An Accountable Framework for Sensing-Oriented Mobile Cloud Computing

Zhifeng Xiao
Penn State Erie, the Behrend College
Erie, PA, 16509 USA
Email: zux2@psu.edu

Yang Xiao
The University of Alabama
Tuscaloosa, AL 35487-0290 USA
Email: yangxiao@ieee.org

Hui Chen
Virginia State University
Petersburg, VA 23806 USA
Emails: huichen@ieee.org

Prof. Yang Xiao is the corresponding author

Abstract

Recent trends in Cloud Computing have further stimulated the popularization of mobile device industry, creating a novel computing paradigm called Mobile Cloud Computing (MCC). MCC takes advantage of the powerful computation and storage capability of cloud servers by offloading heavy computing or storing tasks from mobile devices to cloud servers to keep a thin frontend on the mobile devices. Such benefit is important to MCC leveraging various sensors equipped in modern mobile devices. We explore the sensing capability of MCC and design an application framework that enables a class of exciting mobile applications to be developed in the sensing-oriented MCC environment. A critical issue in such an environment is accountability. We provide a comprehensive analysis of the accountability issues in this new computing context and show how the accountability function is integrated into the application framework.

Keywords: mobile cloud computing, accountability, sensing-oriented application

1 Introduction

Cloud Computing enables cloud vendors to deliver powerful computation and storage as a service to their customers. Meanwhile, Cloud Computing provides a multitenant environment, which allows numerous cloud customers to share platform resources by running different applications simultaneously. Instead of maintaining dedicated computation and storage facility by themselves, customers can significantly reduce the cost of service provisioning through moving their computing and storage facility to the cloud. Recent advances in mobile/portable device industry make it possible to take advantage of the great potential of Cloud Computing with mobile devices. A computing paradigm, called Mobile Cloud Computing (MCC) integrating both mobile devices and Cloud Computing has emerged [2].

MCC extends the current cloud to include numerous mobile devices. With MCC, applications running on mobile devices become thinner since heavy tasks can be moved to cloud servers. Although offloading is the focus of existing works for MCC [14, 15], it is not the end of the story. In addition to the two well-known functional dimensions (i.e., computation and storage), MCC presents a third functional dimension, i.e., sensing [6]. The ability of sensing tremendously increases the ways of data collection and sharing on MCC. There are some other related papers about sensing [36-45].

For example, in a disaster recovery application, when a natural disaster happens, mobile/stationary devices are connected via ad hoc networks or have bi-directional communications with the cloud; people carrying the devices can provide nearby sensing data to the cloud, which could service a rescue center.

This paper is focused on a sensing-oriented MCC whose main function includes computation sensing data gathering, processing, and presentation. Broadly defined, sensing is the capturing of any real world data, ranging from basic environmental parameters such as temperature, humidity, and light intensity, to location and motion data, and to more complex data types such as image, sound, video, etc. Little advancement in mobile sensing applications for MCC has been made until recent technologies of the mobile/portable devices (e.g., smart phones) have revolutionarily changed people's lives. One of the exciting features of these devices is that they come with a suite of sensors. For example, an Apple iPhone 5 has a gyroscope, proximity sensor, compass, accelerometer, ambient light sensor, microphone, and cameras. Leveraging the sensing capability of mobile devices, mobile sensing applications are gaining rapid growth. Some applications [12, 13] have been developed to accomplish specific sensing tasks, while a unified application framework still remains to be developed. To this end, we propose a task-driven application framework for sensing-oriented MCC. Based on the framework, third party developers are able to provide a variety of applications, which will specify the ways to handle sensing tasks. In addition,

we consider accountability as another design priority for our framework due to its significance in dependable systems. Accountability has been a longstanding concern of trustworthy computer systems [3], and it has recently been elevated to a first class design principle for dependable networked systems [1]. Accountability implies that an entity should be held responsible for its own actions or behaviors [4]. We will fully explore the accountability issues in sensing-oriented MCC, and integrate our solution that is based on secure logging and third party auditing to the proposed framework. Therefore, applications based on our framework are able to detect various misbehaviors with undeniable evidence.

The rest of this paper is organized as follows. An overview of cloud computing and MCC is given in Section 2. Accountability issues in MCC are then proposed in Section 3. Section 4 describes the design of an application framework for sensing-oriented MCC. We conclude this paper in the Section 5.

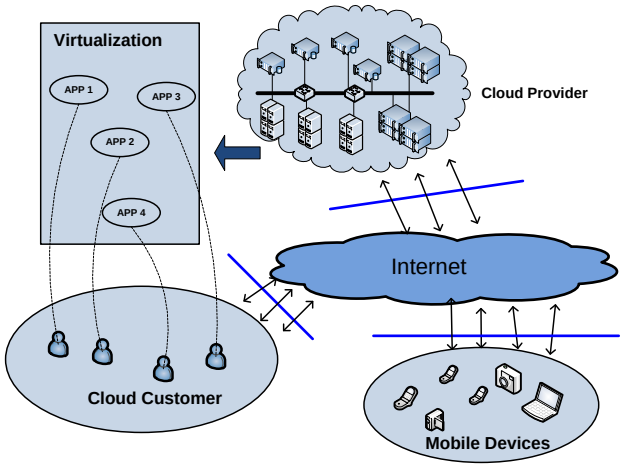

Fig. 1: An overview of MCC

2 Cloud Computing and Mobile Cloud Computing

In this section, we discuss cloud computing in general and mobile cloud computing, and focus on the security aspects.

2.1 Cloud Computing

Recent advances have witnessed the success and popularity of cloud computing. The feature of on-demand provisioning of computation, storage, and bandwidth resources has driven many businesses to utilize cloud computing services. Although the cloud is considered cutting edge technology, it has become critical to the functions of many large companies in diverse business segments.

2.1.1 Cloud Computing Characteristics

Compared to the traditional computing paradigm, cloud computing has five major distinguishing characteristics as follows [16].

- **On-demand self-service** means that cloud customers can obtain computing capabilities on demand.
- **Broad network access** enables customers to access cloud services via any communication mechanisms.
- **Resource pooling** means that the multi-tenant customers can demand physical and virtual resources (such as storage, processing, memory, network bandwidth, and virtual machines) dynamically by pooling.
- **Rapid elasticity** means that cloud services are elastic and can rapidly scale in/out so that their resources appear to be unlimited to the customers
- **Measured service** enables monitoring, controlling, and metering the provided services, as well as reporting.

2.1.2 Example Cloud Providers

Amazon Elastic Compute Cloud (Amazon EC2) [25] provides web cloud computing services, and is easily accessible to developers with simple service interfaces for customers. Customers can control computing resources and programs in a proven computing environment. Amazon EC2 is efficient for starting new instances, scaling capacity, allowing users to pay exact needed capacity, and building failure resilient applications.

Google App Engine [26] is easy to use, maintain, and scale for developers without servers to maintain so that users can run their uploaded applications easily.

Microsoft's Windows Azure platform [27, 28] provides a specific set of cloud services, which support both cloud applications and on-premise applications.

2.1.3 Cloud Security

Security issues have been a long-term concern for cloud computing and many consider this concern as main obstacle of the widespread use of cloud computing [16, 30-33]. Three main challenges for building a secure and trustworthy cloud are [16]:

- **Outsourcing** to the cloud decreases cloud customers' capital expenditure and operational expenditure, with the tradeoff of customers' losing physical control of hardware, software, and data. Therefore, the cloud should offer the customers the

capability of verifying data and computation in terms of confidentiality, integrity, and other security services.

- **Multi-tenancy** (via virtualization of resource allocation and management) allows different customers to save data on the same physical machine. Leveraging multi-tenancy, attackers may launch various attacks such as data breach, flooding attack, etc.
- **Massive data** and **intensive computation** have huge computation or communication overhead, and such overhead poses new challenges to achieve security goals. Therefore, new security requirements are needed.

The above challenges give rise to a set of completely novel vulnerabilities and threats that have never occurred in traditional computing systems. We briefly summarize these new security issues below.

- **Cloud confidentiality** means that data and computation tasks of customers should be confidential from both other customers and the cloud provider. Cloud confidentiality can be violated by exploiting the VM co-residency vulnerability, through which adversaries can launch a cross-VM attack via a side channel to steal sensitive information from VMs that co-reside on the same physical machine [17]. In addition, malicious system admin can exploit the VM co-residency vulnerability to cause data breach [18].
- **Cloud integrity** includes cloud data integrity and cloud computation integrity, and both present unsolved issues. Data integrity indicates that any violations on cloud data (e.g., data missing, modification, or confidential compromising) can be detected. Computation integrity indicates that any distortion on the cloud programs' execution by malware, cloud vendor, or other users can be detected. The main challenge for data integrity is that the tremendous size of cloud data makes classic MAC-based approaches ineffective or inefficient. Researchers have developed new approaches such as Provable Data Possession (PDP) [19] and its variants [20, 21] to protect cloud data integrity. Computation integrity in cloud computing aims to ensure that a machine can verify the correctness of an outsourced computation task to a remote server without running the task locally. Proposed schemes include re-computation, replication, auditing, trusted computing, etc.
- **Cloud availability** implies that cloud services can be consistently delivered. Besides conventional Denial of Service (DoS) attacks, a DoS attack is developed to target cloud servers to saturate the limited network bandwidth in cloud environment [22], and another attack, called Economic Denial of Sustainability (EDoS), can cause denials of availability of publicly accessible hosting web contents [23].
- **Cloud accountability** implies that undeniable evidence is obtained to identify a party being responsible for specific events. Service Level Agreement (SLA) violation, hidden identity of adversaries, dishonest MapReduce, and inaccurate billing of resource consumption are among reported new threats [10, 24]. A few use cases are provided in [5] to address the accountability issue in cloud: 1) Mis-configuration or defectiveness can corrupt customers' data or return incorrect results; 2) Accidental insufficient resource location can degrade the services and cause unsatisfied SLAs; 3) Data can be stolen, the machines can be captured when some software bugs are exploited by using spam or DoS attacks; 4) Loss of data or unavailable data causes the data unavailable. Root causes of data leaking to a competitor vendor and incorrect computation results are difficult to figure out without solid evidences, and therefore the solution becomes to achieve accountability [5]. Each of the threats involves a cloud entity that attempts to misbehave. The overall goal of cloud accountability is to ensure that any violation of cloud security policies will be discovered with provable evidence. Pearson and Charlesworth [29] argue that the following elements are the key to provision accountability in the cloud:
    - **Transparency**. Cloud customers should be adequately informed about how their data and computation tasks are handled in the cloud and that the responsibilities of entities should be clearly identified.
    - **Assurance**. Cloud users should provide assurance to their clients through certain privacy policy, while requiring similar assurances from the cloud vendor through contractual measures and audits.

- **User trust**. Accountability is a premise of user trust. It is crucial for users to understand that why their personal data is requested and processed by another party, or users will become suspicious and then distrust occurs.
- **Responsibility** should be pre-determined via contracts, as information is shared and processed within the cloud, preempts perceptions of regulatory failure, which may impair user trust.

2.2 Mobile Cloud Computing

A big picture of MCC is depicted in Fig. 1. There are several roles in the MCC environment.

- **Cloud provider** is the owner of cloud servers and other hardware infrastructure. Cloud provider offers various combinations of computation, storage, and sensing capability to its customers. Resources on the cloud are well organized and managed through virtualization.
- **Cloud customers** are people who purchase/use certain services from the cloud provider. Traditional cloud providers attract customers by powerful but affordable computation and storage service. In the MCC context, information is not only processed and stored in cloud, it is also sensed, collected, and obtained from the cloud, forming a complete information chain. Therefore, customers will have more abundant choices of mobile applications.
- **Mobile devices** become the main source of data. In addition, MCC enables mobile devices to offload certain computing and storing tasks to the cloud to save energy and resource.

We provide an application scenario of sensing-oriented MCC called Sensing Map to demonstrate how the framework works. In Sensing Map, a digital geographical map is accessible to each cloud customer who is interested in certain environment information within a particular area. To fully explore the area, the customer issues a task to the cloud, which distributes the task to every mobile device currently locating in the area. A selected mobile device can choose to accept or deny the task. The main incentive to accept such a task is based on a principle: the more you give, the more you can obtain. In other words, if one chooses to comply with the task, he/she will be granted higher ability (e.g., ability to request image or even video from other people). With this principle, being selfish would not help a mobile user upgrade his/her ability. If the task is accepted, a mobile device needs to report sensing data to the cloud. After information assembling, aggregation, and post-processing, the cloud delivers the sensing result to the task owner (i.e., customer). For example, a tourist may be interested in one of the scenery spots in his/her destination city; he/she may first find the specific location in Sensing Map, then select the kinds of information (e.g., image and video) interesting to him/her, and then issue a task to the cloud. Upon receiving the new task, the cloud first searches its database to see if there are similar tasks submitted by other customers. If there are, it returns the existing data to the customer; otherwise it disseminates the task to mobile devices that are geographically close to the scenery spot. If a mobile device chooses to accept the task, it will take pictures or videos of the scenery spot, and report the sensing data back to cloud.

The above case indicates the great potential of sensing-oriented MCC. However, it also presents some issues that are worth further studies.

- **Privacy**. MCC enables the cloud to obtain a large amount of information, some of which should be protected from being revealed. For example, the current location of a mobile device should remain secret to the cloud or other.
- **Accountability**. MCC also provide chances for malicious participants to misbehave. Detecting an abnormal event or misbehavior with undeniable evidence is another priority. Accountability is the focus of this paper.
- **Energy saving**. If a mobile device accepts too many tasks, its battery may drain rapidly. For the interest of a mobile device, a trade-off can be sought between energy saving and task management.

3 Accountability in Mobile Cloud Computing

Accountability in MCC is a significant security aspect. Since there are three roles involved in MCC, it is essential to restrict accountability boundaries among different groups to facilitate blame assignment when an anomaly happens. In this paper, we divide the issues of accountability in MCC into three categories according to the groups and boundaries among them:

- Accountability Level 1 (AL #1 for short) defines the trust relationship between cloud

provider and customers. Since customers deploy their software on the cloud, accountability is needed to ensure whether the Service Level Agreement (SLA) is fulfilled. If it is not, evidences should be provided to indicate who the responsible unit is. The problem exists in both general cloud and MCC, and it has been discussed in [5].

- Accountability Level 2 (AL #2) addresses the accountability issues among cloud machines or virtual machines. A data center usually consists of thousands of computers working together to finish tasks. However, some machines may be attacked, compromised, or mis-configured, and the resulting misbehavior usually jeopardizes the tasks running on the cloud. For example, a popular parallel computing paradigm called MapReduce, adopted by major cloud vendors splits large data into multiple blocks to allow a number of working machines to work on them in parallel [7]. However, compromised working machines may be manipulated not to return correct results. Therefore, accountability is needed to identify the malicious workers with undeniable evidence. This problem has been discussed in [8], [10], [24].
- Accountability Level 3 (AL #3): In MCC, a mobile sensing application involves the cloud, customers, and mobile devices. Therefore, we need to ensure that 1) applications on the cloud are accountable, and this means that applications should be committed to what they have done, e.g., issuing a new task or other control messages to mobile devices; 2) sensing devices should be accountable for the sensing data and other messages that they send to data center.

In this paper, we focus on AL #3, because the other two levels are not MCC-specific and have been addressed in general cloud computing environments. We study the accountability issues in MCC from four different angles, which are the four design objectives for our framework as well.

3.1 Message Accountability

Cloud software, cloud customer, and mobile devices should be committed to the messages that they send or receive; also, all of them should be able to defend themselves to the messages that they never send or receive. For the entire MCC system, all messages should be accurately traced back to the responsible unit. There are two properties: 1) **accuracy**, i.e., each message can be accurately traced back to its source; and 2) **completeness**, i.e., each message generator is able to defend itself from false accusation. In MCC, messages can be categorized based on the type of source and destination:

- Customer-to-cloud: customers usually interact with cloud through a web client, which means messages are carried in the HTTP protocol.
- In-cloud: machines in cloud need to exchange messages with each other to collaboratively accomplish tasks.
- Cloud-to-mobile device: messages in this category are unique to MCC. Obviously, although a mobile device can be also a customer, we emphasize its sensing function and the role as a data origin. There are mainly two types of messages in this category:
  - Control message: tasks are issued by the sensing application operator, which might be the cloud customer, who not only run their software in the cloud, but also manage the software to issue commands to the mobile devices. We assume that there are well-defined interface among cloud, cloud customer, and mobile user. Tasks should be made accountable. First, the entity that issues the task command should be able to prove that it has indeed issued this task and it is unable to deny the tasks that it has issued. Second, mobile devices, i.e., the receivers of task commands, should be able to prove that whether they really receive the task commands or not; similarly, they are unable to pretend they receive the task and return some bad data.
  - Sensing data: mobile devices start to sense after they accept tasks from the cloud. Mobile devices should be accountable for the sensing data they captured, i.e., sensing data should be known that where it comes from, and the data source device cannot deny it.

3.2 Behavior Accountability

Mobile devices sense the environment according to the commands that they receive. However, it is possible that some devices do not fulfill the tasks assigned to them. This could happen if some malicious mobile devices 1) totally ignore the task commands, or 2) provide fake data. Therefore, it is

critical for the system to detect the devices that do not comply with the application regulation or protocol.

3.3 Temporal Accountability

Temporal Accountability (T-Accountability) has been previously studied [11]. In MCC, some service is time-critical. First, SLAs may include time constraints. For example, a cloud customer needs his/her data to be completely processed in 24 hrs. Second, a sensing data message has a timestamp indicating when it is sampled. With T-Accountability, any entity violating the time constraint will be detected.

3.4 Spatial Accountability

In MCC, we assume that each mobile device can obtain its spatial location in real time. For some location-sensitive applications like Sensing Map, location information is critical. Hence it is essential to guarantee that each device is accountable for its location.

4 An Accountable Framework for Sensing-oriented MCC

We describe the proposed application framework for Sensing-Oriented MCC in Fig.2. The system is comprised of four parties: customers, cloud, mobile devices, and a Third Party Auditor (TPA). The first three parties jointly contribute to MCC applications, while the TPA is responsible for monitoring and examining the behavior of other parties to ensure accountability. We assume that the TPA is trusted by the whole system.

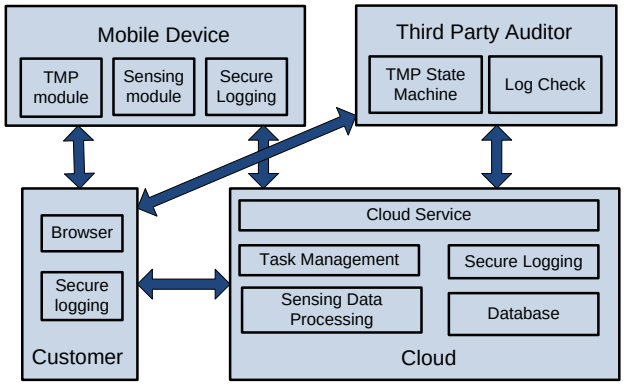

Fig. 2: An application framework for MCC

4.1 Task Management Protocol (TMP)

Task Management Protocol (TMP) is the core of the MCC framework. TMP handles the life cycle of tasks: task creation, task dissemination, task execution, task synchronization, and task revocation. Since TMP is defined in the application level, TMP messages are independent on the lower level communication standards (e.g., cellular network, 3G/4G, WiFi, etc.). We assume that a public and private key pair, and a certificate signed by a Certificate Authority (CA) are possessed by every each customer and each mobile device.

4.1.1 Task Creation

A task defines a customer's interests. By creating and issuing a task, cloud customers are able to initiate a sensing and data collection procedure with specific purposes. Each task is aiming at sensing one or more locations with particular time constraints. A task message is comprised of the following fields:

- **Customer ID** - a customer ID is the account ID that he/she uses to access the web portal of MCC. A customer creating a task is also called the task owner.
- **Task ID** - a task ID is a unique identifier for each task. To prevent adversaries from predicting the task ID, it is necessary to use a Pseudo-Random Number Generator with sufficiently long period to generate the task ID. The approach in [34] can be used to generate pseudo-random numbers with a long period.
- **Data type** - A complete option list of data type is offered to cloud customers. Each customer can choose multiple interested data types; thus the field is a vector of data types chosen by the task owner. Let $D_i = [d_1, d_2, ..., d_k]$ denote the data type vector for task *i*.
- **Report Frequency** - this field specifies that how often mobile devices should report data to cloud.
- **Time frame** - this field is a vector of time intervals that the task owner assigns to mobile devices. If a mobile device chooses to accept the task, it will be desired to finish the task within the time frame. Let $T_i = [(t_{s1}, t_{e1}), (t_{s2}, t_{e2}), ..., (t_{sn}, t_{en})]$ denote the time frame vector for task *i*.
- **Location** - this field specifies a vector of geo-locations that the task owner is interested in. It could be a unique physical address or a famous spot name like "the Statue of Liberty". Let $L_i = [l_1, l_2, ..., l_m]$ denote the location vector for task *i*.
- **Expire date** - this field specifies when a task is expired to 1) prevent malicious task owners from submitting the task repeatedly, and 2) prevent attackers from re-submitting the task.

- **Note** - the note field specifies a human readable message to the mobile device owners as other requirements or preference.

After a task message in generated, the task owner signs its signature with his private key to ensure message integrity, and submits the task to the TMP module on the cloud.

4.1.2 Task Pre-processing

Upon receiving the task, the cloud first examines the attached signature. If the message is intact, the cloud extracts three fields, i.e., data type, time frame, and location, from the task message. The cloud maintains a database that records all task information for all customers. Before assigning the task to mobile devices, the cloud will search the database to look up if a task with the same request has been processed earlier. To do that, the cloud picks one item from each of the three vectors to form a 3-tuple, which is used as a token to conduct a exhaustive search over the database. Let Q_i = {q_j = <d_j, t_j, l_j> | $d_j \in D_i, t_j \in T_i, l_j \in L_i$ } denote a complete search tokens for a task *i*. If q_j is hit, it is moved from Q_i to Q_i', otherwise it stays in Q_i. Once the search is finished, if Q_i is empty, then there is no need to involve mobile devices since the cloud can provide all the information in need; if Q_i is not empty, the cloud has to find eligible mobile devices for data collection. For each element $q_j \in Q_i$, the cloud generates a subtask, which will be sent to eligible mobile devices using multicast. A subtask contains the following: taskID, subtaskID, data type, report frequency, time frame, location, and expire date. In these fields, taskeID, report frequency, and expire date inherit from the original task; data type, time frame, and location form a 3-tuple belonging to Q_i.

4.1.2 Task Dissemination

The cloud also maintains a hash table to store registered mobile devices that are willing to join the TMP. The hash table maps a key that is defined as ($d_j | l_j$), through a hash function $h(x)$, to a slot in the table where IDs of mobile devices with identical keys form a linked list.

A mobile device that supports TMP will turn on the push notification. Meanwhile, a mobile device will report its current location to the cloud periodically, and update the hash table on the cloud end accordingly. Since the reveal of real-time location information to cloud may not be agreed by most mobile users, it is necessary to design a privacy-preserving approach to handle the location data. The scheme presented in [35] can be used here.

For a task *i*, for each $q_j \in Q_i$, the cloud looks up the hash table with key ($d_j | l_j$), and obtains a list of device IDs, which become the recipients of the subtask produced by q_j. After all subtasks are sent, the task dissemination completes.

4.1.3 Task Scheduling and Execution

A mobile device keeps a subtask list in chronological order based on the time frame field in the subtask message. Upon receiving a subtask message, a mobile device first decrypts the message with the session key, and then examines the message content to ensure its integrity and freshness. If the message is intact and fresh, the TMP module on the mobile device generates a message to the device owner, asking the device owner to make a choice to accept it or not. If the subtask is accepted, it is inserted into the subtask list; otherwise the mobile device returns a signed message of denial to the cloud.

The scheduler only needs to check the start time of the first task in the list since the list is sorted chronologically. When it reaches the start time of the first task, the task should be executed automatically or with the assistance from the device owner, depending on the data type. When a task is executed once, the next execution time for the task will be scheduled based on the report frequency. Therefore, the task list needs to be re-ordered to ensure that the next task to be executed is the first node of the list.

4.1.4 Data Reporting

Before the sensing data is reported to the cloud, the mobile device constructs a data message containing the following: device ID, data content, subtask ID, timestamp, location, and a signature of all previous contents from the mobile device. The whole message is encrypted using a session key to prevent eavesdropping. If the subtask is finished, the mobile device sends a control message “TMP_FIN” to the cloud, indicating the end of the subtask.

Upon receiving the data message, the cloud will first examine the data (i.e., confidentiality, integrity, and freshness). After that, the data is either returned to the customer or buffered on the cloud, depending on the requirement of timeliness from the customer.

4.1.5 Task Synchronization

Before a task is entirely finished or expired, its owner is always able to modify the task and update it. Once the task information is changed (e.g., the time is extended, more locations are added), the task should be synchronized in the cloud and in each mobile device that is handling the subtask of it. If a set of fields of the task are changed, a control message

"TMP_DIFF" will be created and sent to the cloud. Let $F_i = \{f_1, f_2, \ldots f_v\}$ denote the fields of task i before the update, and let $F_i' = \{f_1', f_2', \ldots, f_v'\}$ denote the fields of task i after the update. An TMP_DIFF Δ_i is calculated as $\Delta_i = \{diff(f_j', f_j) | j = 1, 2, \ldots, v\}$. Operation $diff(f_j', f_j)$ outputs two sets: $S_1 = f_j' - f_j$ and $S_2 = f_j - f_j'$, in which '-' is the difference operator applied on sets. Therefore, S_1 implies expansion on f_j, while S_2 implies reduction on f_j. The "TMP_DIFF" message includes Δ_i as its content. Upon receiving the "TMP_DIFF" message, the cloud will update task i based on Δ_i. After the task is updated on the cloud, a series of control messages "TMP_SYN" will be constructed and sent to specific mobile devices to update the corresponding subtasks.

4.1.6 Task Post-processing

Upon receiving a "TMP_FIN" message, the cloud will send an acknowledgement "TMP_FIN_ACK" back to the mobile device. When all subtasks are finished, the cloud will assemble all sensing data, no matter if it is from the mobile devices or from the task database. The final result may be processed in the cloud based on the customer preference.

4.1.7 Task Revocation

Once a task is finished, it is labeled as expired even if the expire date is not reached yet. In addition, the cloud will store the sensing data into an indexed database to potentially speed up future query and retrieval.

4.1.8 Accountable TMP

TMP spans the entire platform of MCC. A fundamental need of accountability is to ensure that every participant of TMP strictly follows the protocol specification, and detect any deviation from the protocol with undeniable evidence. To address this problem, we employ a public auditor to check the correctness of TMP execution for each participant. A public auditor is an external party that is trusted by every component of MCC. We use a public auditor because the trust relation among cloud provider, cloud customers, and mobile devices may not be fully established. In other words, a responsible party has to be identified when anomaly is detected by a trustworthy party to every component of MCC. For example, after a customer updates the task, how could he/she ensure that the cloud actually synchronizes the task instead of ignoring it?

4.2 Secure Event Logging

For each TMP participant, every TMP event will be recorded into a tamper-evident secure log file. In other words, log data should be protected from any malicious modification and unauthorized access; in addition, any security violation on log data will be detected and recorded. In this context, we employ a hash-chain based secure logging scheme [9]. Each TMP event has the following attributes: event timestamp, event owner, event location (for mobile device only), event type, and event contents. Let e_k denote the k-th event to be logged. To secure the log, e_k will be associated with a hash value h_k, which can be calculated recursively based on the previous log entries as $h_k = H(h_{k-1} | e_k)$, in which $H(x)$ denotes a hash function, and | the concatenation operator. The base value h_0 is prefixed and known to the system. Therefore, a hash chain is formed along with the log file. Consider two parties A and B, each of which keeps a secure log, by sending a signed hash value h_k from A to B, A commits to having logged event e_k and all events happened before e_k. Any attempt to manipulate the log file before e_k will result in a different h_k, which can be used as an evidence to detect security violation on the log file.

Since mobile devices are storage limited, they store the log files on the cloud by sending a control message called "TMP_LOG". The cloud will maintain a secure log file for each mobile device. A cloud customer can choose to either store log files locally or remotely on the cloud. For each log file, no matter who produces it, there is a chain of hash values; these hash values will be signed by the log file owner (i.e., the one who generates the log entries), and sent to the public auditor from time to time for verification purpose.

4.3 Auditing

A public auditor is responsible for auditing the protocol execution for each TMP participant to check if anyone deviates from the protocol specification. A state machine of TMP is maintained by the auditor as a reference. The auditing process is sketched as follows: periodically, the public auditor will request log files and corresponding hash values from the cloud and its customers. Upon receiving the log files, the public auditor first checks the integrity of the log files by recalculating the hash chain, which is compared with those maintained at other parties of MCC. If the log files are intact, the auditor starts to replay the events in the log files according to the TMP specification. If any entity's behavior deviates from the protocol, the relevant log events will be revealed to the auditor and the auditor may stop the auditing process. Since the public auditor is assumed to be trustworthy, it can use the log files and the replay algorithm based on the TMP state machine as evidence to identify a misbehaved entity.

5 Conclusion

In this paper, we study Sensing-Oriented Mobile Cloud Computing. We propose a general application framework that is centered on a task management protocol. We investigate related accountability issues for the Sensing-Oriented MCC application framework. We believe Sensing-Oriented MCC will bring more opportunities for both researchers and practitioners in the foreseeable future in many application domains.

Acknowledgement

This work is supported in part by the US National Science Foundation under grant numbers 0737325, 0716211, 0829827, 1059265, 1036253, 1040254, and 1044841, as well as the National Natural Science Foundation of China under grant number 61374200.

References

[1] A.R. Yumerefendi and J.S. Chase, "The role of accountability in dependable distributed systems," Proc. of HotDep, 2005.

[2] Huang, D, "Mobile cloud computing," IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter, 2011.

[3] Department of Defense. Trusted Computer System Evaluation Criteria. Technical Report 5200.28-STD, Department of Defense, 1985.

[4] Y. Xiao, "Flow-Net Methodology for Accountability in Wireless Networks," IEEE Network, Vol. 23, No. 5, Sept./Oct. 2009, pp. 30-37.

[5] A. Haeberlen. "A Case for the Accountable Cloud" *3rd ACM SIGOPS International Workshop on Large-Scale Distributed Systems and Middleware (LADIS '09)*, Big Sky, MT, October 2009

[6] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," IEEE Communications Magazine, vol. 48, no. 9, pp. 140 –150, Sep. 2010.

[7] J. Dean, and S. Ghemawat. "*Mapreduce: simplified data processing on large clusters*." In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation. USENIX Association, Berkeley, CA, USA, 2004.

[8] W. Wei, J. Du, T. Yu, and X. Gu, "SecureMR: A Service Integrity Assurance Framework for MapReduce," Proceedings of the 2009 Annual Computer Security Applications Conference, 2009, pp. 73–82.

[9] P. Maniatis and M. Baker. Secure history preservation through timeline entanglement. In Proc. of the 11th USENIX Security Symposium, Jan 2002.

[10] Z. Xiao and Y. Xiao, "Accountable MapReduce in Cloud Computing," in 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2011, pp. 1082 –1087.

[11] J. Liu, and Y. Xiao, "Temporal Accountability and Anonymity in Medical Sensor Networks," ACM/Springer Mobile Networks and Applications (MONET), Vol. 16, No. 6, pp. 695-712, Dec. 2011.

[12] CENS/UCLA, "Participatory Sensing / Urban Sensing Projects"; http://research.cens.ucla.edu/

[13] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones," in Pro. of the 7th ACM Conference on Embedded Networked Sensor Systems, 2009, pp. 85–98.

[14] K. Kumar and Y.-H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," Computer, vol. 43, no. 4, pp. 51 –56, Apr. 2010.

[15] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in Proc. of the 12th conference on Hot topics in operating systems, Berkeley, CA, USA, 2009.

[16] Z. Xiao and Y. Xiao, "Security and Privacy in Cloud Computing," Communications Surveys & Tutorials, IEEE , vol.15, no.2, pp.843,859, Second Quarter 2013

[17] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," Proc. 16th ACM conference on Computer and communications security, 2009, pp. 199-212.

[18] B. D. Payne, M. Carbone, and W. Lee, "Secure and Flexible Monitoring of Virtual Machines," In Proc. ACSAC'07, 2007.

[19] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," In ACM CCS, pages 598-609, 2007.

[20] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," SecureComm, 2008.

[21] K.D. Bowers, A. Juels, and A. Oprea, “HAIL: A high-availability and integrity layer for cloud storage,” Proc. 16th ACM conference on Computer and communications security, 2009, pp. 187-198.
[22] H. Liu, “A New Form of DOS Attack in a Cloud and Its Avoidance Mechanism”, Cloud Computing Security Workshop 2010.
[23] J. Idziorek, M. Tannian, and D. Jacobson, “Detecting fraudulent use of cloud resources,” in Proc. 3rd ACM workshop on Cloud computing security workshop, New York, NY, USA, 2011, pp. 61-72.
[24] Z. Xiao and Y. Xiao, “Achieving Accountable MapReduce in cloud computing,” Future Generation Computer Systems, Vol. 30, No.1, Jan. 2014, pp. 1–13.
[25] http://aws.amazon.com/ec2/
[26] http://code.google.com/appengine/
[27] http://www.microsoft.com/windowsazure/
[28] Introducing the Window Azure Platform, http://view.atdmt.com/action/mrtyou_FY10AzurewhitepapterIntroWindowsAzurePl_1
[29] S. Pearson and A. Charlesworth, “Accountability as a Way Forward for Privacy Protection in the Cloud,” Proceedings of the 1st International Conference on Cloud Computing, Beijing, China: Springer-Verlag, 2009, pp. 131-144.
[30] M. Barua, X. Liang, R. Lu, X. Shen, "ESPAC: Enabling Security and Patient-centric Access Control for eHealth in cloud computing ," International Journal of Security and Networks, Vol. 6 Nos. 2/3, 2011, pp. 67-76.
[31] D. Sun, G. Chang, C. Miao, and X. Wang,"Modelling and evaluating a high serviceability fault tolerance strategy in cloud computing environments," International Journal of Security and Networks," Vol. 7, No. 4, 2012, pp. 196-210.
[32] S. Kalra and S. Sood, "ECC-based anti-phishing protocol for cloud computing services," International Journal of Security and Networks, Vol. 8, No. 3, 2013, pp. 130-138,
[33] D. Santos, T. Nascimento, C. Westphall, M. Leandro, and C. Westphall, "Privacy-preserving identity federations in the cloud: a proof of concept," International Journal of Security and Networks, Vol. 9, No. 1, 2014, pp. 1-11.
[34] A. Olteanu, Y. Xiao, F. Hu, B. Sun, and H. Deng, " A Lightweight Block Cipher Based on a Multiple Recursive Generator for Wireless Sensor Networks and RFID," Wireless Communications and Mobile Computing (WCMC) Journal, John Wiley & Sons, Vol. 11, No. 2, pp. 254-266, Feb. 2011.
[35] S. Ozdemir, M. Peng, and Y. Xiao, " PRDA: Polynomial Regression Based Privacy Preserving Data Aggregation for Wireless Sensor Networks," Wireless Communications and Mobile Computing (WCMC), accepted. DOI: 10.1002/wcm.2369
[36] S. Yue, Y. Xiao, and G. Xie, "Fault Tolerance Experiments in 4D Future Internet Architecture," Journal of Internet Technology, Vol. 11 No. 4, pp. 543-552, July 2010.
[37] X. Yan, B. Chen, L. Tong, X. Hu, and Y. Pan, "Adaptive dual cluster heads collaborative target tracking in wireless sensor networks," International Journal of Sensor Networks, Vol. 15, No. 1, 2014, pp. 11-22.
[38] P. Zou and Y. Liu,"Low energy WSN data aggregation algorithm based on improved aggregation tree model," International Journal of Sensor Networks, Vol. 15, No. 3, 2014, pp.149-156.
[39] C. Liu, K. Wu, Y. Xiao, and B. Sun," Random Coverage with Guaranteed Connectivity: Joint Scheduling for Wireless Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 17, No. 6, June 2006, pp. 562-575.
[40] S. Ozdemir and Y. Xiao, "Secure Data Aggregation in Wireless Sensor Networks: A Comprehensive Overview," Computer Networks, Vol. 53, No. 12, Aug. 2009, pp. 2022–2037.
[41] Y. Wang, W. Chu, Y. Zhang, and X. Li, "Partial sensing coverage with connectivity in lattice wireless sensor networks," International Journal of Sensor Networks, Vol. 14, No. 4, 2013, pp. 226-240.
[42] K. Wu, Y. Gao, F. Li, and Y. Xiao, "Lightweight Deployment-Aware Scheduling for Wireless Sensor Networks," ACM/Springer Mobile Networks and Applications (MONET), Vol. 10, No.6, pp. 837–852, Dec. 2005.
[43] L. Wang and Y. Xiao, "A Survey of Energy-Efficient Scheduling Mechanisms in Sensor Networks,” ACM/Springer Mobile Networks and Applications (MONET), Vol. 11, No. 5, 2006, pp. 723-740.
[44] Y. Xiao, V. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A Survey of Key Management Schemes in Wireless Sensor Networks," Computer communications, Vol. 30 No. 11-12, Sep. 2007. pp. 2314–2341.
[45] Wen-Lung Shiau, Han-Chieh Chao, Chia-Pin Chou, “An Innovation of an Academic

Cloud Computing Service", Journal of Software Engineering and Applications, Vol. 5, No 11, pp. 938-943, November 2012.