Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 18 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i18.57223

PAPER

Enhancing Mobile Application Security through Mobile Crowd Sensing and Sourcing Solutions

Rakesh Ranjan[1], Siddhanta Kumar Singh[2](✉), Saurabh Dhyani[3], Doddi Srilatha[4], Dharmesh Dhabliya[5], Sumit Kumar[6]

[1]Department of Computer Science and Engineering, ABES Engineering College, Ghaziabad, Uttar Pradesh, India

[2]Department of Computer and Communication Engineering, Manipal University Jaipur, Jaipur, Rajasthan, India

[3]Uttaranchal School of Computing Sciences, Uttaranchal University, Dehradun, Uttarakhand, India

[4]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Bowrampet, Hyderabad, Telangana, India

[5]Department of Information Technology, Vishwakarma Institute of Technology, Pune, Maharashtra, India

[6]Department of CSE, Shivalik College of Engineering, Dehradun, Uttarakhand, India

siddhanta.singh@jaipur.manipal.edu

ABSTRACT

The market for Android applications has expanded dramatically, providing users with an ever-expanding array of functionality to meet a range of needs. Because mobile applications are so widely used, users are sharing more sensitive data, so protecting personal data is essential. However, this expansion has also led to a commensurate increase in cyber security threats, particularly for malware and adware that target mobile devices. To strengthen the mobile ecosystem, it is essential to divide mobile applications into discrete categories such as benign, adware, and malware. A unique sensing technique called Mobile Crowd Sensing/Sourcing (MCS) uses users' collective participation and their mobile devices to gather sensing data. Artificial intelligence (AI) approaches are used to make well-informed decisions that help optimize system performance as the MCS platform stores and processes vast amounts of data. The investigation primarily focuses on how AI is being applied to various MCS components, such as task distribution and data aggregation, to boost security and performance. Additionally, a novel categorization system that may be modified to compare research in this field is proposed in this paper. Because it makes it easier to identify attack surfaces that adversaries can exploit, this framework can be used to study AML in the context of MCS. It also highlights the potential vulnerabilities of AI-based MCS systems to adversarial attacks, which encourages future research to concentrate on designing resilient systems.

KEYWORDS

mobile crowd sensing/sourcing (MCS), Android applications, artificial intelligence (AI), medusa

1 INTRODUCTION

Since mobile devices with strong sensing capabilities are so widely available, people are actively taking part in extensive data collection projects and offering insightful commentary on a range of phenomena. Expanding data collecting and processing capabilities with today's smart gadgets can help businesses creatively address several issues that affect residents and progress in many scientific sectors [1].

Ranjan, R., Singh, S. K., Dhyani, S., Srilatha, D., Dhabliya, D., Kumar, S. (2025). Enhancing Mobile Application Security through Mobile Crowd Sensing and Sourcing Solutions. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(18), pp. 63–76. https://doi.org/10.3991/ijim.v19i18.57223

Article submitted 2025-04-17. Revision uploaded 2025-06-23. Final acceptance 2025-06-25.

Using the combined participation of a wide range of users and the sophisticated detection capabilities of their cell phones, Mobile Crowd Sensing/Sourcing (MCS) is a unique sensing technique that gathers sensing information. In mobile crowd sensing, people respond to a task posted by a task publisher by actively or passively contributing data using detectors built into their smart devices. On the other hand, mobile crowdsourcing assigns jobs to a team of individuals who must travel to the task site to gather data.

Three entities are commonly included in MCS systems, as seen in Figure 1 task requesters, mobile users, and the leadership platform. First, the requesters send in sensing activities, outlining the type of information to be gathered and the needs and limitations, like the task's start and finish times, budget, and location [2]. The software then assigns jobs to individuals, who gather the sensing data utilizing their intelligent gadgets. Following data collection, the platform gathers and stores the gathered information and provides the job requesters with the final sensing outcome. In addition, users receive payment for their involvement.

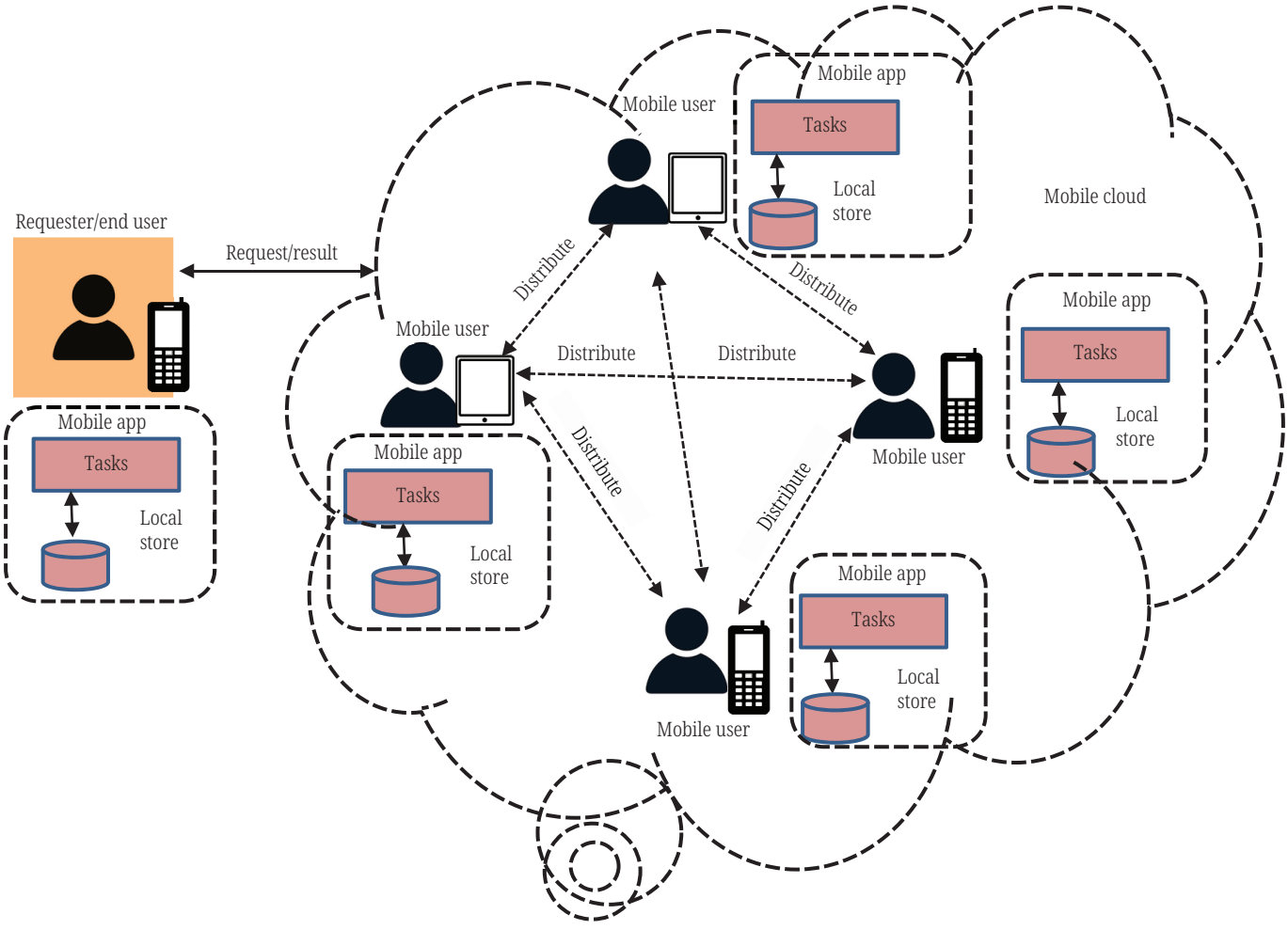

Fig. 1. Generic architecture of decentralized crowdsourcing

Figure 1 shows an illustration of the architecture. The design is comparable to a mobile peer-to-peer network, in which any peer can interact with nearby peers and help discover patterns of mobility. Mobile devices either intentionally or inadvertently collect sensor data in the background throughout the crowdsourcing process.

Information processing and storage in the local database and mobile device are handled by the decentralized architecture.

While artificial intelligence (AI)-based MCS systems have advanced to a point where they are effective in raising the overall quality of service of these frameworks, new issues are raised that, ironically, may hurt the system's functionality. For example, the machine learning models used to identify fraudulent requests or forecast future ride-sharing application demands may be the target of adversarial assaults. Attackers can therefore send drivers fictitious orders to the same address, which would result in heavy traffic. The investigation of attacks intended to impair the performance of machine learning models and the development of defenses against these attacks is the focus of the field of adversarial machine learning, or AML [3].

The organization of this document is described in the structure that follows. The body of existing literature that is relevant to the subject is examined in Section 2. Section 3 defines the problem and describes the architecture of the system model. Section 4 discusses the model training process, the simulation techniques employed, and the results analyses. An overview of the findings is provided in Section 5, which wraps up the report.

2 RELATED WORKS

The edge cloud, which can offer customers compute offloading services, will be implemented in the ultra-dense network of the future communication network [4]. Users can assign computationally demanding tasks to the edge clouds. However, the majority of current research on mobile edge computing and ultra-dense networks is done independently. The following obstacles must be overcome for mobile edge computing to be used effectively in ultra-dense networks:

(i) The extensive deployment of small cell base stations in ultra-dense networks makes network settings more complex. Furthermore, edge cloud computing power is constrained. Controlling these dispersed computing resources is therefore a difficult problem; (ii) users have little knowledge about the wireless networks to be accessed, such as the processing load of edge clouds and the traffic load of the accessible network, when they offload computing activities.

MCSC is a large-scale sensing paradigm that leverages the capabilities of user-companion devices, such as wearable technology, smart cars, and cell phones. A growing number of mobile phone users can share local knowledge (such as traffic conditions, noise levels, ambient context, and local information) obtained by their sensor-enhanced devices thanks to MCSC. The data can also be further gathered in the cloud for community intelligence mining and large-scale sensing [5]. Because of the vast number of mobile users, MCSC is a flexible platform that frequently takes the role of static sensor infrastructures. Thus, a wide range of uses are made possible, such as public safety, mobile social suggestion, environmental monitoring, traffic planning, and so forth.

Despite all of MCS's advantages, there are still a lot of significant issues with trust, security, and privacy. First, because of the nature of openness and mobility, it is simple to act selfishly and launch attacks. Serious security risks to MCS would result from this, including collusion, altered data uploading, surveillance, and tracking. Second, a key concern with MCS is privacy. Numerous sensitive details about mobile users may be included in the data gathered through MCS, which is directly

related to their anonymity [6]. This makes it possible for hackers to deduce private user information from the data that has been gathered. Certain MCS programs, for instance, gather GPS fixes or cell phone IDs, which allow for the inference of a user's location and physical activity. Furthermore, because the work an MCS service requester requests may involve sensitive data, their privacy may potentially be at jeopardy.

A new twist on crowdsourcing apps has recently been brought about by the increasing rise of online social media, where users can actively use their underlying social network to invite others to assist in completing the crowd-sourced assignment. In addition to trying to complete the assignment alone, these people encourage others in their social networks to try the task as well, expanding the pool of crowd workers. Crowd workers actively participate in the WoM-based MCS system since, unlike in direct mode, tasks are distributed to a pool of starting workers and disseminated worker-to-worker [7]. MCS architecture with pre-existing apps to provide ubiquitous cloud services, where mobile users can create a mobile cloud on their own to carry out crowdsourcing tasks including computing, data collection, and processing. By using different device-to-device technologies (such as Bluetooth, Wi-Fi Direct, and LTE Direct) as the basic communications substrate, we contend that the local MCS can be effectively constructed using the WoM mode.

Wireless sensor networks (WSN) are the primary focus of opportunistic sensing, also known as crowd-sensing techniques, as detailed in the literature [8]. Interestingly, the majority of these opportunistic sensing designs use the sensors as data sources and do mobile analytics on faraway servers or in the cloud. An opportunistic method of disseminating data for sensor networks on mobile phones. This strategy uses intermediately fixed infrastructure that manages mobile data analytics to get data from mobile devices. Efficient data routing from source to destination is the main goal of the mobile data analytics suggested in this study. Infinite recursion and heap exhaustion attempts were the two attacks that it failed to detect since they targeted specifically resource utilization. Dynamic tracking is necessary for these attacks; we will show how Medusa's dynamic resource tracking feature works next.

Mobile crowd sensing is a novel sensing paradigm that makes use of mobile devices, particularly smartphones, in contrast to the conventional physical sensor-based sensing paradigm. A new and rapidly expanding sensing paradigm is mobile crowd sensing, which involves using sensor-enhanced cellphones to gather local knowledge about things like location, personal and environmental context, noise level, automobile traffic, and, in the future, more specialized information like pollution. This knowledge can then be shared with others in the social circle, healthcare providers, and utility providers [9]. In contrast to static platforms based on physical sensors, mobile crowd sensing is a dynamic sensing platform since the data it gathers is typically from people's mobile devices. Mobile crowd sensing has the potential to detect using a ubiquitous and inexpensive mechanism because of its dynamic feature.

Given these difficulties, this study first examines the body of research on mobile crowdsourcing, which is based on an analysis of noteworthy initiatives that combine mobile sensing with the capacity to assign tasks or computations to people. We go over the fundamentals of the mobile crowd-sourced paradigm while showcasing several illustrative mobile platforms [10]. With an emphasis on the features of applications and system architectures, we also offer a taxonomy of the problems in this field along with several dimensions and methodologies that have been used

to address them. Additionally, we list some frameworks, approaches, important factors, and difficulties in creating mobile crowdsourcing apps. Lastly, we enlarge each dimension to emphasize the distinct set of implementation requirements that must be taken into account when creating and assessing such cooperative systems. Next, we go over upcoming developments in mobile crowdsourcing as well as the extension of mobile crowd-sourced to think crowdsourcing.

3 METHODS AND MATERIALS

3.1 Architectures of crowdsourcing and proposed crowdsourcing

The sorts of crowdsourcing architectures and their research will be covered in this section. The general framework and essential elements of crowdsourcing are covered first in this section. Generally speaking, client-server architectures and the Internet can be used to construct crowdsourcing systems that make use of everyday gadgets such as cell phones and PCs. There will be a thorough evaluation of the research projects [11]. The centralized and decentralized crowdsourcing architectures are also presented in this section. A client-server design or model is usually used in a centralized crowdsourcing technique. The primary or centralized information center houses the computing resources, and the main hub or server handles the computational demands of these services. Conversely, crowdsourcing's decentralized architecture is a dispersed strategy. Peer-to-peer and in-situ processing are made possible by the layered structure, which removes the need for all components to link to a central hub and eases traffic congestion there. After examining both centralized and decentralized crowdsourcing designs, it is seen that while numerous architectures have been put forth for crowdsourcing, they are not specially made for crowdsourcing the Internet of Things. Therefore, a novel architecture for crowdsourcing IoT, known as Crowd-IoT, will be proposed in this part.

3.2 Centralized architectures of crowdsourcing

The majority of research on general crowdsourcing architectures concentrated on features such as work management, data processing, storage, capture, security, scalability, and privacy. Several application architectures were centered on assigning tasks to participants and managing them. The broad or centralized crowdsourcing architecture serves as the foundation for the majority of these projects [12]. There are reviews and research on crowdsourcing, including its technologies, methods, and applications. As far as we are aware, no thorough analysis of IoT structures for crowdsourcing or IoT designs for crowdsourcing has been done. The scholarship on mobile crowd-sourced research is surveyed by the authors. Based on common responsibilities, components, and functionalities, the authors categorized crowdsourcing architectures from existing mobile applications and architectures [13]. The examination offers a thorough grasp of common features, design elements, and issues that arise when developing mobile crowdsourcing systems.

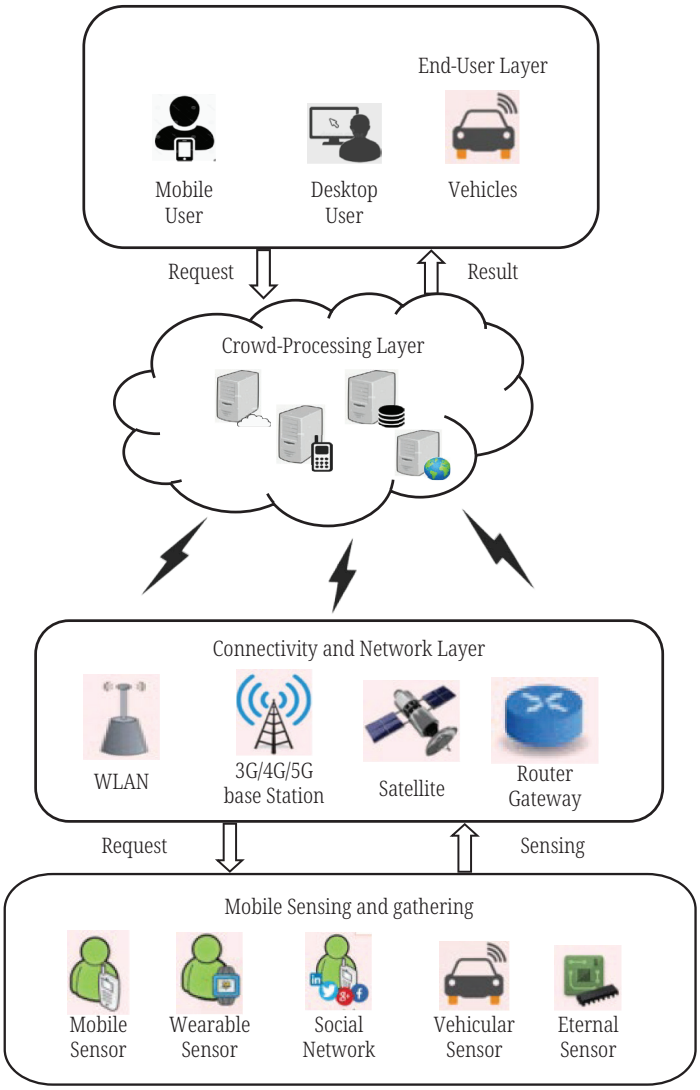

Fig. 2. Centralized architecture for mobile crowdsourcing

IoT-based centralized mobile crowdsourcing architecture that is generic. Centralized mobile crowdsourcing architecture for the IoT is shown in Figure 2, where the cloud server handles information processing. Four levels make up this architecture: (1) mobile sensing/gathering; (2) network/connection; (3) crowd processing; and (4) end-user. Sensors or sensing devices, such as wearable technology, smart appliances, smart cars, smartphones, and other user devices, are part of the mobile sensing layer.

Providing network connectivity for mobile crowdsourcing is the connectivity layer's primary responsibility. Cellular networks (3G, 4G, and 5G) [14]; wireless sensor networks, Wi-Fi, Bluetooth, and vehicle ad hoc networks (VANETs) are examples of these communication networks. These communication methods send the information gathered by sensors—including mobile devices—to a server or cloud. Data from sensors, mobile devices, and crowd tasks are saved, processed, examined, and displayed on the crowd processing layer. The centralized paradigm of cloud computing is essential to the aforementioned goals.

Task distribution and the growth of conventional crowdsourcing networks can also be achieved using the Social Internet of Things (SIoT). One benefit of using

crowdsourcing in SIoT is that it can expand the number of workers by fostering the growth of big social networks. Since SIoT is typically coordinated based on friendship, it can be acquired without any incentive problems in SIoT crowdsourcing architecture. A reliable crowdsourcing paradigm for SIoT was put forth by the authors. Sensing entities, end users, and the social cloud are the three parts of the architecture. The foundation layer, components layer, and application layer make up its three layers. The base layer's job is to maintain the upper-layer services by providing the computational and storage infrastructure. Tools for fundamental component implementation, including object profiling, ID management, and owner control, are provided by the component layer. The purpose of the application layer is to act as a conduit between end users, human actors, and sensing entities.

3.3 Mobile crowd-sensing applications

This section will provide a concise overview of the many mobile crowd-sensing applications that are now available, outline their features, highlight a number of research obstacles, and then go over potential fixes. Based on the kind of occurrence being seen, MCS applications can be broadly divided into two categories: community sensing applications and personal sensing applications. Community-sensing apps are used to monitor large-scale phenomena that are difficult to detect with individual applications, whereas personal sensing applications deal with phenomena that affect a single person. These apps connect computing devices, exchange data, and then use the information to map common occurrences that typically fall into one of four categories: social life, infrastructure, healthcare, and the environment.

Smartphones can link people to medical services through wireless communication networks for monitoring and diagnostic purposes in healthcare monitoring. Water levels in creeks, urban air pollution, and wildlife habitats are some of the primary phenomena to be felt in environmental applications in order to track their behavior for future research. The primary phenomena to be observed in infrastructure applications are traffic jams, roadway conditions, availability of parking, and public works outages. For example, people can compare their daily workout regimens and share how much time they spend exercising each day. An additional illustration would be the ability for people to exchange exercise data and subsequently assess how active they are in comparison to the community at large. Furthermore, the folks might improve their everyday workouts by using this comparison. For the four distinct crowd-sensing categories of observed occurrences, the summary is displayed in Table 1.

The four types of MCS monitoring applications indicated in Table 1 are adequately served by mobile sensing and smartphone devices.

Table 1. Crowd-sensing type of measured phenomena

MCS Application	Used in	Examples
Healthcare	Monitoring the vital signs in healthcare	Calculate heart rate, EEG, and ECG
Environmental	Evaluation of the natural environment's parameters	Habitats for wildfires, air pollution, and water levels
Infrastructure	Evaluating the public infrastructure's condition	Roads, traffic jams, bridal flaws, and structural health monitoring
Social	Calculating information about a person's social life	Movies that a person goes to, regular workouts, or sports

4 IMPLEMENTATION AND EXPERIMENTAL RESULTS

Stage initiation alerts are obtained by the Stage Tracker through SMS message interception. The Stage Tracker parses the XML stage description sent in each SMS message, downloads the stage binary if required [15], and starts the stage's execution. Also included in the SMS message are the values of named variables that were instantiated during earlier computation stages; keep in mind that these contain references to data items that are kept on the phone. Before the stage runs, the namespace—which contains these named variables and their values—must be set up by the Stage Tracker. Along with implementing the triggers required to commence stage execution (such as to begin recording a video clip), it is also in charge of providing the human mediation needed for annotations.

Table 2. Property and components of the medusa system

System Components	Requirements
MedScript Interpreter	Timeliness enforcement
Task Tracker	Support Multiple users, Un-Synchronized user operation
Worker Manager	Incentives, Crowd-curation, Anonymity
Stage Library	In-Network processing, Sandboxing
Stage Tracker	Multiple Concurrent stages, Robustness
MedusaBox	Stateless Execution, Sandboxing, Concurrent Stages

Together, Medusa's many parts meet the intricate network of crowd-sensing specifications. Table 2 lists the requirements that each component satisfies [16]; several components may meet a condition.

Table 3. Put crowd-sensing apps into practice

Application	LOC*	Sensors	Properties
Video Documentation	95	Camera	In-network processing, Crowd curation
Collaborative Learning	71	Accelerometer, Audio	Different Sensors
Auditioning	75	Camera	Reverse incentive mechanism, Crowd curation
Forensic Analysis	95	GPS, Audio	Access to Stored Data
Spot Reporter	54	Camera, Audio	Text/Voice tagging
Road Monitoring	81	Accelerometer	Fork-join construct, App from PRISM
Citizen Journalist	54	GPS	Multiple trigger, App from PRISM
Party Thermometer	73	GPS, Audio	Fork constraint App from PRISM
WiFi and Bluetooth Scanner	54	Network Sensors	Apps from Anony-Sense

Medusa makes it possible for many of the new jobs in Table 3. We go over some of the nuances of the video documentation assignment here, even though we have already covered the code. First, because of space constraints, we depict the task code as a control flow diagram for this task and the others that are covered below. A complete program listing for the other tasks is not possible. Squares stand for

HIT stages, while ovals indicate SPC phases. The failure connectors are condensed for conciseness, and only the positive connectors are displayed.

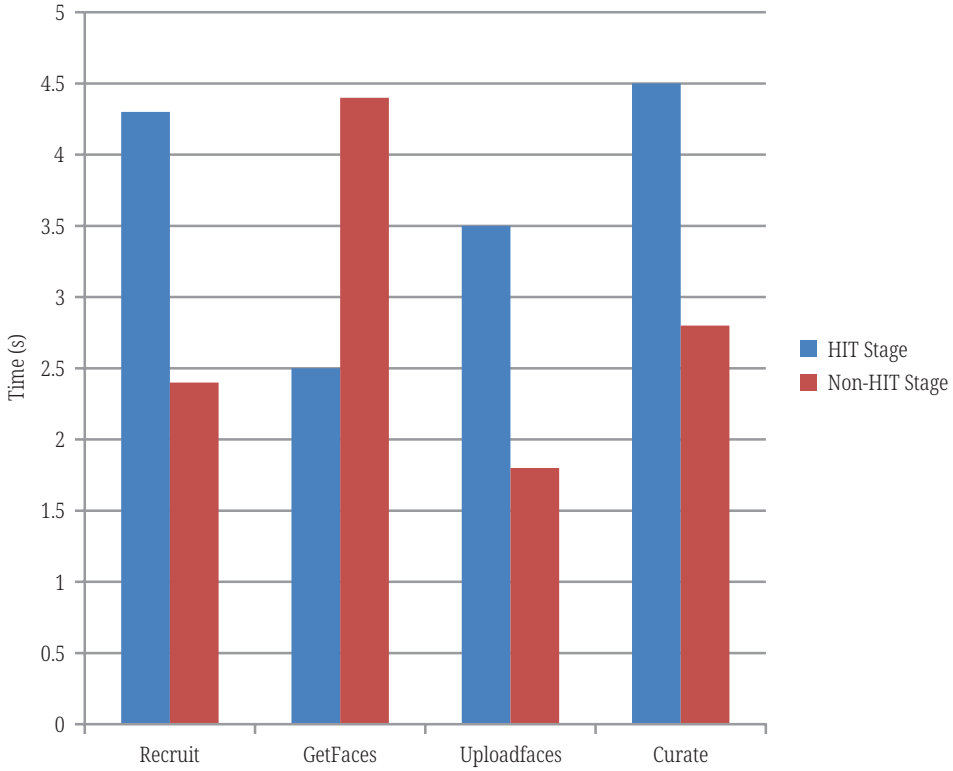

Fig. 3. Analysis of forensics

Security personnel can get pictures of individuals who were present at a specific place at a specific time when an incident happened thanks to our forensic analysis work (see Figure 3). Access to previous sensor data saved on the smartphone is the task's unique feature. The GetImages step makes this access possible by retrieving photos whose metadata corresponds to a specific spatiotemporal predicate. Each image's faces are extracted in the GetFaces stage that follows. For curation, only the faces that have been extracted are posted.

Table 4. Comparison of lines of code

	Medusa	Standalone
Video Documentation	97	9764
Collaborative Learning	73	8077
Spot Reporter	55	19239

Our previous independent concepts for collaborative learning and video recording served as inspiration for Medusa. Although the earlier prototypes featured extra user interface components for software configuration and management, they lacked support for worker mediation or incentives. Furthermore, we work with a research team that has developed a spot reporter program that assigns human agents to transmit text, audio, or video comments from the field. Three steps are needed to complete this task: recruiting agents, filming the video (along with annotations), and uploading the finished product.

Table 4 uses lines of code as a metric to quantify the variation in code complexity. The Medusa versions are roughly two orders of magnitude more compact than the standalone versions, despite the variations in implementation! Many of the features shared by these stand-alone apps are implemented by Medusa's runtime, which lessens the workload for programmers.

Table 5. Telephone delay breakdown

Components	Average	Maximum	Minimum
Retrieve and parse SMS/MMS command message	49.3	67	37
Stage binary initiation time	503.8	605	275
Stage Runner Latency	7.3	23	4
Stage termination latency	13.8	23	4
Total overhead imposed	558.9	697	508

The delay breakdown for the Medusa runtime on a Nexus One phone is shown in Table 5. It is necessary to combine several SMS messages and parse the resulting XML data object in order to retrieve a stage start message from Task Tracker. The average latency for this is 28.2 ms. The primary cause of the phone's delay is the 502.7 ms required to instantiate the stage implementation binary (in our tests, the stage implementation was cached on the phone). Android imposes this overhead; the Android runtime must unpack our stage implementations, which are Android package files, and load the class files into the virtual machine.

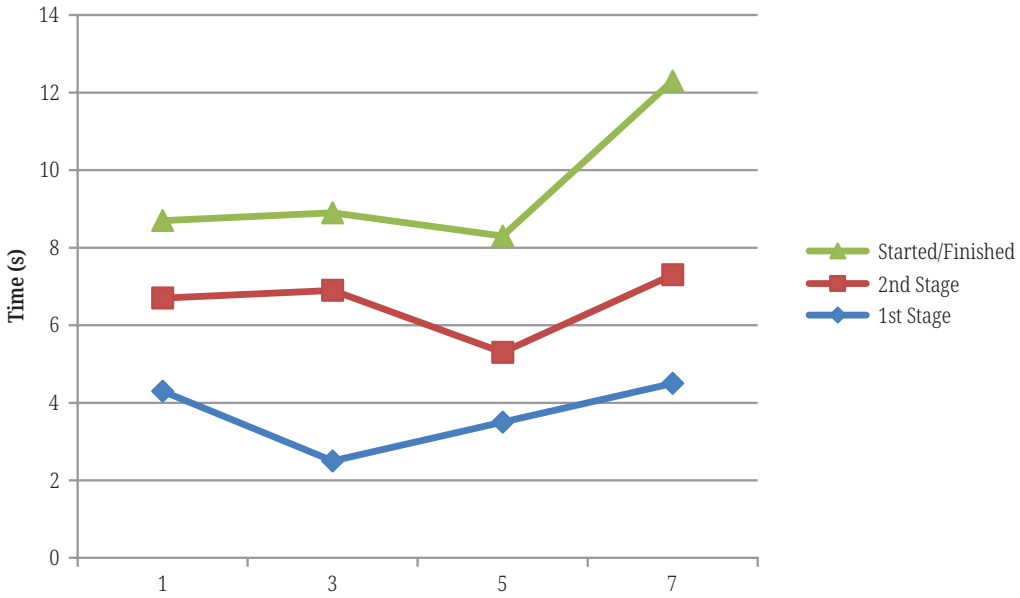

Fig. 4. Robustness: imposing a computation constraint

We constructed a single long-running stage and concurrently executed two versions of that stage with constraints of 5 and 10 seconds, respectively, to demonstrate how these constraints are applied. In our approach, MedBox's watchdog timer verifies stage resource utilization every three seconds. Our stages were successfully ended at 5.36 and 12.85 seconds, as seen in Figure 4; but, given the minuteness of

our timer, processes might get bounded with extra power beyond their limitations. Such a system can defend against attacks that use up all of the CPU's resources.

Table 6. The static analyzer's ability to prevent arbitrary code changes

Code Modification	Result
Make one function sleep for a long time	REJECT
Write a file to SD Card	REJECT
Time String Format translation	REJECT
Open HTTP Connection	REJECT
Delete All files in SD card	REJECT
Vector operations	ACCEPT
Throws Exceptions	REJECT
System.exit(–1)	REJECT
Recursive Calls	ACCEPT
Fills the heap until memory is Exhausted	ACCEPT

We enlisted three volunteers (none of whom are the paper's authors) to insert dangerous code into pre-existing stage implementations in order to show how effective static analysis is for sandboxing. We next examined if our analysis could prevent these types of attacks, which included accessing the SD card, initiating sleep functions, and opening HTTP connections, among others. Seven of the nine changes the volunteers made were successfully detected by our static analyzer (refer to Table 6).

5 CONCLUSION

In conclusion, the study demonstrates how implementing AI could enhance MCS system performance across a range of phases, such as data collection and task distribution.

MCS's widespread use and adaptability have made it a successful and efficient technique for gathering and analyzing data. Because of its openness and unreliability, MCS still has a lot of issues with security, privacy, and trust despite its many advantages. In both academia and industry, several problems have not yet been thoroughly examined. We conducted a comprehensive analysis of MCS's confidentiality, safety, and belief in this investigation. By contrasting MCS with WSN and conventional online crowdsourcing, we presented the fundamental structures of MCS and examined its unique features.

Medusa is a crowd-sensing programming system. High-level abstractions for crowd-sensing job phases and control flow between them are offered by the MedScript programming language. Medusa allows the sensing workflow to specify different types of worker mediation. A runtime that is divided between the cloud and smartphones satisfies a wide range of needs, from user-specified controls on smartphone resource utilization to support for incentives. The runtime system has minimal overhead, and Medusa job descriptions are brief.

We thoroughly examined the current literature and discussed the benefits and drawbacks of the work that has already been done, keeping the requirements as

our primary criterion. Lastly, we looked at the unresolved problems that haven't been thoroughly examined yet and suggested some lines of inquiry to help guide future work.

6 REFERENCES

[1] M. H. ur Rehman, S. L. Chee, T. Y. Wah, A. Iqbal, and P. P. Jayaraman, "Opportunistic computation offloading in mobile edge cloud computing environments," in *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, 2016, vol. 1, pp. 208–213.

[2] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018. https://doi.org/10.1109/TCOMM.2018.2866572

[3] T. M. Fernández-Caramés, I. Froiz-Míguez, O. Blanco-Novoa, and P. Fraga-Lamas, "Enabling the internet of mobile crowdsourcing health things: A mobile fog computing, blockchain and IoT based continuous glucose monitoring system for diabetes mellitus research and care," *Sensors*, vol. 19, no. 15, p. 3319, 2019. https://doi.org/10.3390/s19153319

[4] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018. https://doi.org/10.1109/JSAC.2018.2815360

[5] B. Guo *et al.*, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 1–31, 2015. https://doi.org/10.1145/2794400

[6] A. Zaslavsky, P. P. Jayaraman, and S. Krishnaswamy, "Sharelikescrowd: Mobile analytics for participatory sensing and crowd-sourcing applications," in *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW),* 2013, pp. 128–135. https://doi.org/10.1109/ICDEW.2013.6547440

[7] V. C. Farias da Costa, L. Oliveira, and J. de Souza, "Internet of Everything (IoE) taxonomies: A survey and a novel knowledge-based taxonomy," *Sensors*, vol. 21, no. 2, p. 568, 2021. https://doi.org/10.3390/s21020568

[8] M. Tabatabaie and S. He, "Naturalistic e-scooter maneuver recognition with federated contrastive rider interaction learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 6, no. 4, pp. 1–27, 2023. https://doi.org/10.1145/3570345

[9] A. H. Salem, I. W. Damaj, and H. T. Mouftah, "Vehicle as a computational resource: Optimizing quality of experience for connected vehicles in a smart city," *Vehicular Communications*, vol. 33, p. 100432, 2022. https://doi.org/10.1016/j.vehcom.2021.100432

[10] S. Han *et al.*, "Location privacy-preserving distance computation for spatial crowdsourcing," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7550–7563, 2020. https://doi.org/10.1109/JIOT.2020.2985454

[11] E. Zhang, R. Trujillo, J. M. Templeton, and C. Poellabauer, "A study on mobile crowd sensing systems for healthcare scenarios," *IEEE Access*, vol. 11, pp. 140325–140347, 2023. https://doi.org/10.1109/ACCESS.2023.3342158

[12] S. Schuhbäck, L. Wischhof, and J. Ott, "Cellular sidelink enabled decentralized pedestrian sensing," *IEEE Access*, vol. 11, pp. 13349–13369, 2023. https://doi.org/10.1109/ACCESS.2023.3242946

[13] A. Abozeid, A. A. AlHabshy, and K. ElDahshan, "A software security optimization architecture (SoSOA) and its adaptation for mobile applications," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, no. 11, pp. 148–165, 2021. https://doi.org/10.3991/ijim.v15i11.20133

[14] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[15] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018. https://doi.org/10.1109/ACCESS.2018.2805798

[16] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016. https://doi.org/10.1109/JSAC.2016.2611964

7 AUTHORS

Dr. Rakesh Ranjan currently working as a Professor in the department of Computer Science and Engineering at ABES Engineering College, Ghaziabad, Uttar Pradesh-201009, India. He is actively involved in the field of academic and research from more than last 24 years. He has completed Bachelor of Engineering and Master of Engineering in the field of Computer Science & Engineering, from the Kharkov National Technical National University, Ukraine (Former USSR) in 1998. He has also earned a degree of Master of Technology in the field of Computer Science & Engineering from Uttarakhand Technical University, Dehradun, India. He has earned Ph.D. in his area of interest image processing in 2016. He has large number of publications in renowned referred journals/conference proceedings and Patents (E-mail: rakesh.ranjan@abes.ac.in).

Dr. Siddhanta Kumar Singh has 25 years of work experience in the fields of software training in engineering colleges, university, and multinational companies in IT technologies in India and China. He is working as an Assistant Professor at Manipal University Jaipur. He did his B.E. (Computer Science and Engineering), M.Tech (Computer Science. and Engineering) and Ph.D. (Computer Science and Engineering). His areas of research interest includes board diagnostic II, driving behavioral analysis, ML, and IoT (E-mail: siddhanta.singh@jaipur.manipal.edu).

Dr. Saurabh Dhyani is working as an Assistant Professor in Department of Computing Sciences at Uttaranchal University, Dehradun, Uttarakhand. He has 4 years of teaching experience. He holds a Ph.D. degree in Computer Applications from Maulana Azad National Institute of Technology, Bhopal, M.P. India. His research area includes Big Data, recommendation system, natural language processing, machine learning and deep learning (E-mail: saurabhdhyani@uumail.in).

Dr. Doddi Srilatha is working as an Associate Professor in the Department of CSE at Koneru Lakshmaiah Education Foundation, Bachupally Campus, Hyderabad, India. She received her B.Tech and M.Tech from JNTU Hyderabad, India. She awarded Ph.D. from REVA University, Bengaluru, India. Her area of interest includes software engineering, cloud computing, network security, data mining, and machine learning. She is an Oracle-certified Java Programmer and AWS-certified Cloud Practitioner and Solutions Architect. She has published more than 15 research papers in reputed journals (E-mail: psrilatha@klh.edu.in).

Dharmesh Dhabliya has graduated from Vishwakarma Institute of Information Technology with Computer Engineering Specialization and obtained Master's Degree from Tulsiramji Gaikwad Patil college of Engineering and Technology Nagpur India. Currently, he is working as a Professor at Vishwakarma Institute of Information

Technology Pune. He has published 22 SCI papers and 33 Scopus papers with seven books on his name (E-mail: dharmesh.dhabliya@viit.ac.in).

Dr. Sumit Kumar is currently a Professor and Head of Department of CSE, Shivalik College of Engineering, Dehradun. He got awarded with Doctorate Degree in the year 2019 in the field of Computer Science and Engineering Major for his research work "Prediction of student's performance using K-Star Algorithm". He obtained his M. Tech in CSE degree in the year 2012 from BPUT, Rourkela, Odisha. He obtained his B. Tech in CSE in the year 2006 from KIIT Deemed University, Bhubaneswar which is recognized as center of Eminence in India and have obtained a QS World Ranking of 800–1000. He has more than 14 years of long teaching experience and he taught various major subjects of Computer Science and Engineering during this tenure. In essence, he embodies excellence in AI & ML education, research and institutional leadership (E-mail: headcse@sce.org.in).