**Enhanced data privacy preservation model for Mobile crowd sensing system using blockchain technology**

***Abstract.*** With the recent advancement in computing and sensing capabilities of smart terminals, mobile crowdsensing systems built on smart and ubiquitous mobile terminals have become one of the rising applications allowing better sensing and scalable systems. However, the prior mobile crowdsensing system mostly employs centralized architecture, which is subject to low reliability and are vulnerable to malicious attacks. Furthermore, it is difficult to guarantee the privacy with the transparency of MCS systems. These factors have been a major hindrance to the growth of MCS applications. Hence it is imperative to ensure security and privacy of such systems in order to meet the wide spectrum of its’ applications. So we propose a decentralized blockchain crowdsensing system to harness the benefits of this sensing paradigm. We plan to integrate blockchain technology into the crowdsensing system. Meanwhile, we also focus on how to protect data privacy in blockchains to develop an accountable MCS system. By leveraging these technologies, we believe that MCS system will have a good performance in privacy protection and security enhancement.

**Keywords**— Mobile Crowdsensing (MCS), Blockchain, Ethereum, Smart Contracts, security.

# 1 Introduction

Data sensing technology has evolved significantly in recent times as prominence of the data has been realized. As a part of that, fixed-location sensing approaches have become a trend for data collection. Fixed-location sensing approaches include web-based devices which make use of systems like sensors, processors and other hardware devices in order to gather the data from the surroundings analyse the data collected and transmit the processed data. Though these are helpful in reducing and/or eliminating human intervention, these data sensing approaches doesn’t meet the requirements of practical applications as they are constrained due to the less geographical coverage, mobility, high deployment cost and high maintenance cost [1]. Mobile crowdsensing has emerged as a novel data sensing approach to overcome the disadvantages of the above mentioned approach. Mobile terminals are becoming increasingly ubiquitous and crowdsensing has drawn the attention as the next generation sensing technology attributing to its merits that suits the happening informatization era. To be specific, crowdsensing mitigate the deployment cost and provides mobility and scalability.

Mobile crowdsensing is a sensing paradigm in which anyone possessing mobile terminals equipped with the robust sensors is capable of sensing the data of common interest [2-3]. This can also be implied as crowdsourcing of sensory data from mobile devices about anything, at any time and from anywhere. In this technique, intelligent mobile terminals equipped with high sensing and computing capabilities are leveraged to gain knowledge and deliver high quality data. Typically, crowdsensing is comprised of three main parties: requesters, workers and a crowdsensing system. The requesters are sensing task publishers .They issue a sensing task to the system and acquire the sensing data by recruiting the workers. The workers are sensing data contributors. Workers, who are interested in the provided task, take up the task from the system and are responsible for collecting and submitting the relevant data to the crowdsensing system. The system is responsible for dealing with the task operations the requester gets the requested data from the system and once the requester is satisfied with the data provided, the worker will be rewarded for his provided service. In specific, MCS extends the functionality of the existing sensing systems without introducing any additional devices.

With the crowdsensing gaining popularity, it has unfolded opportunities for several applications and provided a proper way of assessing the environment (e.g. weather monitoring, noise, air quality), infrastructure (e.g. road condition, traffic analysis) [4] ,social (e.g. healthcare, travel) and many more. Several technology companies are leveraging this technology to provide the services based on the collected data, some of the famous examples being Google and Uber. The robust sensors incorporated in the mobile devices are utilized to collect the pertinent data on a large scale. However, for these applications to be useful in the real-world scenarios there are certain challenges to be dealt with. The immoral exploitation of crowdsensing applications would cause consequential risks. MCS systems have a high extent of malicious security threats, so it is not easy to avoid tampering of data. The prior mobile crowdsensing systems mostly employs centralized architecture where the system is controlled by a central server while such systems are subject to low reliability and are vulnerable to malicious attacks. With centralized architecture, there is also a need to expend on the central platform. MCS have high concern about the privacy too as the sensitive information of the participant may be leaked to the platform for service processing and it is not easy to guarantee the privacy of the users [5-6]. Furthermore, it is imperative to ensure the privacy in order to gratify the participants for their contribution.

# 2 Related Work

Security of crowdsensing applications has become a major concern of the research association and is highly demanded. While it is crucial to understand and adapt to the new sensing technology, it is equally important to embed trust in a potentially untrusted environment because of the high degree of modularity. Also it is hard to set restrictions on the users and data contributors in the crowdsensing applications. Significant works have been carried out to develop the secure applications based on crowdsensing. We briefly explain the related existing work in MCS in this subsection. In [7] the authors propose a multi blockchain based smart parking system with mobile crowdsensing technology in which the public blockchain is used by data contributors and private blockchain is used by the service providers. The bridge node acts as an interface between the two blockchains. The intuition behind the proposed idea is that a huge number of participants can be accommodated while mitigating the risk of personal exposure. But the proposed schemes have not taken into account the potential security threats of the bridge nodes. CrowdBC [8] is a decentralized framework developed based on blockchain for crowdsensing. It is an Ethereum public test network-based tangible framework which eliminates the need of the conventional centralised system. The proposed framework was successful in mitigating the malicious security attacks like single point of failure by extracting the benefits of blockchains and Ethereum smart contracts. Nonetheless, CrowdBC is not capable of handling the complex situations and hence considered to be impractical in the real time spaces. CrowdBLPS [9] is a location privacy preserving crowdsensing system which integrated the concept of blockchain into crowdsensing, understanding the need of decentralization to get rid of security issues. Leveraging the concept of smart contract, the proposed scheme has also achieved its’ purpose in preserving the privacy of the location and in controlling the data quality by following a two-stage approach. The authors has planned to enhance the reliability of the system to ensure that the system meet the requirements of the real-world.

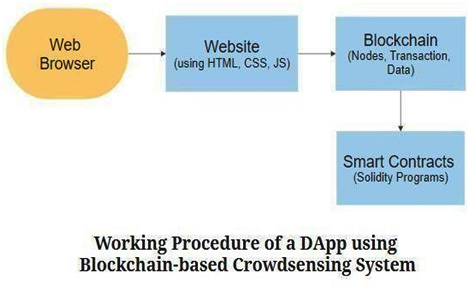
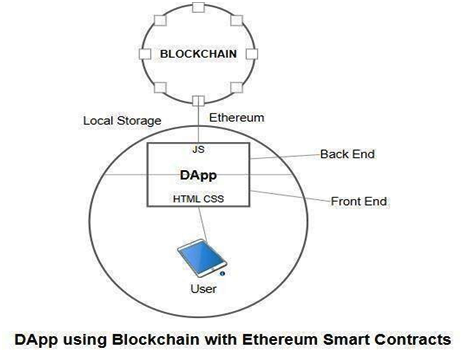
In [10] the authors realized the importance of secure, privacy preserving and incentive crowdsensing services. So they proposed a reliable incentive mechanism for motivating and encouraging the sensing participants to participate in the crowdsensing tasks. They also implemented a privacy preserving scheme by leveraging differential privacy technique through adding the noise data assuming the participants as not totally credible. Then they used homomorphic encryption for securing the sensing data. In future they plan to concentrate on the tradeoff between accuracy and privacy. In [11] the authors propose mobile crowdsensing by leveraging the algorithms of deep learning to enhance the security. To thwart security attacks of the crowdsensing systems, the authors investigated the authentication and protection of privacy based on deep learning. The security approaches based on DL are found to be propitious for providing the Quality of Experience based multimedia services. Nonetheless, it was realized that the authentication based on deep neural networks need high training times and the hardware devices used for DL-based computation are not compatible to the MCS devices thus found to be complicated for the practical implementation of mobile crowdsensing systems. In [12] the authors analysed the challenges and the case studies of mobile crowdsensing systems. They have analysed the fact that the illegitimate task requesters are engaged in keeping the servers busy while needlessly expending energy of servers. The proposed scheme focused on the need to differentiate legitimate and illegitimate tasks and leverages the AI-based machine learning techniques in order to provide a solution to the security vulnerabilities of crowd sensing. However, the proposed work is conducted solely on a location-based and energybased DoS attacks and plan to consider more practical scenario in future.

SenseCrypt [13] is a secure framework that encrypts the sensitive information of the participants of mobile crowdsensing. The proposed framework leveraged the K means algorithm for the clustering of data into sensitive and non-sensitive data. In this the encryption techniques are leveraged in order to secure the sensitive location data of the crowdsensing users. The novelty of the proposed framework is that it has alleviated the cost of computation and communication overhead by leveraging the efficient data compression techniques. However, the paper solely considered the location information as the sensitive data of interest. In [14] the authors proposed a custom-defined location sharing strategy for protecting the privacy of the participant in mobile social networks. They considered the vicinity area as a small bounding rectangle for the proposed scheme where the accuracy of the results is affected by the vicinity area. [15] Focused on providing the incentives in a privacy-preserving manner. They aimed for achieving security and privacy protection in MCS. Ethereum enables extending the functionality of blockchain using smart contracts. Ethereum uses “Proof of Work” which enables the network of Ethereum to accept on the state of information recorded on the Ethereum [16].

# 3 Proposed System

We propose a secure blockchain based crowdsensing framework. The proposed system replaces the conventional triangular architecture by an efficient decentralized framework. The single point of failure attack can be avoided by using the decentralized framework. We leverage the smart contracts deployed on blockchain technology. The task requesters are initially required to deposit certain amount of money through which the enthusiasm of miners and workers can be raised. By this, certain attacks like sybil attack, false- reporting attacks can be prevented efficiently. Thus the proposed system provides an advantage of alleviating the malicious attacks and mitigating the charge of the central platform. The privacy can also be guaranteed by allowing the participants to participate without true identity.

# 4 Working Procedure

In a traditional or conventional centralised system, all users or nodes are connected to a central server or network. However, we propose an approach that makes up for the lack of security in the former system. Instead of connecting directly to a server, we will access our application via a web browser, and we will correlate to the client-side app that we will build, a simple app on a web server. This client-side app won't talk to a web backend and database; instead, it'll directly speak to a Blockchain. We will have code written with Ethereum Smart Contract that will contain our pre-conditions and logic for our use case on the blockchain. That's how a blockchain works fundamentally and how it's different from a traditional web- chain application. A blockchain is a separate peer to peer network of nodes that talk to one another, a distributed network and so different computers talking to one another. We can connect to an individual node on the network; our web appl is doing it here. All nodes participate in running the network; each node contains all code and data on the blockchain. The data on the blockchain has bundles of records. A blockchain is secure and unchangeable, making it robust. All code on the

**Fig.1 Fig .2**

blockchain is on smart contracts, which are the building blocks of blockchain apps. We will write a Smart Contract that will contain all the tasks as set by the requester. Smart Contracts are written in a programming language called Solidity. Blockchain is trustless, we know no one can change it, and thus the app will behave in the same way every time. Blockchains are like a microservice that executes business logic. We will build a client-side application that will talk directly to the blockchain and be deployed on it. The process has been illustrated clearly in **Fig. 1** given. **Fig. 2** depicts a decentralised application is essentially a frontend and a smart contract backend combined in one package. It is also very popularly known as the backbone of Web 3.0. The frontend and backend together work as an integrated utility for the entire smart contract. Each user is connected to the decentralised app and is able to obtain and utilise all its features by operating through the front end for easier accessibility. The backend is connected to the Ethereum smart contract in the form of local storage that incorporates into the blockchain.

**4.1 Working Procedure of Smart Contracts**

The working of a Blockchain maybe split up into the following three categories:

1. Initiating an innovative smart contract
2. Writing a smart contract message to a Blockchain
3. Public key cryptography

# 4.1.1 Initiating an innovative smart contract

Primarily, a public-key encryption infrastructure is used in Blockchain. An initiator coveting to participate in a smart contract hosted on an unauthorised blockchain can use the position to produce a key and write it on the system.

**4.1.2 Writing a smart contract message to a Blockchain**

In a conventional blockchain implementation, when an adequate number of other members or connections arrive at the same outcome, the relevant consent rules determine that the smart contract's information is appended to the Blockchain. Another way is a blockchain's administrator may conclude with it.

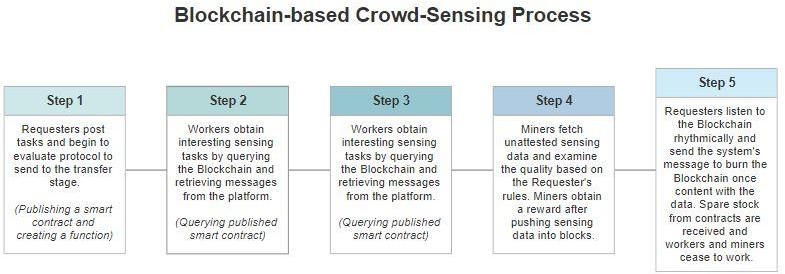
# 4.1.3 Public key cryptography

Decentralised applications or 'DApps' are trustless or peer-to-peer with the differentiating feature that there is no individual server. A DApp uses the blockchain at the core of its data storage, computation and processing, implemented by smart contracts. A traditional website model creates the User Interface for the decentralised application.After the user accesses the Dapp via a web browser, it will connect to the client-side application built on the server. However, the application does not use a traditional web-based database. Instead, the database is the blockchain on which the users connected to the Crowd-Sensing Network perform various tasks, including WifiSensing tasks.

Inside a DApp, a user's account is a digital blockchain key stored on the mobile device's memory. Popular tools that help users manage the accounts and interact with the DApp is MetaMask, an extension for web browsers. These are used primarily by the end-users. A suite of software tools used by developers during the construction of a DApp includes integrated development environments (IDEs) such as Remix IDE and the dominant programming language for Ethereum Smart Contracts called Solidity, as used in this use-case.

# 4.2 Proposed Framework

We introduce smart contracts applications to reward users in our aimed framework, which sensing-task requesters should first launch with particular reserve deposit. There is no centralised stage in the crowd-sensing process in the decentralised system, which we recommend anymore. Alternatively, by implementing blockchain algorithms, the crowd-sensing process is managed by a decentralised system. As proposed in our framework, the Blockchain-based Crowd-sensing process has been explained below in the form of a Flowchart as shown in **Fig. 3** below.



# Fig. 3

**4.3 Implementation of Proposed Model**

The use case chosen for the project is Road-Sensing. Requesters create the intelligent contract 'Road-Sensing', initiate it, and encrypt sensitive data by private keys. Various parameters and features incorporated into the contract include rewards assigned to each task, the required count of data points, the obtained points, the source and destination of the data point collected, and the task's requester. Several states of the requester's assigned tasks are Uncreated, Created, and Inactive.Additionally, the smart contract denotes multiple conditions for the road sensing task that includes- worst, poor, average, good, and excellent. To better depict the crowdsensing process by the use of smart contracts and Blockchain, we have split into three phases as follows-Abbreviations used in algorithms for the sake of simplicity are R- Requester, W- Worker.

The first algorithm, **Algorithm 1**, in the implementation deals with creating our Smart Contract and posting the requester's required sensory tasks. The inputs taken in by the algorithm include rewards assigned for the task, the required data for the task, the source as well as the destination for the task. The output will be the status of the completion of the task. The pre-condition set for the algorithm is if the state is Uncreated and if the assigned task's message value is either greater than or equal to the entire product of the reward allocated. Moreover, the required count and the details specified as per the smart contract will be assigned, and the state of the contract becomes 'Created'. Lastly, the status of the contract is returned along with the TaskInitiated() function and log.

In **Algorithm 2**, the TaskState and the ObtainedCount are parameters of input. The task status, source, destination, and rewards assigned on the road sensing task's registered count are considered output. If the contract's state is 'Created', the worker can only review the task details. When aborting the task, the requester will again check for the contract state's conditions if the amount of obtained count is less than equal to the required count of the data points assigned to the task. If these pre-requisites are met, the contract's state will become 'Inactive'. Finally, the transaction's remaining balance will go to the requester and the AbortTask() function is returned.

**Algorithm 1- Smart Contract Creation and Posting of Tasks by Requester**

**Input:** R*wd*- reward designated for the task, *ReqC*- required data points for the task, S*rc*- the source of data collection of the task, *Dest*- destination location **Output:** *Status-* status of the task

1: Setting of Task

2: **if** State = Uncreated **and** Msgvalue >= *Rwd* \* *ReqC* **then**

3: R = *MsgSender*

4: reward = *Rwd*

5: requiredCount = *ReqC*

6: source = *Src*

7: destination = *Dest*

8: state = Created

9: **return** *Status*

10: **end if**

11: **return** TaskInitiated

The project's final algorithm, **Algorithm 3**, administers the task's committing and designates awards to the workers after data is uploaded and registers successfully in the Blockchain. The source, destination and road condition is taken in as the input, whereas the average road speed, the obtained data count and the task status are the outputs. In order for the worker to commit the task, the following conditions need to be checked- if the contract's state is 'Created' if they obtained count is less than the required count of the source, as well as the destination of the worker's data, submitted matches the criteria set by the requester. An additional condition that is checked, and finally, if the road's average speed is greater than 10. When all the above-stated conditions are satisfied, the obtained count is incremented by 1. Lastly, when the required count becomes equal to the obtained count, the state is finally changed to 'Inactive', indicating the smart contract process's end. The balance of the transaction is transferred back to the requester and the status is returned. The rewards allocated for the task go to the worker, and the DataCommited() function is executed.

**Algorithm 2- Worker Receiving Task Details**

**Input:** Task*State, ObtC* - obtained count

**Output:** *Status-* status of the task, *Src, Dest, Rwd, ReqC*

1: Getting of Task

2: **if** State = Created **then**

3: **return** ViewTask

4: **end if**

5: Aborting of Task

6: **if** State = Created **and** 7: **if** ObtC <= ReqC **then** 8: State = Inactive

9: Transfer balance to R 10: **return** AbortedTask

11: **end if**

**Algorithm 3- Committing of Task and Designating Rewards**

**Input:** S*rc*- the source of data collection of the task, *Dest*- destination

location, *RoC*- Road condition, *AvgS*- the average speed of the road, *ObtC* - obtained count

**Output:** *Status-* status of the task

1: Committing of Task

2: **if** State = Created **and**

3: **if** *ObtC* < *ReqC* **and**

4: **if** LengthCheck(source, *Src*) = **true and**

5: **if** LengthCheck(destination, *Dest*) = **true and**

6: **if** *AvgS* > 10 **then**

7: *ObtC* +=1

8: **end if**

9: **if** *ObtC* = *ReqC* **then**

10: State = Inactive

11: Transfer balance to R 12: **return** *Status*

13: **end if**

14: Transfer rewards to W

15: **return** DataCommited

# Results and Discussion

The main aim is to develop a decentralized crowdsensing system for the road sensing application. We outlined the process of mobile crowd sensing by implementing Ethereum smart contracts on blockchain.

Our proposed Road sensing decentralised application was implemented by leveraging Ethereum smart contracts. We used solidity and javascript as the programming languages and HTML as the markup language . Solidity is a contract-oriented, object-oriented high-level programming language designed for executing smart contracts on blockchain platform. Solidity is native to Ethereum Virtual Machine. In addition, we leverage web3.js which allows in designing the decentralised application that interacts with the Ethereum blockchain. Web3.js is an Ethereum API based on javascript which provides a way to interact with the Ethereum node.

The usage of smart contracts enhances, intensifies and refurbishes the security aspect of the project because of the following grounds-

1. Smart contracts refer to the immutable computer protocols that digitally aid an agreement's execution. They are a speedier, more economical, more agile and more secure way of administering and operating agreements. Hence, the security of the mobile crowd sensing Blockchain system enhances.
2. The setting of pre-conditions using smart contracts also heightens security. Various modifiers and requirements created and emphasized in the intelligent contract demand the worker's authentication before a task commits and rewards distributed.
3. Cryptography is used to secure transactions in smart contracts to prevent attacks like double spending.

# Investigation of the Secure Mobile Crowd-Sensing System using Blockchain

In this section, we briefly examine and investigate our crowdsensing based road sensing system from the security aspect.

1. Before participating in the sensing task, participants making the deposits can help in inhibiting several attacks efficiently.
2. By executing smart contracts, only the ETH accounts of the participants (workers and requesters) are known and thus guarantee the anonymity of the participants.
3. A fundamental defect exists in the execution of the project that arises from the loophole of its execution. Intelligent intruders and attackers may use front-run attacking to seize or withdraw partially the rewards meant to provide the workers. For instance, attackers monitoring the network may fetch the workers' committed data and proffer them instantly. The designated worker might not get the expected reward in its entirety if the data's uniqueness has been considered.
4. However, there have been several solutions that may be successful in avoiding the attack—one of those is proclaiming hoaxes.

The implementation of the project as executed successfully on the Remix IDE platform is presented as follows- Requesters create the road sensing task by specifying the input parameters like rewards, number of data required, source and destination and then call setTask() function. The execution of task publishing (setTask) phase is being depicted in the **Fig 4**. Requesters have the option to view the amount of data received so far by calling getDataCnt() function. The data count is incremented everytime the worker submits the appropriate data. **Fig 5**. depicts the execution of the function getDataCnt() function. For submitting the collected data, workers make a call to commitTask() function. Once the preset conditions are met, the worker submits the data and the data count will be incremented by one on successful submission.

**Transaction costs** deal with the expense of transmitting data, and a total of three components account for the transaction cost of the Blockchain's function. They include-

* + transaction's base charge
  + the smart contract's disposition charge
  + the expense of the contract's zero and non-zero bytes of data

# 

# Fig 4: Execution of setTask() function Fig 5: Execution of getDataCnt() function

**Execution costs** include the value of calculation processes, which are performed as a completion of the transaction.As illustrated in **Fig.6**, the comprehensive report of the transaction expenses for each task function used in the project has been portrayed. The various task functions used were-- setTask, getDataCnt(), getTask(), abort(), and commitTask(). An average for all the costs taken into account is provided for a better estimate of our project's prospects. The transaction costs have been given in terms of the gas being spent on each function.

**Fig 6: Graph- Analysis of Transaction Costs for each Task Function**

Thus, the total cost incurred for the data calculations is sustained, and hence, the proposed system's efficiency is enhanced. However, the transaction costs may be enhanced further by reducing the base cost of the transactions of each of the task functions.The objective of this project is to develop a decentralised crowdsensing system that replaces the conventional triangular architecture. We also aimed at leveraging the Ethereum smart contracts by which a successful trade-off between data quality and security preservation can occur. Hence, both these objectives are satisfied by the execution of our project and proposed model.

# 6 Conclusion

In conclusion, we proposed an architecture for a decentralised application of the Mobile Participatory Crowdsensing paradigm based road sensing system. We realized that the conventional crowdsensing architecture suffers from single point of failure threat. We leveraged Ethereum Smart Contract deployed on Blockchain technology. The proposed architecture exercises smart contracts that enable crowdsensing providers to proffer their requests and run an action to ascertain the most fitting mobile users engaged in administering the crowdsensing tasks. The framework also employs a reward system for workers and deals with accessing the crowdsensing provider by successfully regulating blockchain. The proposed architecture implements an incentive-based optimal option for users who handle each crowdsensing task. Hence, we achieve the objectives of the project and enhance the focused parameters of the problem. As a future study, we focus on fully protecting the privacy for the application to meet the real world requirements. We also consider authenticating the data contributors to avoid data fallacies for getting the reliable data.

# References

1. J. Huang et al., "Blockchain-Based Mobile Crowd Sensing in Industrial Systems," 2020 in *IEEE Transactions on Industrial Informatics*, vol. **16**, no. 10, pp. 6553-6563
2. L. K. Shien and M. M. Singh, “A Secure Mobile Crowdsensing (MCS) Location Tracker for Elderly in Smart City,” 2017 in *AIP Conference Proceedings*, vol. **1**, AIP Publishing, p. 020085.
3. M.ArulPrakash, "Dynamic Evolutionary Information Diffusion over Mobile Social Networks”, 2016,

*ARPN Journal of Engineering and Applied Sciences*, **11**(19), pp. 1145711464.

1. M.ArulPrakash, Kamal, A., Manisha, A., "QR-Code scanner based vehicle sharing”, 2018, *ARPN Journal of Engineering and Applied Sciences*, **13**(10), pp. 3441-3448.
2. C. Luo et al., "Predictable Privacy-Preserving Mobile Crowd Sensing: A Tale of Two Roles," 2019, in

*IEEE/ACM Transactions on Networking*, vol. **27**, no. 1, pp. 361-374

1. Toch, Eran, Crowdsourcing privacy preferences in context-aware applications, 2014, *Personal and Ubiquitous Computing*. vol. **18**. 10.1007/s00779-012-0632-0.
2. Kim, M.; Kim, Y. Multi-Blockchain Structure for a Crowdsensing-Based Smart Parking System, 2020

*Future Internet*, vol. **12**, 90.

1. M. Li et al., "CrowdBC: A Blockchain-Based Decentralized Framework for Crowdsourcing,", 2019, in

*IEEE Transactions on Parallel and Distributed Systems*, vol. **30**, no. 6, pp. 1251-1266

1. S. Zou, J. Xi, H. Wang and G. Xu, "CrowdBLPS: A Blockchain-Based Location-Privacy-Preserving Mobile Crowdsensing System,", 2020, in *IEEE Transactions on Industrial Informatics*, vol. **16**, no. 6, pp. 4206-4218
2. Jinbo Xiong, Rong Ma, Lei Chen, Youliang Tian, Li Lin, Biao Jin, "Achieving Incentive, Security, and Scalable Privacy Protection in Mobile Crowdsensing Services", 2018, *Wireless Communications and Mobile Computing*

[11] L. Xiao, D. Jiang, D. Xu, W. Su, N. An and D. Wang, "Secure mobile crowdsensing based on deep learning,", 2018, in *China Communications*, vol. **15**, no. 10, pp. 1-11

[12] Y. Zhang and B. Kantarci, "Invited Paper: AI-Based Security Design of Mobile Crowdsensing Systems: Review, Challenges and Case Studies," 2019 *IEEE International Conference on Service-Oriented System Engineering (SOSE)*, San Francisco, CA, USA, 2019, pp. 17-1709

1. Pius Owoh N, Mahinderjit Singh M. SenseCrypt: A Security Framework for Mobile Crowd Sensing Applications. *Sensors (Basel)*. 2020; 20(11):3280
2. Tao Peng, Jierong Liu, Guojun Wang, Qin Liu, Jianer Chen, Jiawei Zhu, "A User-Defined LocationSharing Scheme with Efficiency and Privacy in Mobile Social Networks", 2020, *Scientific Programming*, vol. **2020**, Article ID 7832875, 13 pages

[15] S. Gisdakis, T. Giannetsos and P. Papadimitratos, "Security, Privacy, and Incentive Provision for Mobile Crowd Sensing Systems,", 2016, in *IEEE Internet of Things Journal*, vol. **3**, no. 5, pp. 839-853

[16] K. AlJemy, M. AlAnazi, M. AlSofiry and A. Baig, "Improving IoT Security Using Blockchain," *2019 IEEE 10th GCC Conference & Exhibition (GCC)*, Kuwait, Kuwait, pp. 1-6