Fifa Ticketing System using Blockchain Hyperledger Sawtooth Platform

Milton Bron Lima, R. M. Banakar

Abstract: In this paper we present an approach to investigate how Blockchain Hyperledger Sawtooth platform can be used in ticketing system to process a secure transaction over a network environment. A framework for a ticketing industry has been proposed to overcome the transactional issues between distributors, vendors, and the customers. A discrete three-layer architectural model is presented that is suitable for the application design development. The Blockchain architecture allows us to maintain the list of records contained in the distributed data base on the network and the validator validates the transaction allowing the transaction processor to authorizes the nodes. The outline of Docker compose demonstrates on how to start the transaction processor from the console for the ticketing application. The methodology indicates the use of algorithm to generate a unique key by importing the python software development kit in our design implementation Finally, the paper also provides the detail analysis of how the transactions are initiated, retrieved, and exported to view the complete block of encrypted data and achieve safe and secure transaction between two parties over a network.

Keywords : ***Blockchain, Hyperledger Sawtooth, Smart Contracts, Python SDK, Validators, Transaction Processors***

I. INTRODUCTION

The ticketing sales industry is currently worth $80 Billion in annual ticket sales worldwide and it remains to grow year over year. The increased internet usage and growth usage of mobile applications for booking live events, and movie tickets via mobile applications are likely to effect growth. Customers have revealed that they are ready to pay additional charges to avoid waiting in long queues, which completely changed the global market. When issuing ticket went digital, ticket dealer went from reversing tickets outside of venues to digital businesses that make up a $12 Billion global resale market. The present ticket market splits entertainers and their fans, increases the cost of tickets to your favorite show when purchased unverified tickets from third party channels while performers, teams, organizers, and venues fail to receive revenue and data from the secondary markets. The ticket bookings through applications on the smartphones are expected to increase day by day. Mobile devices have become the ideal device for purchasing tickets. The advancement and cost effective high-speed 4G connectivity have been gaining popularity and allowing customers to book tickets anywhere and anytime.

Revised Manuscript Received on April 11, 2020.
* Correspondence Author

Milton Bron Lima*, M.Tech, Electronics and Communication Engineering, BVB College of Engineering and Technology, Hubli Karnataka, India
Dr. Rajeshwari M. Banakar, Professor in BVB Engineering College Hubli

As compared to the traditional booking via laptops and computers, mobile booking is considered as easy and effective way of booking. The software developers have been developing user-friendly applications with frequent updates and offers to enhance customer experience.

Many events have started accepting mobile-tickets and electronic-tickets to avoid waiting time in the queue and wastage of printing paper. These printed tickets are equipped with QR codes and barcodes which are scanned by the customer for authentication. The overall method has turned out to be secure, reliable, and fast. In addition, this increases convenience for both service provider and the consumers. With the proposed work, we believe the solution to this problem is to offer a safe and transparent transactions between FIFA fans and performers, teams, organizers, and venues. The use of smart contracts enables the performers, teams, organizers, and venues to decide the way they like to buy and resell their tickets. By this they can track each tickets and the revenue generated from all other source of transaction. The customer can now purchase confirmed FIFA tickets across different source without any risk. This will allow the tickets distribution to go through the trusted sponsors while providing better customer service. Section 2 discusses the detailed outline of Blockchain and the Hyperledger platform. Section 3 highlights how the transaction is validated in Hyperledger Sawtooth network.

V. BLOCKCHAIN HYPERLEDGER PLATFORM

Fig. 1. shows the block diagram of the Blockchain architecture. Blockchain is a state-of-the-art revolutionary technology. It is essentially a distributed database, that contains list of records called blocks. These blocks contain a time stamp that are linked to previous block. The part of the block header is Merkle root and it ensures that the network receives transactions securely. The idea here is to create a database that is decentralized record keeping system [1]. Here everyone who participates in maintaining the record keeping system validate the authenticity of those transactions. This means, there is no centralized authority [2], [3]. Now we have companies that own these database that collect user information. Eg the bank records transaction between people who have money. In this case the database is distributed among different nodes and is controlled by the bank. Every time a transaction takes places in the Blockchain, these nodes validate the transaction using asymmetric code encryption [4]. Blockchain enables the exchange of digital assets over the internet without any intermediators acting as third party trust.

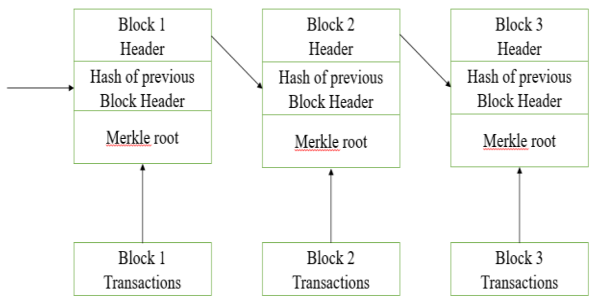

Fig. 1. Blockchain Architecture [5]

Hyperledger is a project under Linux foundation. Like other projects under Linux, Hyperledger is an open source development project where anyone can use to develop as a software and a platform [6]. Hyperledger can be thought of a software that enables developers across to develop Blockchain based solutions for their business. Currently, Blockchain is more accessible to the world. The developers have started to test the interaction between the applications and secure Blockchain network [7], [8]. With scalability, the network takes exhaustive measure to ensure security and above that any transaction that has a measure of confidentiality and privacy attached to it cannot be executed on the Blockchain due to its public nature [9]. Every ledger on the network gets updated and confidentiality is lost. On the Hyperledger network the participants that are directly associated with the transaction are alerted and the ledger is updated. This allows to maintain privacy and confidentiality. In the Hyperledger network, the consenter node is responsible for running consensus algorithm on the network and validates every transactions and decides whether to add on the ledger. By using Hyperledger Sawtooth platform we can manage the transactions enabling ownership to be transferred on the Blockchain as per Smart Contracts [8].

VI. VALIDATING TRANSACTION IN HYPERLEDGER SAWTOOTH

A Hyperledger Sawtooth is a modular platform for building, deploying and running distributed ledgers. This uses consensus algorithm called 'Proof of Elapsed time' (POET) which uses minimum resource consumption. One of the feature of Hyperledger Sawtooth is the demarcation of the core system and application layer of the network. This allows the developer to deploy new applications easily. It also introduces advance power scheduler which allows to split into sections in parallel flows [10], [11]. Finally, it provides the naming consensus making this algorithm more scalable. Hyperledger sawtooth is specially designed for enterprise network and it resides on a private network behind the firewall on the converged network architecture. This can be configured in the form of permissioned or permissionless access. The new feature includes the metrics to submit batches and transactions, processing the transaction, modifying the dispatcher queue sizes and thread pools. The internal metric library is designed to match logging API and supports backwards compatible upgrades.

Fig. 2. shows a Hyperledger Sawtooth Network. The network contains two Transaction Processors and two validator nodes. The frontend client sends the transaction to the backend REST-API. The transaction processor will then move the transaction to the suitable handler function. In the Hyperledger Sawtooth environment, the Transaction processor controls the Business logic which allows or restricts the transactions from being added to the state. Each node within the Hyperledger Sawtooth network runs a transaction processor [12]. When a client sends a transaction, the transaction processor validates the transaction and apply the changes to the state which will be placed into the next block. This transaction processor processes received transactions submitted by authorized nodes.

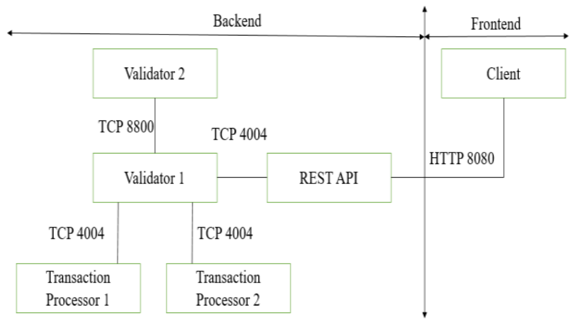

Fig. 2. Validating Transaction in Hyperledger Sawtooth [13]

Transactions in Hyperledger Sawtooth can be described in terms of three components:

- Header: The header contains metadata and the hash of the payload. If any changes are made to the transaction header, the signature will not validate.
- Signature: The signature allows the nodes in the network to authorize the signer.
- Payload: Transaction processors use the payload to change state.

The purpose of the clients in the Hyperledger Sawtooth is to submit transactions to the validator and query the database. The validator then mediates access to the database and validates the transaction. The transaction processor holds the business logic of the payload. Each transaction has a payload and the payload is sent to the validator. The validator validates the cryptographic function and sends back to the transaction processor with the payload. As of today, the developers can write codes in wide range of languages [14]. From the frontend, the client code in the Hyperledger Sawtooth can be written in SDKs such as Python, Javascript, Go and Rust. Also, in the backend the transaction process can be written in SDKs such as Java, C++, Python, Javascript, Go, and Rest. The purpose of the client is to package and sign transaction, encode the transaction and decode the processed data from state. The client in this environment can be apps, browser or CLI that can talk to REST API or built-in APIs.

Transaction processor are like smart contracts that encapsulates the business logic of the transaction family. The transaction processor will first register with the validator [15]. When a new transaction appears, the validator will call an apply method and is passed to the transaction. The transaction processor responds with get/set value to the validator.

A transaction family includes the following components:

- Clients – Creates the transaction.
- Data model – Data model for state defines what addresses to be used, what are the different entities in the state look like.
- Business logic – Business logic to the payload decides what should be saved in the state.

The transaction validators validate the transaction blocks and ensure that transactions result in the state changes that are constant across all the participants in the hyperledger sawtooth network. The validator node will verify to ensure that the transaction signature is valid, but further logic can be applied in the transaction processor to check more detailed transaction requirement [16], [17]. The transaction processor is provided with the Python software development kit. To start up the transaction processor run the following command from your console **"docker-compose -f <filename>.yaml up"** The register attempt on the transaction processor output should display 'OK'. To bring the docker container down, issue the following command from your console **"docker-compose -f <filename> down"**. The transaction processor is now started and stopped from your console. Now, to process the transactions, we must implement the transaction handler functions [18], [19]. The transaction handling in the validator processes the parallel scheduling and looks for the addresses in state that a transaction wants to read, modify and organize the schedule for handling transactions and serializing the transaction.

To implement a transaction handler function, we must receive and unpack the transactions. In the Hyperledger Sawtooth environment, the transaction processors are called by means of the apply method. When a transaction processor receives a transaction, the transaction is executed by the apply function [20], [21]. The apply function will then call another function called unpack transaction and based on the transaction type this will determine the right handler function to call. To apply the transaction, the handler function must be determined as suitable. When one unpacks a transaction, the payload is extracted, this stores the transaction information and the authorized public key. The current data stored on the Blockchain displays the State [22], [23]. When a data is inserted an address is assigned so that the data can be retrieved quickly. These addresses can be created in variety of ways and the length of addresses are limited to 70 characters. To implement a logic for the transaction processor, the IP addresses from the static backend that we query must be changed [24]. To process the transaction, the transaction will reach the transaction processor and this will apply state to the transaction.

The function should check for the duplicate project name and for the complete payload. It can only create a project for inserting into the state if these test passes. This will help when the client will create tasks in the project. This project will contain list of public keys of the clients authorized to modify the project. To create a task and process the transaction, one must check if the relevant information is complete within the payload and verify if author who owns the transaction is authorized to create a task in the project [25]. Then, check if there is a duplicate task name in the list. If not, then create a new task name to the list. Now, the task object is created and the task can be added at its address to the container. To edit a task, description, and set in state, one must check the authorization and payload integration.

VII. DESIGN IMPLEMENTATION

As shown in Fig. 3, an event organizer is responsible for making of any events for the benefit of business. They are proficient and specialized in their business that they serve with wide variety of skills, qualities, and abilities to fulfill the demands of the business. The role of event organizer is challenging for anyone with true passion for creativity and world of business. The advantage of event organizer is that the business will benefit in increasing the sales and market share. For an event to be successful, the event organizer should meet the demands and expectations of the business. Regardless of the nature of the event, the venue must be decided by the event organizer in order the event to take place. In our case, the organizer must decide a stadium. Once all the arrangements are complete, the event organizer is responsible keeping the stakeholders engaged in the event and to advertise the distributors that the event will take place.

A business organization who is involved in buying the tickets from the event organizer is called a distributor. The event organizer is responsible to provide an update to the distributors on what they would expect from the event. They are also responsible to arrange and let the distributors know at least 15 days prior before the tickets go on sale and available for purchase. A distributor purchases the ticket from the event organizer for a minimal price and charges more than the cost purchased for the event organizer. Normally, the distributor purchases the tickets for himself and sell them to the vendors from their stock. The vendors then buy the tickets from the distributors and sell them to the customer. A distributor is a mediator between organizer and vendor.

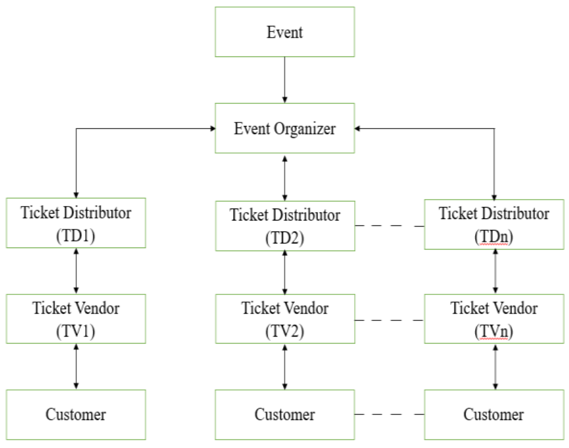

Fig. 3. Design Implementation

Fig. 3. shows the block diagram of the ticketing system using Hyperledger Sawtooth platform. Using the docker compose file we have developed a multi-container Hyperledger Sawtooth application for the ticketing system. We have imported Python SDK in our implementation and this uses ECDSA algorithm to generate a key [26], [27], [28]. The proposed methodology uses the following steps to perform a secure transaction.

- The ticket distributors contact the event organizer for the number of ticket that they wish to purchase.
- The event organizer processes this transaction and the distributor acts as a client that initiates a transaction requesting for 'x' number of tickets and this transaction is authorized using the transaction processor and placed into the blocks.
- The client can communicate directly with the validator core or through the REST API on the network.
- The transaction can be ordered in a batch for complex state change. The batch contains the header, public key, and title of the application.
- Here, the event organizer and Distributor are directly associated with the transaction and the ledger is updated.
- The client container is required to connect the validators or their networks. We must connect to the client container using a terminal window and verify if the validator is active.
- If the validator is active, open a new window to verify the connectivity from the local host system. The transaction between the participants is submitted using the batch submit parameter.

VIII. RESULT AND DISCUSSION

The frontend client code in Hyperledger Sawtooth is written using Python SDK. The transactions between two parties are initiated and then the transactions are retrieved by first identifying the block from the block list. This list displays the batch detail, block id and the number of transaction occurred between the participants. The block id is then exported using 'export' command. Finally, execute the 'Sawtooth block show' command to display the details of the Sawtooth block. The block contains the encrypted data of the transaction such as authorized signer public key, payload details, block id etc. ensuring safe transaction between two participants on the network.

Table- I: Command and Result

Command	Result
#sawtooth block list	"batches": 1, "block_id": "7a3c1e6acb351c98ebd140c49cab0c37b469a04216e3b210a3b87ba4e8 b3ed28cde0fb06dfd55c985e739dec592ac43dea5397ed76632c53abd59 c21ecd6ac5" "num": "0", "signer": "1c45108ca53629efde3b87c4c9126c3de09ac47c65d601316259c13eca 49b4a957b1" "txns": 1
Export	"7a3c1e6acb351c98ebd140c49cab0c37b469a04216e3b210a3b87ba4e 8b3ed28cde0fb06dfd55c985e739dec592ac43dea 5397ed76632c53abd59c21ecd6ac5"
#sawtooth block show $BLOCK_ID	"7a3c1e6acb351c98ebd140c49cab0c37b469a04216e3b210a3b87ba4e 8b3ed28cde0fb06dfd55c985e739dec592ac43dea5397ed76632c53abd5 9c21ecd6ac5" "previous_block_id": "0000000000000000", "signer_public_key": "fa526ab8f4e03f9bcbde532d8242e85be3d0466d694f80b46eca0583b8 f6c24828", "header_signature": "f89eec31421060390c5972bcfdfef273006735059d0cedc1f68566cd78 6093f778b21cdc22721185a4eca17e5abd0626ff5628be786ccbc774f09 20ea12d307"

IX. CONCLUSION

This paper presents a framework to the customer, vendor, and distributor who wish to perform ticket distribution safely and securely across different participant over a large cluster network using Hyperledger Sawtooth platform. We use Python SDK for implementing our secured network for data transmission. The SDK uses a ECDSA algorithm to create a key that is unique to each transaction. The use of the current Blockchain Hyperledger Sawtooth platform efficient distribution of ledgers and smart contracts secure the transaction on the network and is more robust as each transaction is authorized before it is processed.

A unique method of ticket distribution is discussed, where transaction between customer, vendor, and distributor is kept confidential as the key generated are different for each transaction over the
larger network. With the use of transaction validators, we have validated each transaction blocks and the validator node will authenticate and ensure that it signs each transaction and authorize for further processing. The design implementation presented in the paper is verified for a transaction over a simple network environment. Future work is focused on addition of more number of distributors and vendors to perform transactions on a larger cluster.

REFERENCES

1. Si Chen, Jinyu Zhang, Rui Shi, Jiaqi Yan and Qing Ke, "A Comparative Testing on Performance of Blockchain and Relational Database: Foundation for Applying Smart Technology into Current Business Systems," Nanjing University, Nanjing Jiangsu 210023, China, vol. 1, May. 2018, pp. 1-5.
2. G. Tien Tuan Anh Dinh, Rui Liu, and Meihui Zhang, "Untangling Blockchain: A Data Processing View of Blockchain Systems," Member IEEE, vol. 1, Jul. 2018, pp. 1366 – 1385.
3. Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso , Paul Rimba, "A Taxonomy of Blockchain-Based Systems for Architecture Design," School of Computer Science and Engineering, UNSW, Sydney, Australia, INSPEC Accession Number 16913008, vol. 1, Apr. 2017, pp.1-4.
4. Michael Mainelli, and Alistair Milne, "The Inpact and potential of Blockchain on the securities Transaction Lifecycle," SWIFT Institute, May. 2016, pp. 1-6.
5. Block hashing algorithm [Online]. Available: https://en.bitcoinwiki.org/
6. Nitesh Emmadi, R.Vigneswaran, Srujana Kanchanapalli, Lakshmipadmaja Maddali, and Harika Narumanchi, "Practical Deployability of Permissioned Blockchains, TCS Innovation Labs," Hyderabad, India, vol. 1, Jul. 2018, pp. 229 – 243.
7. Hyperledger, Blockchain technologies for business [Online]. Available: https://www.hyperledger.org/
8. Qinghua Lu and Xiwei Xu, "Adaptable Blockchain Based Systems," The Commonwealth Scientific and Industrial Research Organization, vol. 1, Nov. 2017, pp. 21–27.
9. Paulus Andreas Corten, "Implementation of Blockchain Powered Smart Contracts in Governmental Services," Delft University of Technology, vol. 1, Mar. 2018, pp.1-4.
10. Burkhard Stiller, "Design and Implementation of a Smart Contract Application," Master Thesis, Florian Schüpfer Lucern, Switzerland, Aug. 2017pp. 1-25.
11. Q. Lin, P. Chang, G. Chen, B. C. Ooi, K. Tan, and Z. Wang, " Towards a non-2pc transaction management in distributed database systems," in Proceedings of ACM International Conference on Management of Data (SIGMOD), San Francisco, CA, USA, Jan. 2016, pp. 1659– 1674.
12. C Mohan, "Tutorial: blockchains and databases," IBM Almaden Research Center, Aug. 2017, pp. 2000 – 2001.
13. Sawtooth Hyperledger for business [Online]. Available: https://sawtooth.hyperledger.org
14. T. T. A. Dinh, J. Wang, G. Chen, L. Rui, K.L. Tan, and B. C. Ooi, "Blockbench: a benchmarking framework for analyzing private blockchains," in SIGMOD, vol. 1, Dec. 2017, pp. 1-4.
15. K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. Gun, "On scaling decentralized blockchains," in Proceedings of 3rd Workshop on Bitcoin and Blockchain Research, vol. 1, Jan. 2016, pp. 1-4.
16. M. Swan, "Blockchain Consensus Protocols," Bitcoin Meetup, May 2015.
17. "Distributed Ledger Technology: Beyond Block Chain," UK Government Chief Scientific Advisor Jan. 2016. [Online]. Available: https://www.gov.uk/government/news
18. Kenji Saito, and Hiroyuki Yamada "What's So Different about Blockchain? - Blockchain is a Probabilistic State Machine," IEEE 36th International Conference on Distributed Computing Systems Workshops, Japan, Jan. 2016.
19. Thanh Bui, and Tuomas Aura, "Application of Public Ledgers to Revocation in Distributed Access Control," Cornell University, Aug. 2016, pp. 1-5.
20. M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack A global naming and storage system secured by blockchains," In USENIX ATC, Santa Clara, CA, Jan. 2016.
21. T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto,"BigchainDB: A scalable blockchain database," white paper on BigChainDB, May. 2016, pp. 1-6.
22. F. Tian, "An agri-food supply chain traceability system for China based on RFID & blockchain technology," International Conference on Service Systems and Service Management IEEE, Jun. 2016, pp. 1-6.
23. M. Swan., "Blockchain: Blueprint for a New Economy," O'Reilly, US, Dec. 2015, pp.1-6.
24. Okada, H., Yamasaki, S., & Bracamonte, V, "Proposed classification of blockchains based on authority and incentive dimensions," 19th international conference on advanced communication technology, vol. 1, Dec. 2017, pp. 593–597.
25. D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm," in 2014 USENIX Annual Technical Conference (USENIX ATC 14), Philadelphia, PA, USA, Dec. 2014, pp. 1-6.
26. M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," Applied Innovation 2, vol. 1, May. 2016, pp. 6–10.
27. A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Čapkun, "On the security and performance of proof of work blockchains," In ACM CCS 2016, Vienna, Austria, Dec. 2016, pp.1-6.
28. Li Z, Kang J, Yu R, D Ye, Q Deng, Y Zhang, "Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things," IEEE Transactions on Industrial Informatics, Jan. 2017 pp. (99):1-1.

AUTHORS PROFILE

Milton Bron Lima Received Diploma degree in Electronics and Communication from RN Shetty Polytechnic, Sirsi, Karnataka, India in 2004, B.E. degree in Electronics and Communication Engineering from Bangalore College of Engineering and Technology, Bangalore, Karnataka, India in 2008 and M.Tech degree in Electronics and Communication Engineering from BVB College of Engineering and Technology, Hubli Karnataka, India in 2013. He is a Research scholar at KLE Technological University, Hubli, Karnataka, India and presently working as Technical Lead at a reputed Telecom company in Bangalore, India. His current areas of research include Blockchain, Data Networking, Wireless Networks, and Routing and Switching. He has presented three research papers at National and International Conferences.

Dr. Rajeshwari M. Banakar Received B.E. degree in Electronics and Communication Engineering from Karnatak University India in 1984 and M. Tech in Digital Communication from Regional Engineering College, Surathkal, Karnataka. She has a couple of years experience in Indian Space Research Organization (ISRO). She completed her Ph.D. in the area of Low Power Application Specific Design Methodology from IIT Delhi in 2004. Presently she is working as Professor in BVB Engineering College Hubli **Karnataka**. She is the member of ISTE, IETE, MIE and IEEE scientific and professional societies. Prof. Banakar's work on Low power scratch pad memory organization presented at CODES 2002 has been rated as highly cited work in short period of four years (2002-2006). She has won the best paper award in 2004 at International conference - MobiComNet, India. She has also contributed to a chapter on Turbo Equalizer in Handbook of Research in Mobile Business (Australia), 2006. In Cadence design contest 2009, she is the recipient of runner up award. Her current areas of research include SOC, VLSI Architecture, WCDMA and Adhoc networks. She has published over 55 research papers.