support operators and best secure encryption scheme. One way to execute RAN is by utilizing Blowfish [41] in order to encrypt integer values by exploiting the benefits of its limited 64-bit block size and also by utilizing AES [42] in order to encrypt the remaining.

2) Deterministic encryption (DET) let's fairness comparisons upon encrypted data. First, develop DET utilizing AES and Blowfish permutation block ciphers for estimations of 12 bits and 64 bits respectively. Then, pad minute values properly to coordinate the normal block size. The approach for values greater than 128 bits, check the approach utilized in CryptDB [43]. Later, implement the Order Preserving Encryption (OPE) scheme that permits to arrange correlations utilizing the order preserving symmetric encryption usage from CryptDB. Paillier cryptosystem to implement additive homo-morphic encryption (AHE) which allows additions over encrypted data and ElGamal [44] cryptosystem to implement multiplicative homomorphism encryption (MHE).

4. Execution

When all required encrypted data is loaded in the cloud, the execution handler requests to start executing the job.

5. Crypsis UDFs

Crypsis does not impose any changes to the PigLatin service. Instead, operations on encrypted data are handled by a set of pre-defined UDFs stored in the cloud storage along with the encrypted data.

6. Re-encryption

At the time of target script execution, it is possible that intermediate data are generated after some operations are performed. Encryption scheme of the particular data relies upon the previous operation executed on that data. This situation is handled by Crypsisthrough re-encryption of intermediate data. In particular, this intermediate data is directed to the user where this data can be decrypted without any risk. Later, the decrypted data is encrypted using the specific encryption scheme and then again sent to cloud. After the re-encryption is finished, execution of target script is continued.

7. Results

Results are again sent to user when the job is finished.

5.7. Program and Transformation Analysis

Analysis and process involved in PigLatinCrypsis are represented briefly in Listing 1. It has two input files: input1, input2. Input1 has two arguments and input 2 has one argument. Line 3 in script is used to filter all rows less than or equal to 10. The subsequent lines (*i.e.*, Line 4 and Line 5) perform addition operation on second argument of input1 by grouping first argument. Each group sum will be performed in Line 6 using input 2. Finally, Line 7 displays the result stored in output file. Figure 13 illustrates outline of the different steps and intermediate data structures in program transformation.

Input script analysis

First, Crypsis checks the user submitted source (PigLatin) script for syntax errors and generates a directed, acyclic data flow (DAG) representation of it. The data flow representation uses relations as vertices and the data flow between relations as the edges. Also generate two additional data structures: 1) MET (Map of Expression Trees) 2) SAF (Set of Annotated Fields) [40]. Source script expressions will be stored in MET. In data flow graph, each vertex has keys for all expressions. SAF contains one entry for each field specified for each relation. In input script analysis, AF (Annotated Field) is used to represent individual entries.

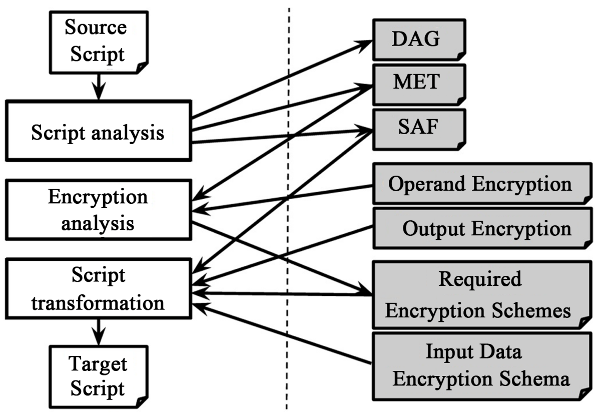

Figure 13. Program Transformation in Crypsis.

```
1 A = LOAD 'input1' AS (a0, a1);
2 B = LOAD 'input2' AS (x0);
3 C = FILTER A BY a0 > 10;
4 D = GROUP C BY a1 ;
5 E = FOREACH D GENERATE group AS b0,
    SUM(C.a0) AS b1;
6 F = JOIN E BY b0, B BY x0;
7 STORE F into 'out';
```

Listing 1. Source pig latin script S_1.

```
A = LOAD 'enc_input1' AS (a0_ope,
  a0_ah, a1_det);
B = LOAD 'enc_input2' AS (x0_det);
C = FILTER A BY OPE_GREATER(a0_ope ,
  '0xD0004D3D841327F2CCE7133ABE1EFC14');
D = GROUP C BY a1_det ;
E = FOREACH D GENERATE group AS b0,
  SUM(B.a0_ah) AS b1;
F = JOIN E BY b0, B BY x0_det;
STORE F into 'out';
```

Listing 2. Transformed pig latin script.

Encryption analysis

The program transformation component identifies the encryption scheme required for each field. It identifies the encryption of each field by observing MET. In script, all operators are already registered with encryption technique. But, some relational operators involved in PigLatin require precise encryption schemes.

Script transformation

After knowing encryption scheme required for each field, decision will be made for which encrypted file to be loaded. Script checks valid encryption technique, if it is not available re-encrypt operation will be initialized. It calls encryption scheme to change the required fields into a specific encryption technique. The transformed PigLatin script is represented in Listing 2.

5.8. Evaluation of Big Data Analysis

Micro-benchmarks

Construct a micro-benchmark that compares unencrypted data with encrypted data based on the size and time requires to execute. It is represented in Table 2. The evaluation of this micro-benchmark was performed on a single machine with two 32 bit CPUs and 3 GB of RAM. While running benchmark, one problem we faced was in PigLatin scripts that projects the value of **map** fields using **chararray**constants as keys (**map**#'key').

PigMix

Run the Apache PigMix2 [45] benchmark to calculate the Crypsis performance. PigMix2 is a set of 17 Pig Latin scripts that tests the latency and scalability of the Pig runtime. The experiment was performed using Amazon EC2 [46].

5.9. Information Leakage

Information leakage [47] gives the information about how the privacy is getting disturbed in mobile environment. Two types of attacks are possible in mobile environment: 1) External attack, 2) Internal attack. Both attacks are used to extract the user information. It is possible to perform these attacks without the user using devices of attacker.

Table 2. Comparing size of data and latency of addition and multiplication operations over plaintext and encrypted data. †^ is the number of operations performed in multiples of 1000. ‡NE denotes no encryption or plaintext data.

	Size (KB)			Time (ms)			
				Add		Multiply	
^[†]	NE[‡]	AHE	MHE	NE	AHE	NE	MHE
2	269	12,071	12,153	32	477	32	2267
4	538	24,142	24,306	63	895	62	4118
6	807	36,212	36,459	92	1314	90	5978
8	1076	48,283	48,611	121	1730	118	7818
10	1345	60,354	60,764	150	2147	147	9658

5.10. Background of Mobile Analytic Service

Background of mobile analytic service concentrates on developers, users, applications, networks, etc. This section deals with app ecosystem and mobile analytics.

5.10.1. App Ecosystem

Application developers use a technique called ad networks to increase the profit of applications. Recent study shows that, top applications available in Android Market (*i.e.*, 52.1% of applications) are enclosed with at least one ad networks. App ecosystem is represented in Figure 14. It illustrates the flow of information among users. Mobile applications are enclosed with analytic library. Main function of analytic library is to collect attributes related user and send it to the servers maintained by analytic companies. The information will be processed and will be given to ad networks like Flurry, Google Ad, etc. to provide appropriate ad for the user.

5.10.2. Mobile Analytics and Tracking

Mobile analytics are used to measure the performance of the applications based on prior knowledge about users, applications, etc. Dashboard performance of flurry is represented in Figure 15. It gives the information about various interests of a user.

5.11. User Profile Extraction

User profile extraction used to extract various information about users. Different services are used to collect distinct information about users (*i.e.*, name, age, etc.). In order to extract information about user, first step is to act on behalf of user. Next step for google is to extract the user profile illustrated by google. For flurry, target information should be send to analytics application which in turn extracts user profile.

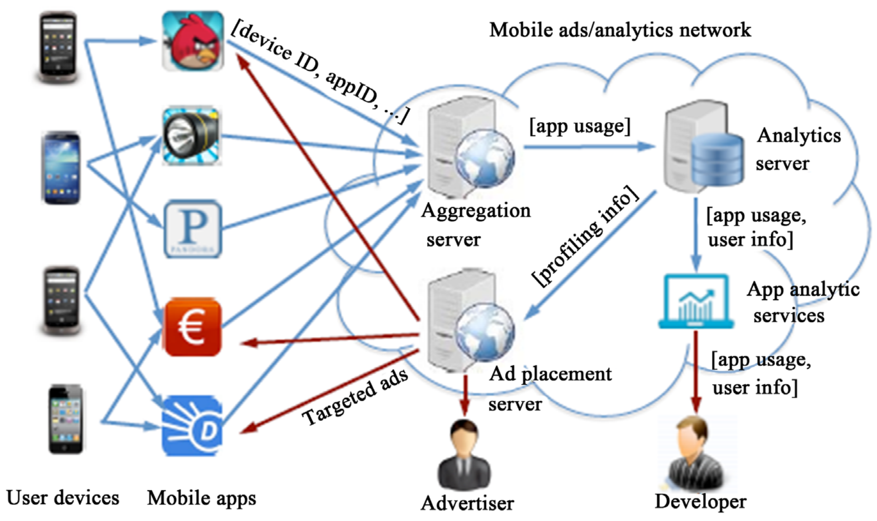

Figure 14. App ecosystem.

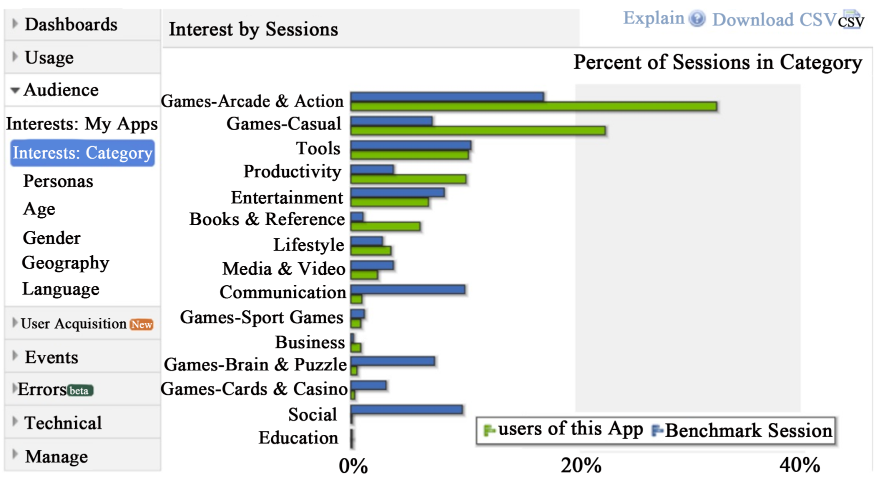

Figure 15. Flurry analytics.

5.11.1. Device id Spoofing

Getting access to device id.

Device id of an android user can be accessed by using two different methods.

1) It can be obtained by grabbing message send by third party.

2) It can be extracted by capturing identifier of a target device.

Device id spoofing

Android users can be easily identified by combining device id with device information. It is possible to device information using the methods described above. Once the information about the device is available, device id spoofing will be done by changing the values in identifying parameters. It is represented in Table 3.

5.11.2. Extracting User Information

Google:

One main advantage of using android is, it allows users to manage their application preferences. This facility extracts user information from Google server. Using this opportunity, anyone can access user profile.

Flurry:

Unlike Google, Flurry will not allow its users to access their information. Profile extraction of Flurry is represented in Figure 16. Here, spoofing will be done by identifying target device id. After identifying the id, it will make the Flurry to generate report message (appID$_x$). All user information can be accessed using this technique. Another method of extracting user profile is by extracting audience report. It can be done by capturing report (P_t) at time t. Flurry also provides an additional feature to distinguish user according to age, name, group, etc.

5.12. Deceiving User Profiles

Second attack targets analytic results. It will attack analytic service and provides inappropriate ads to the user. It will be done by identifying target device and destroy the user information by supplying irrelevant usage reports. This attack will reduce the benefits of ad companies.

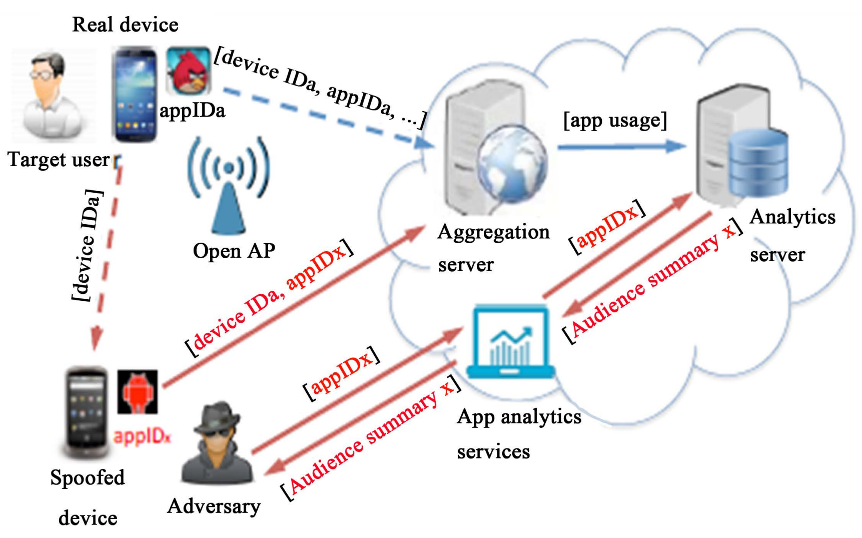

Figure 16. Privacy leakage attack scenario.

Table 3. Android path identifier.

Parameter	File path in Android file system
Android ID	/data/data/com.android.providers.settings/databases/settings.db
ro.build.id	/system/build.prop
ro.build.version.release	
ro.product.brand	
ro.product.name	
ro.product.device	
ro.product.model	

Attacking Technique

This attack will be done in two ways: 1) validating user information, 2) implementing ad influence attack. Both way uses following steps.

Training

In this method, attackers create new user profiles and train them according to different categories. By doing this, ad developers (*i.e.*, Google and Flurry) will be updating user profile according to the reports they received from various categories. The response time will be different for both ad developers. Google takes 6 hours to update user profile and Flurry takes one week to update its user profile.

Collecting Ads

HTTP protocol is used to deliver ads to the users who are using Google or Flurry. Attackers will run tcpdump to extract ads from TCP but it is possible only in Google. In Flurry, redirection methods will be done to get ads.

5.13. User Authentication in MCC

In MCC user authentication is used to validate the user identity. Authentication is used to protect the user against privacy and security issues [48]. It is used to prevent unauthorized access of the user. We have to concentrate security on three major components in MCC. The three components are cloud, wireless communication and mobile device. The efficient algorithm will have the following qualities least possible computing, memory and storage over heads. The purpose of authentication algorithm is to reduce the security threats in mobile devices. Some of the threats commonly occurred in mobile devices are denial of service, loss of device, malfunction of device etc. [48]. The authentication in MCC varies in following scenarios when compared with cloud computing: 1) resource limitations, 2) sensors, 3) high mobility, 4) network heterogeneity.

6. Conclusion and Future Work

In this paper, we have reviewed and explained in detail about offloading, mobile distribution and privacy in MCC. Also, to implement next generation wireless networking with low cost, we explained Wireless Mesh Networking with MCC (WM-MCC). After reviewing various aspects, we found that MCC can be used to provide efficient data storage and processing but the factors affecting MCC are computation power, bandwidth, security and energy. We also found that use of encryption method in offloading and remote execution leads to performance degradation.

New research and expansions programs are required to make offloading decisions more feasible and improve security in the mobile cloud. Furthermore, users want to migrate their data from smartphone to cloud but this migration poses some technical issues. Hence, we need a concrete effort from academia and industry to improve these shortcomings.

References

[1] Zhao, W., Sun, Y. and Dai, L. (2010) Improving Computer Basis Teaching through Mobile Communication and Cloud Computing Technology. *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE*'10).

[2] Heavy Reading Real World Research (2013) The Mobile Cloud Market Outlook to 2017

[3] ABI (2009) Mobile Cloud Computing Subscribers to Total Nearly One Billion by 2014, Tech. Rep., ABI Research.

[4] Mobile Cloud Computing: A Survey, State of Art and Future Directions M. Reza Rahimi · JianRen · Chi Harold Liu · Athanasios V. Vasilakos · NaliniVenkatasubramanian.

[5] Fernando, N., Loke, S.W. and Rahayu, W. (2013) Mobile Cloud Computing: A Survey. *Future Generation Computer Systems*, **29**, 84-106.

[6] Fernando, N., Loke, S.W. and Rahayu, W. (2013) Mobile Cloud Computing: A Survey. *Future Generation Computer Systems*, **29**, 84-106.

[7] Wilbaut, C., Hanafi, S. and Salhi, S. (2008) A Survey of Effective Heuristics and Their Application to a Variety of Knapsack Problems. *IMA Journal of Management Mathematics*, **19**, 227-244. https://doi.org/10.1093/imaman/dpn004

[8] Kchaou, H., Kechaou, Z. and Alimi, A.M. (2015) Towards an Offloading Framework Based on Big Data Analytics in Mobile Cloud Computing Environments. *Procedia Computer Science*, **53**, 292-297. https://doi.org/10.1016/j.procs.2015.07.306

[9] Bahl, P., *et al.* (2012) Advancing the State of Mobile Cloud Computing. *Proceedings of the* 3*rd ACM Workshop on Mobile Cloud Computing and Services*, Low Wood Bay, 25 June 2012, 21-28. https://doi.org/10.1145/2307849.2307856

[10] Verbelen, T., *et al.* (2012) Cloudlets: Bringing the Cloud to the Mobile User. *Proceedings of the* 3*rd ACM Workshop on Mobile Cloud Computing and Services*, ACM. https://doi.org/10.1145/2307849.2307858

[11] Chen, M., Jin, H., Wen, Y. and Leung, V.C.M. (2013) Enabling Technologies for Future Data Center Networking: A Primer. *IEEE Network*, **27**, 8-15. https://doi.org/10.1109/MNET.2013.6574659

[12] Kumar, K. and Lu, Y.-H. (2010) Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? *IEEE Computer*, **43**, 51-56. https://doi.org/10.1109/MC.2010.98

[13] Yang, K., Ou, S. and Chen, H.-H. (2008) On Effective Offloading Services for Resource-Constrained Mobile Devices Running Heavier Mobile Internet Applications. *IEEE Communications Magazine*, **46**, 56-63. https://doi.org/10.1109/MCOM.2008.4427231

[14] Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R. and Bahl, P. (2010) MAUI: Making Smartphones Last Longer with Code Offload. *ACM MobiSys*'10, 49-62. https://doi.org/10.1145/1814433.1814441

[15] Chun, B.-G., Ihm, S., Maniatis, P., Naik, M. and Patti, A. (2011) CloneCloud: Elastic Execution between Mobile Device and Cloud. *ACM EuroSys*'11, 301-314. https://doi.org/10.1145/1966445.1966473

[16] Huang, B.-K., *et al.* (2015) A Cloud-Based Offloading Service for Computation-Intensive Mobile Applications. *IEEE* 21*st International Conference on Embedded and Real-Time Computing Systems and Applications* (*RTCSA*), 19-21 August 2015. https://doi.org/10.1109/rtcsa.2015.17

[17] Ra, M.R., Sheth, A., Mummert, L., Pillai, P., Wetherall, D. and Govindan, R. (2011) Odessa: Enabling Interactive Perception Applications on Mobile Devices. *Proceedings of the* 9*th International Conference on Mobile Systems, Applications, and Services*, Bethesda, 28 June-1 July 2011, 43-56. https://doi.org/10.1145/1999995.2000000

[18] Bradski, G. and Kaehler, A. (2008) Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media.

[19] Romea, A.C., Berenson, D., Srinivasa, S. and Ferguson, D. (2009) Object Recognition and Full Pose Registration from Single Image for Robotic Manipulation. *IEEE International Conference on Robotics and Automation*.

[20] Lowe, D. (2004) Distinctive Image Features from Scale-Invariant Keypoints. *International Journal on Computer Vision* (*IJCV*), **60**, 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[21] Pillai, P.S., Mummert, L.B., Schlosser, S.W., Sukthankar, R. and Helfrich, C.J. (2009) "SLIPstream: Scalable Low-Latency Interactive Perception on Streaming Data. ACM International Workshop on Network and Operating System Support for Digital Audio and Video.

[22] Kosta, S., Aucinas, A., Hui, P., Mortier, R. and Zhang, X. (2012) Thinkair: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading. In INFOCOM, 2012 Proceedings IEEE, 945-953.

[23] Kosta, S., Perta, V.C., Stefa, J., Hui, P. and Mei, A. (2013) Clone2clone (c2c): Peer-to-Peer Networking of Smartphones on the Cloud. In 5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud13).

[24] Feldman, A.J., Zeller, W.P., Freedman, M.J. and Felten, E.W. Sporc: Group Collaboration Using Untrusted Cloud Resources. In Proc. OSDI'10.

[25] Ellis, C.A. and Gibbs, S.J. (1989) Concurrency Control in Groupware Systems. *SIGMOD REC*, **18**, 399-407. https://doi.org/10.1145/66926.66963

[26] Ravindranath, L., Thiagarajan, A., Balakrishnan, H. and Madden, S. (2012) Code in the Air: Simplifying Sensing and Coordination Tasks on Smartphones. *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, p. 4, ACM.

[27] Stuedi, P., Mohomed, I. and Terry, D. (2010) WhereStore: Location-Based Data Storage for Mobile Devices Interacting with the Cloud. *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services*: *Social Networks and Beyond*, p. 1, ACM.

[28] Trestian, I., Ranjan, S., Kuzmanovic, A. and Nucci, A. (2009) Measuring Serendipity: Connecting People, Locations and Interests in a Mobile 3g Network. *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, New York, 267-279. https://doi.org/10.1145/1644893.1644926

[29] Ananthanarayanan, G., Haridasan, M., Mohomed, I., Terry, D. and Thekkath, C.A. (2009) Startrack: A Framework for Enabling Track-Based Applications. *Proceedings of the 7th International Conference on Mobile Systems*, *Applications*, *and Services*, New York, 207-220. https://doi.org/10.1145/1555816.1555838

[30] Mun, M., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., Hansen, M., Howard, E., West, R. and Peir, P.B. (2009) The Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. *MobiSys*'09: *Proceedings of the 7th International Conference on Mobile Systems*, *Applications*, *and Services*, New York, 55-68. https://doi.org/10.1145/1555816.1555823

[31] Krumm, J. and Horvitz, E. (2007) Predestination: Where Do You Want to Go Today? *Computer*, **40**, 105-107.

[32] Ramasubramanian, V., Rodeheer, T.L., Terry, D.B., Walraed-Sullivan, M., Wobber, T., Marshall, C.C. and Vahdat, A. (2009) Cimbiosys: A Platform for Content-Based Partial Replication. *NSDI*'09: *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, Berkeley, 261-276.

[33] Ananthanarayanan, G., Haridasan, M., Mohomed, I., Terry, D. and Thekkath, C.A. Startrack: A Framework for Enabling Track-Based Applications. *MobiSys*'09: *Proceedings of the 7th International Conference on Mobile Systems*, *Applications*, *and Services*, New York, 207-220. https://doi.org/10.1145/1555816.1555838

[34] Bickford, J. and Cáceres, R. (2013) Towards Synchronization of Live Virtual Machines among Mobile Devices. *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, p. 13, ACM. https://doi.org/10.1145/2444776.2444794

[35] Lin, H., *et al.* (2015) A Trustworthy Access Control Model for Mobile Cloud Computing Based on Reputation and Mechanism Design. *Ad Hoc Networks*, **35**, 51-64. https://doi.org/10.1016/j.adhoc.2015.07.007

[36] Dinh, T.T.A. and Datta, A. (2013) Towards Secure Outsourcing of Collaborative Sensing and Analytic Applications to the Cloud-the pCloud Approach. *Proceedings of the First International Workshop on Middleware for Cloud-Enabled Sensing*, p. 2, ACM. https://doi.org/10.1145/2541603.2541606

[37] Goyal, V., Pandey, O., Sahai, A. and Waters, B. (2006) Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In CCS'06.

[38] Green, M., Hohenberger, S. and Waters, B. (2011) Outsourcing the Decryption of Abeciphertexts. In USENIX Security.

[39] Stephen, J.J., Savvides, S., Seidel, R. and Eugster, P. (2014) Practical Confidentiality Preserving Big Data Analysis. *Proceedings of the* 6*th USENIX Conference on Hot Topics in Cloud Computing*, USENIX Association, 10.

[40] Olston, C., Reed, B., Srivastava, U., Kumar, R. and Tomkins, A. (2008) Pig Latin: A Not-So-Foreign Language for Data Processing. In SIGMOD.

[41] Schneier, B. (1994) Description of a New Variable-Length Key, 64- Bit Block Cipher (Blowfish). In Fast Software Encryption. Springer-Verlag, 191-204.

[42] Daemen, J. and Rijmen, V. (2002) The Design of Rijndael: AES—The Advanced Encryption Standard. Springer-Verlag, Berlin, Heidelberg, New York.

[43] Popa, R.A., Redfield, C.M.S., Zeldovich, N. and Balakrishnan, H. (2011) CryptDB: Protecting Confidentiality with Encrypted Query Processing. In SOSP.

[44] Elgamal, T. (1985) A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, **31**, 4.

[45] Ouaknine, K., Carey, M. and Kirkpatrick, S. (2015) The PigMix Benchmark on Pig, MapReduce, and HPCC Systems. *IEEE International Congress on Big Data* (*Big Data Congress*), 27 June-2 July 2015. https://doi.org/10.1109/bigdatacongress.2015.99

[46] Marathe, A., *et al.* (2016) Exploiting Redundancy and Application Scalability for Cost-Effective, Time-Constrained Execution of HPC Applications on Amazon EC2. *IEEE Transactions on Parallel and Distributed Systems*, **27**, 2574-2588. https://doi.org/10.1109/TPDS.2015.2508457

[47] Chen, T., Ullah, I., Kaafar, M.A. and Boreli, R. (2014) Information Leakage through Mobile Analytics Services. *Proceedings of the* 15*th Workshop on Mobile Computing Systems and Applications*, p. 15, ACM. https://doi.org/10.1145/2565585.2565593

[48] Alizadeh, M., *et al.* (2015) Authentication in Mobile Cloud Computing: A Survey. *Journal of Network and Computer Applications*.