A STEP TOWARDS BUILDING TRUSTWORTHY WIRELESS SENSOR NETWORK

Himanshu Goyal

A STEP TOWARDS BUILDING TRUSTWORTHY WIRELESS SENSOR NETWORK

Thesis submitted

in partial fulfillment of the requirements for the award of the

Master of Technology

in

Computer Science & Engineering

(under the Dual-Degree Programme)

by

Himanshu Goyal

17CS02011

Under the supervision of

Dr. Sudipta Saha

School of Electrical Sciences

Indian Institute of Technology Bhubaneswar

Orissa, India - 752050

Indian Institute of Technology Bhubaneswar
School of Electrical Sciences

APPROVAL OF THE VIVA-VOCE BOARD

May 9, 2022

Certified that the report entitled **A STEP TOWARDS BUILDING TRUSTWORTHY WIRELESS SENSOR NETWORK** submitted by **Himanshu Goyal** (17CS02011) to the Indian Institute of Technology Bhubaneswar in partial fulfillment of the requirements for the award of the Master of Technology in Computer Science & Engineering under the Dual-Degree Programme has been accepted by the examiners during the viva-voce examination held today.

Rajat Subhra Chakraborty

(Supervisor) **(External Examiner)**

(Internal Examiner 1) **(Internal Examiner 2)**

DECLARATION BY THE SCHOLAR

I certify that:

- The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.
- The work has not been submitted to any other Institute for any degree or diploma.
- I have followed the guidelines provided by the Institute in writing the report.
- I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- Wherever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
- Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.
- In case if something is used and it belongs to you, let author know about it. He will either attribute you or take it out from here.

Goyal

Himanshu Goyal

17CS02011

CERTIFICATE

This is to certify that the thesis entitled **"A STEP TOWARDS BUILDING TRUSTWORTHY WIRELESS SENSOR NETWORK"**, submitted by **Himanshu Goyal** to Indian Institute of Technology Bhubaneswar, is a record of bonafide research work under my supervision and I consider it worthy of consideration for the award of the Master of Technology in Computer Science & Engineering under the Dual-Degree Programme.

Dr. Sudipta Saha
Supervisor
School of Electrical Sciences
Indian Institute of Technology Bhubaneswar

Place: IIT Bhubaneswar
Date: May 15, 2022

ACKNOWLEDGEMENT

First of all, I want to express my deep and heartfelt gratitude to my respected guide and mentor, Dr. Sudipta Saha, for his evergreen blessings, valuable advice, support, well wishes, and encouragement all along this journey. It has been a great learning experience doing research with him. Without his unwavering support, this work would not have been actually in the light of daylight. He has been a fantastic mentor, and I will miss all the amicable conversations we had over the past years. I am also appreciative of the Decentralised and Smart Systems Research Group (DSSRG) members for their assistance and insightful suggestions during the implementation process. I am very grateful to Chandra Shekhar, Jagnyashini Debadarshini, Abhishek Mishra, and Sourabha Bharadwaj for their assistance in understanding the software and hardware infrastructure required for experimentation purposes. Moreover, I am happy to be working with Krishna Kodali on one of our research projects and value his constructive criticism. I relished every amusing interaction we had while working on this project.

I would like to extend my acknowledgment to my faculty, friends, my juniors and seniors for their never ending support, care and love which made the journey in IIT Bhubaneswar very memorable. This journey is a learning curve for me not only in academic aspects but also in other domain of life. I rejoiced every moment of this journey with them.

Last but not the least I sincerely believe that this arduous journey would not have been possible without the help of my parents and grandparents. I am greatly indebted towards them. Their constant support has been quite strong throughout the last couple of years, and I hope it continues forever. Finally, I would like to take this opportunity to embrace my elder sister Priyanka Goyal, for her regular kind support. She had made me learn the true meaning of perseverance and hard work. The words of wisdom that I received from her are genuinely indisputable and have helped me significantly sail through difficult phases of my life so far. Hence, again a special thanks to all who extended their care and supportive hands in every aspect of my life in these years during the journey of Dual-Degree program at IIT Bhubaneswar.

Place: Indian Institute of Technology Bhubaneswar Himanshu Goyal
Date: May 15, 2022

Abstract

True smart living is fundamentally determined by the available services from the smart systems such as smart-city, smart-building, smart-home, intelligent transportation systems, precision agriculture, intelligent environmental monitoring, smart-grid, and many others. Internet-of-Things (IoT)[1]can be considered the primary enabler of such smart living for humanity. Every such intelligent system fundamentally thrives upon massive decentralized coordination and cooperation among many independent devices. The performance of an IoT system, thus, significantly depends on how well the devices can talk to each other and how smoothly they can coordinate together and execute the necessary tasks. However, the growing concerns of several attacks[2, 3, 4, 5, 6]ranging from data integrity to data privacy have raised severe concerns about adopting IoT-based smart systems. These attacks could potentially cause a serious threat to the end-users thus affect the quality of life and service. Therefore, it has become more important for these systems to be secure, as well as reliable. Moreover, in certain cases, such systems also need to be able to preserve privacy too. However, because of the participation of many devices having low processing capability as well as high energy constraints, unlike traditional systems, it becomes much more difficult to achieve these goals. Therefore, in this work we show how we can build a *"Trustworthy IoT/WSN network"* comprising of low-power devices that is robust against intentional/non-intentional device failures and can also provide the service of carrying out computation over the data held by a set of devices privately and securely.

In this work, we mainly focused on achieving Byzantine fault tolerance in IoT networks and privacy-preserving data-aggregation among the participating nodes. Existing solutions for IoT/WSN systems mostly assume simple non-Byzantine node failures which is not enough to solve the problem. To combat the presence of smart devices with malicious intention, Byzantine fault tolerance support is highly essential in building trustworthy decentralised system. Byzantine fault tolerance has not been addressed much in the context of IoT/WSN because of its inherent requirement of extensive data sharing among the nodes. In this work, we approach to bring a solution to the problem using multi-flooding under time-slotted form. In particular, we show how the the well-known *Practical Byzantine Fault Tolerant* (PBFT) consensus strategy can be remodeled in an efficient form that is suitable for use in IoT/WSN systems.

Use of smart-systems result in increased ease in our day-to-day living. However, being assisted by a smart-system indirectly implies being tracked by the ubiquitous sensors attached with the smart IoT-devices that compose a smart-system. In order to fulfil the goals, the IoT-devices in a smart-system collaborate with each and for which they need to exchange their data. This in turn lead to an enhanced scope of passive learning about personal activities of the users of the smart-systems by the adversaries. Usual way to protect such breach of privacy is to make the smart-systems depend more on the aggregated data without revealing the source/origin of the individual components. However, existing solutions for such privacy-preserving data-aggregation either exploits heavy computation in the devices or introduce quite high degree of additional communication overhead. Both of these issues make them inappropriate for low-power IoT-edge system. In this work, we show an approach where *concurrent-transmission* (CT) is used to fulfil the communication demand and realize the privacy-preserving data-aggregation for low-power systems in a fruitful way.

Keywords: Privacy-Preserving Data aggregation, Threshold Cryptography, Practical Byzantine Fault tolerance, Broadcast Communication, Data Sharing: One-to-all, Many-to-Many, All-to-All,etc.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of the area

The contemporary period has seen the rise of Internet-of-Things (IoT) devices to serve people with various applications such as smart metering, urban traffic monitoring, weather monitoring, and so on. Low Power Wide Area Networks (LPWAN) technical advancements provide promising solutions for effectively realizing these existing applications while also opening the door to sophisticated applications such as Edge Computing, Machine Learning, and Artificial Intelligence. We anticipate that most services in the future will significantly rely on LPWAN because it incorporates features from LoRa, NB-IoT, and SigFox. Consequently, the usage of smart systems has increased in the recent past quite drastically because of such intriguing technologies. As a result of it, we are getting more dependent on these smart systems. However, as people's quality of life improves, so do the risks associated with dependent applications, such as data leakage and the integrity of the underlying network in the event of a failure. So far, it is generally established that it is simple to construct protocols that are secure from the outside world through the usage of traditional cryptographic techniques. However, when we have to

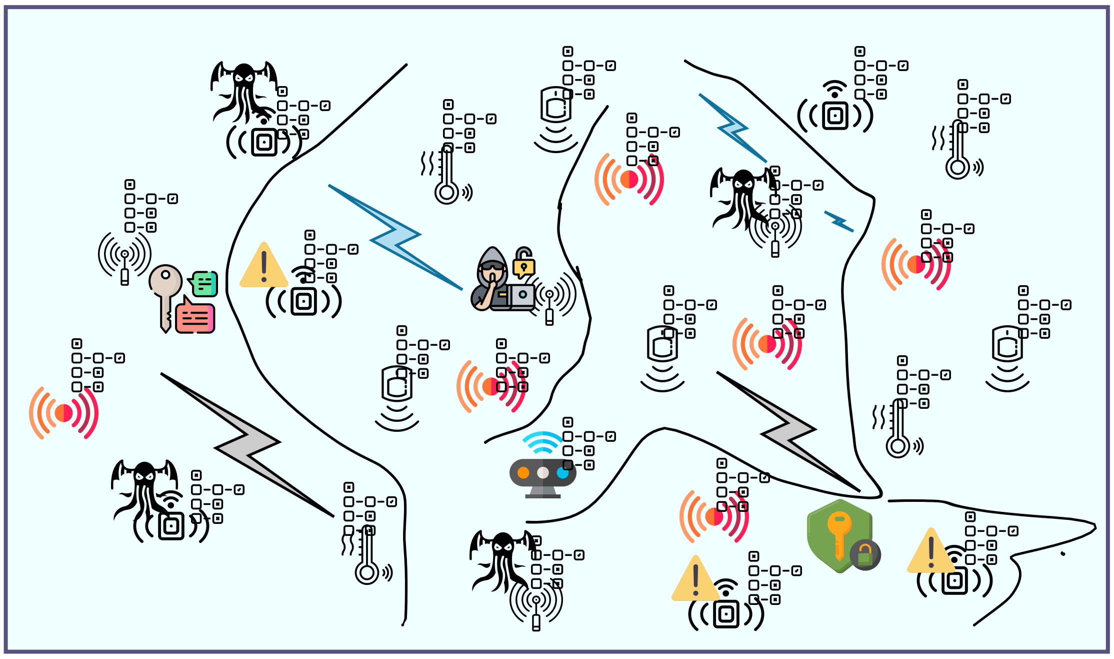

Figure 1.1: Overview of an IoT/WSN network while operating in practical settings

create a secure solution that is robust against failures and internal and external adversaries simultaneously, the scenario changes entirely and becomes non-trivial despite the usage of cryptography. The typical network structure possible in the real-world can be visualized as shown in Fig. 6.2. We need protocols that can provide sufficient guarantees while working in such practical settings.

1.2 Motivation and Objectives of the Study

As a part of our quest for building trustworthy IoT networks, we broadly focus on the following two domains.

Security and Privacy: A system being secure, as well as trustworthy, may not be able to preserve the privacy of the data from individual nodes. The problem of privacy in IoT becomes very important when the sensed data have a relation with specific users. However, preserving privacy under massive IoT systems is very challenging, especially under multi-hop setting. Sensitive user data may not be safe to reveal as its propagation doesn't limit to only one node. The data shared by each node is the same for all the nodes. If it's in plain text format, then all intermediate nodes between the source and sync node can gather relevant information about this sensitive data. If it's not in plain text, then the sync node can play the role of the potential information seeker. The available standard solutions assume sync node(s) as trusted third parties, and each node sends its sensed value to it with an aim to carry out some distributed learning task. Even if messages are encrypted, and intermediate nodes don't have access to them, Sync nodes can decrypt them and see the raw data shared by each node and interpret them locally to gain more information that clearly breaches node privacy. However, some of these applications will substantially use sensitive data to achieve the desired goal. It is understandable from past experience that the exposition of such data (amount of power consumption, Data used in Health monitoring) can have undesired consequences. Therefore, it becomes necessary to provide such services to users at the same comfort without compromising their privacy. Moreover, It has been observed that some of these services need to compute aggregation statistics over the data sensed by the participating nodes. Such aggregation becomes critical when it is being carried out on particular sensitive data. The traditional cryptographic techniques have provided us with optimal encryption and decryption techniques. However, such data obfuscation techniques ensure data security only when it is in the transmission or storage. Therefore, In order to perform any analytics over such data, the data eventually needs to be decrypted at the central server to perform the final aggregation calculation. Moreover, the central server can certainly become a single point of failure, as shown in [7]. Thus, it necessitates a system in place that protects the security of data both while it is in transmission and computation.

However, modern cryptography provides the service of Secure computation that allows a set of parties to compute some joint function of their private inputs while guaranteeing privacy (i.e., the parties learn the output and nothing more) and correctness (meaning the output is correctly distributed). Recently several solutions based on Privacy Enhancing Technologies(PETs) like Homomorphic Encryption(HE), Differential Privacy(DP), and Secure Multi-party Computation (SMPC) have been proposed in the literature. However, we believe that techniques like HE are inappropriate for IoT devices due to their low computation power. Moreover, we believe such techniques involve high computation with an increase in the number of nodes in an underlying network. Therefore, we seek MPC-based techniques to be the best fit for such low-power devices. Despite their low computation overhead, they frequently require interaction with other participants. Consequently, we believe that to realize the complete essence of MPC-based solutions for IoT/WSN networks, we require efficient communication protocols that could quickly achieve the requisite data sharing needs.

There has been a significant interest in inheriting flooding-based data sharing for low-power networks. It is so because of their quick convergence time with high reliability in comparison to asynchronous data sharing-based mechanisms, where nodes fight for their chance for transmission of their data, resulting in large data transmission time. This time significantly gets compounded when all the nodes need to share the data for all other nodes in the network, and this broadcast is a prime requirement for MPC based solution. So, if MPC solutions are less compute-intensive on one side of the spectrum, they require substantial communication infrastructure for realizing the goal. Consequently, we believe this gap can be bridged in the context of IoT networks with the usage of efficient data sharing protocols like [8] in asynchronous domain and flooding based communication mechanisms like [9]. Some works in [9] perform in-network aggregation but do so on plain-text data, which directly breaches the node privacy Moreover, their aggregation can work only for idempotent functions. Consequently, we seek an aggregation protocol that can guarantee: *Security, Privacy, Robustness, and Scalability.*

Trust: Secure communication protocols alone cannot provide trust in a system. For example, nodes in the systems can behave maliciously even after complying with security requirements in the case of applications like distributed database replication. The applications running on top of a flooding protocol assume every node is honest and thus would purely depend on information what their neighbors are saying and assume them to be true blindly. In the case of sensitive, e.g., data replication) applications, an adversary or a group of adversaries can inject false data to divert the network from having a single decision. Byzantine fault tolerance protocols can potentially help in alleviating the problem of failed or maliciously behaving nodes. The most fundamental strategy, known as Byzantine fault consensus, assists in masking arbitrary failures reported by failing nodes. Byzantine fault tolerance protocols can potentially help in alleviating the problem of failed or maliciously behaving nodes. It may therefore be realized that agreement is important for reliable decentralized applications. The way these applications leverages the distributed computing power of IoT nodes is quite motivating and intriguing. These technological elevations make it more likely that malicious attacks and software errors will occur regularly. Thus, it has now become a fundamental requirement to build resilient network services that can withstand a wide range of failure types in distributed systems. However, the incorporation of Byzantine Fault tolerance seems to be a more promising direction toward the adoption of IoT-based decentralized applications. The most fundamental strategy, known as Byzantine fault consensus, assists in masking arbitrary failures reported by failing nodes.

Currently, many applications need extensive coordination among their counterparts to do the defined job more precisely in low-power wireless networks. These applications can range from coordination among industrial controllers(Smart Grid) to mission-critical systems like a swarm of Unmanned Aerial Vehicles(UAVs). For example, The UAVs could be on some mission and would like to compute the exact target location with a common agreement to harm the enemy effectively, a deviation of which can lead to a significant loss. The prior proposed solution in the domain of consensus using the flooding based transmission [9] only manages crash failures. They restricted themselves to handling Byzantine failures as their underlying protocol is designed for in-network data aggregation, whereas we need all-to-all data sharing for byzantine failures. Thus, we plan to derive trustable consensus and aggregation protocols that can potentially help for any network-wide operations.

1.3 Contribution of the Thesis

In this work, particularly we view an IoT system as a massive collection of low-power devices spread over a wide area where each device can do some work, communicate with each other, and have limited energy. Moreover, we also consider that some devices are controlled by adversaries interested in gaining control over the network or want the network to deviate from the desired goal. The main weakness of an IoT system is the resource constraints in the devices. However, we perceive the massive number of devices in IoT as their strength. We view it as a massively distributed system and plan to compensate for the resource limitation in the devices with collaborative computing where device-to-device communication plays a vital role. To appropriately exploit the power of such a massive size, we deviate from the traditional asynchronous transmission-based communication strategies to time-slotted based multi-flooding based transmission strategies.

The success of the concurrent transmission-based protocols heavily depends on physical layer phenomena like constructive interference and capture effects which in turn depend on precise co-ordination among the devices. The underlying assumption for satisfying both of these properties is that all nodes need to share their data at the same time. The data transmitted by each node needs to be the same. Hence, no change in either data content or transmission time. Even any change in the data content of one node may potentially violate these physical properties. These physical layer phenomena make it even harder to make the concurrent transmission-based strategies secure as well as trustworthy. Thus, building secure solutions satisfying both these properties is quite a challenge as compared to asynchronous protocols. Following, we briefly mentions our contributions under the two broad domains:

- **Privacy Preserving data aggregation:** A decentralized system is composed of many independent nodes. Collaborative computation among such nodes is one of the crucial components of such a system. However, in order to achieve the goal, the nodes may need to share their original input with the other nodes in the network which may not be always desired. In general, the nodes may not want to share their private information for any global computation purpose. Computation of such functions itself may be one of the prime needs of the whole system. It's a challenging task to achieve these two conflicting goals together i.e. to compute a joint function on the private inputs. The flooding based all-to-all data sharing protocols[9] do not guarantee an individual node's data privacy if being used to carry out some distributed learning task that incorporates data from all the participating nodes. Making these protocols privacy-preserving would be an immediate requirement to build specific applications like body monitoring through smart wearables, smart meters readings in AMI, etc. The works [10, 11] introduce the notion of privacy-preserving computing using homomorphic techniques which results in an extra overhead in communication. Homomorphic encryption-based techniques have quite promising power for traditional computers. But their large computing cost makes them infeasible for low-power devices. The recent advances in Multi-party Computation (MPC) protocols using secret sharing schemes[12, 13] pose little overhead but provide sufficient security guarantees. In the proposed works, we seek MPC-based techniques to be the best fit for such low-power devices. Despite their low computation overhead, they frequently require interaction with other participants. Consequently, we believe that to realize the complete essence of MPC-based solutions for IoT/WSN networks, we require efficient communication protocols that could quickly achieve the requisite data sharing needs. In this context, we propose using Concurrent transmission-based protocols to attain aggregate statistics out of mutually distrustful entities in a privacy-preserving manner.

- **Byzantine Fault tolerance in IoT:** Trust is the most crucial factor which end-users look for before using any distributed application. Bare secure transmission protocols don't guarantee the system's trust. One way of achieving trust in decentralized networks is through consensus mechanisms. Recent works such as [14, 15] have tried to bring the concept of consensus for handling crash faults, i.e., node failures. But they fail to address the problem of achieving consensus in the presence of Byzantine failures where participants can behave differently from what they intended to do. In the proposed works, we are focussing to provide solutions that can help specific applications to work smoothly even in presence of both crash faults and byzantine faults.

1.4 Experimental-setup

1.4.1 Experimental-setup: Simulation Platform

The experimental set up consists of simulating in ns-2 and MSPSim (TinyOS). We create the network scenarios as per our need and thus, simulated and experiments were being performed. After carrying out the primary experiments on simulation platforms, the same were verified on the emulation platform which consists of real devices.

1.4.2 Experimental-setup: Emulation Platform

For local experiments we used QEMU emulator for ARM architecture based sensor nodes which comprised of 802.15.4 compliant radio devices. Here also, we make emulation setting as per our requirements. For real-devices experiments we used our local testbed Kanad at IIT Bhubaneswar which also comprised of 802.15.4 compliant sensor devices.

Chapter 2

Literature Review

2.1 Communication Mechanisms

Transmissions under asynchronous communication are mostly independent and uncoordinated. Due to the broadcast-driven nature of the wireless medium, asynchronous communication poses quite a high chance of collision among the packets transmitted by different nodes causing high wastage of time, bandwidth as well as energy in the devices. Consequently, applications like flooding, data sharing among multiple nodes perform poorly with an increase in the number of the participating devices. Under in-parallel concurrent communication [9] the transmissions of the packets from different nodes are coordinated which causes them to overlap perfectly in time. This in turn triggers physical layer phenomena called *capture-effect* which ensures correct reception of the packets in the respective receivers.

Works described in [9] shows how to achieve assign time slots to nodes, the primary requirement for flooding based protocols, through lightweight software and hence, gain the benefit in resource-constrained off-the-shelf IoT devices in a purely decentralized setting. There has been immense development in the context of IoT networks with the usage of efficient data sharing protocols like [16, 8] in asynchronous domain and flooding based communication mechanisms like [17].

2.1.1 All-to-all/Many-to-Many data sharing

Many-to-Many/All-to-All data sharing described in [17] has been a very important requirement in IoT/WSN systems. One-to-all flodding based protocols successfully conveys the data from one node to all other nodes in the network. Some works achieves many-to-many sharing through a naive way through sequential repetition of one-to-all floods one-by-one from all the source nodes. However, wide separation among the floods makes such adoption strategy perform poorly with more nodes and large areas. However, some more works are there in the other extreme, and achieves many-to-many data sharing through clever application of heterogeneous capture-effect. The work MiniCast [17] tries to minimize the inter-flood gaps by using a packet-level TDMA schedule and achieves all-to-all data sharing in a very compact form. MiniCast has been shown to outperform with a wide margin for all-to-all data sharing.

There are quite a few other existing all-to-all data sharing strategies as shown in [9]. However, these works either extend the same underlying idea or are based on the concept of MiniCast. The works mostly use multi-channel facilities or efficient network-coding strategies to enhance the performance. Throughout the course of this thesis, we mostly use these protocols as a communication medium.

2.2 Works related to Fault-tolerance in IoT & its drawbacks

2.2.1 Non-Byzantine Fault tolerance

Non-Byzantine node failures have been considered in several works [18, 19, 20, 21]. Paxos [19] is one of the earliest and most well-known non-Byzantine consensus protocols which is later expanded to the protocol Raft [21]. Both Raft and Paxos have been extensively used in many permission-based environments, e.g., Google's globally distributed File System [22], BlockChain enabled Hyperledger Fabric, [23] etc. Efficient many-to-many interactions, data sharing and consensus in resource-constrained IoT/WSN systems have been addressed in many works [9]. Very few of these works deal with the issue of fault tolerance. A recent work [14, 15], ports the protocol Paxos [19] to low-power IoT/WSN system. However, almost none of the existing works attempt to combat the existence of Byzantine nodes in IoT.

2.2.2 Byzantine Fault tolerance in IoT and its applications

Byzantine consensus is a significant component in the implementation of BlockChain technology [24]. For instance, BlockChain assisted crypto-currency BitCoin [25] uses Proof-of-work (PoW); similarly, PPCoin[26] uses Proof-of-Stake (PoS) to mitigate the possibility of Byzantine node failures. While these solutions work quite effectively in a distributed environment comprised of traditional resourceful desktop/server machines, they are not suitable for resource-constrained IoT systems. Some recent works try to bring the BlockChain service for IoT devices. However, these works exploit help from the cloud to carry out Byzantine fault-tolerant consensus. Such a split architecture although works but induces a significant delay and potential issues under higher demand.

Achieving Byzantine fault tolerance has been always a much harder job compared to tackling only non-Byzantine failures. The strategy known as *Practical Byzantine Fault Tolerant*[27] (PBFT) consensus, shows a quite different approach to accomplish the goal. Instead of depending on special issues such as computation capabilities in the participating nodes (as done by PoW, PoS), PBFT relies more on collaboration among the participants through information exchange which makes it more suitable for a generic decentralized system. It has gained quite popularity in various fields. The work [28] shows the application of PBFT in BlockChain-based audit systems to solve various security concerns in the consensus algorithms. The work [29] done at Renault Automobile Corp. demonstrates the application PBFT for the smooth processing of Insurance claims. PBFT has also gained attention in Industrial IoT [30]. In the context of VANET, PBFT has been already used to effectively eliminate illusion attacks [31], as well as, enhance the performance of BlockChain-assisted VANET.

2.2.3 Drawbacks of the prior works

However, communication overhead has been one of the significant challenges in realizing PBFT for real-world smart-systems. There have been attempts to improve the performance of basic PBFT protocol. For instance, the work [27] reduces the complexity from exponential [32] to polynomial. A class of works has been done to make the protocol suitable for IoT systems too. For instance, Pengs et. al. [33] propose a credit-based mechanism using reinforcement learning in order to reduce the communication among participants. The set of works [34, 35, 36, 33, 37] attempts to make PBFT scalable by adopting a divide-and-conquer approach where the system is first divided into multiple layers/groups. The problem is addressed separately in each of these groups and finally, the group leaders collaborate with each other to come to a conclusion. However, all these works show a simulation or theoretical validation of the concepts only. In contrast, in the current work, we make an endeavor to design a lightweight low-latency and scalable realization of PBFT for real-IoT systems.

2.3 Works related to Privacy-Preserving Data aggregation in IoT & its drawbacks

2.3.1 Data-aggregation in wireless sensor networks

Data aggregation protocols in WSN have been studied in many prior works such as [38, 39, 40]. The work [40] evaluates various secure data aggregation protocols based on different security requirements (e.g., *data confidentiality, data integrity*) of WSN. Work presented in [39] investigates secure data aggregation protocols in IoT by categorizing them into three main classes including tree-based, cluster-based and centralized architecture and compares these protocols with different metrics like *Accuracy, Network Lifetime and Latency.* Most of these works on PPDA can be broadly classified into three categories based on the key concept they employ - (a) *Homomorphic Encryption* (HE) (b) *Multi-Party Computation* (MPC) and (c) *Data-Obfuscation* (DO). The strategy proposed in this work is based on collaborative data-obfuscation with the help of *pseudo-random* numbers. Our work is more aligned with the work proposed in PPMP. However, apart from these theoretical concepts, in the following we review the existing works based on a few important issues regarding their practical relevance and applicability to serve real IoT-systems.

2.3.2 Classification of techniques for private aggregation

Homomorphic Encryption

HE: HE allows an aggregation operation to computed directly on cipher text. This alleviates the requirement of the deciphering of the data by the intermediate forwarding nodes in a system and hence compute the partial aggregation in a privacy preserving way. HE has been employed in a quite good number of works to achieve PPDA [41, 42, 43, 44, 45]. However, an HE based system to work, the sink/final destination node needs to know the key to decipher the final aggregation result. This enables the sink nodes to decipher the individual ciphertext of different node. To resolve these issue, various approaches have been taken. For example, the work PEPPDA [43]uses a tree structure and dynamic slicing of the data while the work 3PDA uses an intermediate *Data Collection Unit*. However, although conceptually these are quite correct, they are not practically realizable in IoT-systems as carrying out HE incurs huge computation overhead. Moreover, a significant fraction of the aggregation strategies do not use any security policy. The survey papers [40, 39] summarize the use of cryptography in few works. Lightweight cryptographic techniques are used in various works [41, 42]. However, these approaches are centralised and rely on TTP. Like some exceptions e.g., PPMP [46], our work does not use any specific cryptographic techniques for data hiding and also assume insecure channels for communication.

Secure Multi-party computation

MPC: These works mostly adapt a collaborative approach to preserve privacy. They operate in two rounds of inter-node interactions. In the first round, each node divides its data into multiple "parts" (*shares*) following and then share each of these parts to different nodes. In the second round, the nodes carries out summation of the received values and carry out a global aggregation to get the final result. The algorithms CPDA and SMART, proposed in the work [47] improve upon this basic concepts through application of *secret-splitting* and collaborative computing. A set of works employs *Secure Multi-Party Computation* which was formally introduced in the work [48]. *Shamir's Secret Sharing* [49] is the one of the mostly adapted strategy to achieve SMPC in a decentralized setting. However, all these MPC based, especially those use secure pair-wise channels, involve huge data sharing which is a serious concern in energy-constrained IoT-systems.

Data Obfuscation

DO: Obfuscating the data with the help of random noise has been a very standard way of hiding the original data [50, 51, 52, 46]. Pseudo-random numbers are used for the purpose. The noise is generated in a organized way so that effective noise can be removed after aggregation is completed.

Aggregation using Trusted Central Co-ordinator

Trusted Third Party (TTP): Use of a TTP for key distribution or bootstrapping is a common norm used in most of these existing works [41, 42, 53, 45]. Most of the works employing HE, usually take the help of TTP for key generation since they don't perform collaborative computation rather rely on an centralised entity. However such strategies open up a channel for leaking out private information. In our work we show an efficient way to generate the keys internally through a collaborative mechanism and hence fully avoid the use of TTP.

Centralized/Decentralized: Intrinsically, in many of the above mentioned works [43, 44, 45], use of centralized model of computation has been a common issue. It has been used, either to make HE based approaches compatible by performing heavy computations in a centralised entity like Base Station (BS) which computes the aggregated value or depending on a trusted authority for key-generation. Very few of the works present, carry out all the operations in a purely decentralised form. Most of the data-obfuscation based strategies to our knowledge fall in this category.

Collaborative-Computation: Most of the HE based works follow a centralized model of computation for computing the aggregated value. Such centralized approaches incurs some common issues like single-point-of-failure where BS becomes unavailable due to heavy network traffic which effects reliability of real time data computation. Such scenarios can be avoided by collaborative computation of all nodes in the network or using the concept of SMPC. For example, [47] proposes two algorithms CPDA(Cluster-based Private Data Aggregation) and SMART(Slice-Mix-AggReGaTe) for collaborative data aggregation while preserving data privacy, but they still rely on a centralized entity like BS for computing the final aggregated value. The work [54] realizes SMPC in a decentralized environment. However, it relaxes the original problem by assuming that the data can be shared through a secured channel between any pair of nodes within the network which is not realistic for low-power wireless systems like IoT/WSN. Even many DO based works use collaborative computation. Work present in [50] uses a clever application of splitting its data into chunks to achieve PPDA. However it depends on a TTP for system initialisation.

2.3.3 Drawbacks of the prior works

Fault-tolerance: Fault-tolerance is a very significant issue which is not considered in most of the above mentioned and other existing PPDA works. Sudden drops of multiple nodes quite common in IoT systems and may result in wrong computation of the result in rigid strategies like PPMP. The work [55] considers fault tolerance in Smart Grids, however it relies on TTP in case a node drops. In our work, we try to keep the protocol flexible and fault-tolerant so that they can quickly detect and rectify the result.

The work [46] is the one that is closest to our the approach proposed in our current work. It considers a purely unsecured environment and operates in a decentralized fashion. The execution of the protocol does not rely on any third party. However, PPMP assumes a specific circular arrangement of the nodes. During the bootstrapping phase each node is supposed to share the values only to its adjacent nodes in a specific order which is difficult for maintaining such order in IoT settings. Our proposed protocol does not exploit any such specific ordering. Also PPMP uses a prime number p during the initialisation phase and the algorithm multiplies all the secret values and applies modulo(p^2) over the product. Thus p is supposed to be greater than aggregated value. However as mentioned such situation can be avoided is assuming a large prime number. However in resource constraint devices like WSN/IoT, operating over such large numbers increases computational complexity.

Communication: The works based on MPC and DO mostly exploit heavy communication among the nodes. Naturally, message complexity in such solutions goes very high as ($O(n^2)$, n being the number of nodes). In our work, we do not compromise the message complexity of our DO based scheme. Instead of avoiding we exploit the recent advancements in CT based strategies to efficiently achieve the necessary communication among the devices. As far as the knowledge of the authors goes, this is the first attempt when CT has been exploited to achieve DO based PPDA.

System-implementation: Finally, most of the above mentioned works although claim to be suitable for IoT systems, very rarely implement in an IoT-system. Rather, most of these works either provide theoretical study with mathematical proofs. Some of the works show the implementation in high-end computers or RaspberryPi's (V3) which cannot be claimed as resource-constrained IoT-device.

Table2.1 summarizes the pros and cons of all the above mentioned works with the pros high-lighted using gray color.

Works	Class	Centralized /Decentarzlied	Use of Trusted Third Party	Communicatio	Computation	Implementation in IoT-system
PEPPDA	Homomorphic Encryption	Centralized	Yes	Less Efficient	Less Efficient	Yes(Smart Grid)
3PDA		Centtralized	No	Less Efficient	Less Efficient	Yes(Smart Grid)
P2DA		Centralized	Yes	Efficient	Efficient	Yes(Smart Grid)
LVPDA		Centralised	Yes	Efficient	Efficient	Yes
LPDA		Centralised	Yes	Efficient	Efficient	Yes
CPDA	Multi-Party-Computation	Centralised	No	Less Efficient	Less Efficient	Yes(WSN)
SMART		Centralised	No	Efficient	Less Efficient	Yes(WSN)
SMPC		Decentralised	No	Less Efficient	Less Efficient	No
PPMP		Decentralised	No	Efficient	Efficient	No
HWAPPA	Data-Obfuscation	Centralised	Yes	Efficient	Efficient	Yes(Smart Grid)
DPPDA		Decentralised	No	Less Efficient	Less Efficient	Yes
PEDA		Centralised	Yes	Less Efficient	Less Efficient	Yes(Smart Grid)
LiPi		Decentralzied	No	Very efficient, uses of CT	Efficient	Yes

Works	Class	Use of Cryptography	Fault-Tolerance	Latency	Energy Efficiency	Degree of Collusion Tolerated
PEPPDA	Homomorphic Encryption	Yes	No	Less Efficient	Efficient	Not discussed
3PDA		Yes	Yes	Less Efficient	Less Efficient	Not discussed
P2DA		Yes	No	Less Efficient	Less Efficient	Not discussed
LVPDA		Yes	No	Efficient	Efficient	Not discussed
LPDA		Yes	No	Efficient	Efficient	Not discussed
CPDA	Multi-Party-Computation	Yes	No	Less Efficient	Less Efficient	Not discussed
SMART		Yes	No	Less Efficient	Efficient	Not discussed
SMPC		Yes	No	Efficient	Efficient	Not discussed
PPMP		Yes	No	Efficient	Efficient	Very less
HWAPPA	Data-Obfuscation	Yes	No	Efficient	Efficient	Not discussed
DPPDA		No	No	Efficient	Efficient	Not discussed
PEDA		Yes	No	Less Efficient	Less Efficient	Not discussed
LiPi		No	Yes	Very effciient	Very efficient	Highest

Table 2.1: Summary of the existing approaches to achieve Privacy-Preserving Data aggregation in IoT/WSN systems.

Chapter 3

Practical Byzantine Fault Tolerance in IoT

3.1 Introduction

Human civilization is tending to depend more on the use of smart systems. Technologies such as Internet-of-Things (IoT) and Wireless Sensor Networks (WSN) play crucial roles in building these smart systems. They operate through decentralized collaboration among a large number of independent units, commonly known as IoT devices. For instance, in *automated surveillance systems* [56] the independent devices equipped with image/video/sound sending capability are installed at various places, covering the area under surveillance, to collaboratively monitor the area. Similarly, in a *structure monitoring system* [57], many devices having appropriate sensing capabilities are installed covering the vital parts of the structure (e.g., a bridge or a building), where the status of the structure is obtained through a collaborative effort among all these devices. Similar scenarios can be obtained from other smart systems such as smart-grid systems [58], intelligent-transportation systems, industry-4.0, etc.

Unfortunately, any IoT device in any such smart system is susceptible to various unexpected issues, e.g., software errors, failures, or even various types of attacks. These issues can significantly disturb the overall system integrity of a smart system and may also result in false reporting while interacting with the end-users which may lead to catastrophic effects. For instance, while monitoring a critical structure, a compromised IoT device may wrongly report the status of a component. Similarly, in a flock of UAVs with some mission, such issues may cause some of the UAVs to deviate from the target which in turn may lead to an overall failure and significant loss.

Fault-tolerant consensus protocols[24] play a vital role in establishing the trustworthiness of a system in spite of the chances of node failures. Such failures can be either *non-Byzantine* or *Byzantine*. Consensus protocols to handle non-Byzantine failure assume a weak failure model, e.g., simple *fail-stop* or *node crash*, and are mostly used in systems within a controlled environment, e.g., data center, where they are operated by authenticated users. In contrast, consensus protocols capable of handling *Byzantine* failures [59], entail more comprehensive failure models where nodes can be compromised and have malicious intent behind participation. They are applicable for both open and decentralized systems, e.g., BlockChain [25, 60].

Fast consensus has been addressed by many prior works in the context of IoT/WSN, using aggregation protocols[9]. However, these works do not consider any failure possibilities. Moreover, some of the works realizes the well-known consensus protocol Paxos. But, it considers only non-Byzantine faults. Support for Byzantine failures to build a trustworthy IoT system would be an inevitable issue in near future. However, existing efforts to realize Byzantine fault tolerance in IoT/WSN systems are mostly theoretical and simulation-based. There is almost no work so far that attempts to practically achieve the same for resource-constrained

IoT/WSN systems.

Unlike handling non-Byzantine failures, the challenges in dealing with Byzantine failures are quite different and much harder to address. The earlier work shows how a network-wide max finding operation can be exploited to efficiently manage non-Byzantine failures in IoT. However, in contrast, to manage Byzantine faults the devices need to share the actual opinion/messages with each other to come to a conclusion which naturally involves multiple rounds of many-to-many/all-to-all data-sharing among the nodes making the protocol quite complex and heavy for resource-constrained IoT/WSN systems. In this work, we exploit the recent advancements in flooding-based transmission communication strategies [9] to build an efficient solution for this problem.

The primary contribution of the work is summarized below.

- An efficient solution to achieve Byzantine fault tolerance for resource-constrained IoT systems leveraging multi-flooding based communication framework has been proposed.

- Special design considerations have been adopted to make the strategy scalable and optimize the overall completion time and energy consumption in the devices.

- The proposed design has been implemented for ARM architecture based sensor devices.

- The performance of the protocol has been evaluated in both simulation as well as emualtion.

3.2 Background

A class of existing solutions to mitigate the existence of Byzantine nodes exploit the computational capability of the participating nodes. However, a typical IoT/WSN device mostly lacks enough processing capability to realize these strategies. The approach shown in PBFT [27] demonstrates how to deal with the Byzantine nodes only based on inter-device communication. Unfortunately, limited energy availability in the tiny devices used in IoT/WSN systems heavily restricts their communication ability too. In addition to this, uncoordinated transmissions in traditional asynchronous communication-based protocols in IoT/WSN systems, waste a lot of throughput and energy in the devices due to collisions among the packet making it even harder to meet the communication requirements to realize PBFT.

Recently there has been quite a lot of developments in flooding based data transmission strategies [9]. These works demonstrate high reliability and low-latency communication in systems comprising of a large number of nodes. In the current work, we exploit these strategies to achieve PBFT in IoT/WSN systems. In the following, we first provide a brief description of about PBFT. A prime part of the realization of PBFT in IoT/WSN systems needs to deal with massive data sharing among the participating nodes. Therefore, in this works we exploit flooding protocols to fulfill the need. We explain our need and proposed strategy in the design as detailed in section 4.3.

3.2.1 Practical-Byzantine Fault tolerance (PBFT)

Under non-Byzantine failures, the query under consideration can be first shared with all the nodes in the system, and then replies can be collected from these nodes. Since the nodes are not supposed to reply incorrectly, the reception of at least 50% of the participants in the systems would ensure a consensus to be reached. In contrast, under Byzantine failures, a node forms the notion of consensus through discussing the issue with the other nodes in the system. Specifically, in PBFT, for a certain query, how many other nodes are replying with the same answer is considered to be a very important issue. Formally, a *quorum* is defined as the minimum number of nodes to pass a decision in an assembly. In PBFT, a node to achieve consensus amidst various types of failures needs to reach a *quorum*. The minimum number of nodes necessary to be present in a system to enable every node to reach a quorum in spite of the presence of at most f non-Byzantine failures is $2f + 1$ [27]. Conversely, under non-Byzantine failures, for N number of nodes the quorum size is $\frac{N}{2} + 1$. Under Byzantine failures, in contrast, quorum size is $\frac{2N}{3} + 1$ [61].

A node in the system requesting consensus is referred to as a *client node*. The request from the client node is conveyed to a designated *primary node* which triggers the consensus process composed of five phases as depicted in Fig. 3.1(a). A simplified description of these phases are given below.

1. **LAUNCH:** A client node (c) initiates the process by sending a request message *Req* to the primary node.

2. **PRE-PREPARE:** The primary node, broadcasts *Req* to all the participating nodes.

3. **PREPARE:** All the nodes in the system, validate the correctness of the request in *Req* and broadcast a *Prepare* message in the system conveying their opinion about it.

4. **COMMIT:** In this phase every node first individually decides whether it has reached quorum or not based on the number of alike messages received in the previous phase. If the number of alike messages is more than $2f + 1$,

a node decides to go for commit otherwise not. In this phase also the nodes their decisions with each other.

5. **EXECUTE REPLY:** In this post COMMIT stage its checked globally how many nodes in the system has achieved consensus. A client also consider its request as in consensus if it received $f + 1$ similar decisions. In case $3f + 1$ nodes have reached quorum, a global consensus is assumed to be achieved for *Req*. However, when consensus is not achieved, it goes to the VIEW-CHANGE phase as briefed below.

6. **VIEW-CHANGE:** In PBFT, the three consecutive phases PRE-PREPARE, PREPARE, and COMMIT together are considered to be a *view* formed under the leadership of the current primary node. If a certain view fails to result in consensus, first, a new primary node is decided based on a globally defined rule. Next, the new primary node starts the change of view and restarts the process with the PRE-PREPARE phase. With f number of traitors in the system, in the worst-case maximum $f + 1$, view formations may be needed to achieve a consensus.

In the following, we describe the design of our protocol to achieve Byzantine consensus with the help of combination of several communication mechanisms. We assume the sink node itself to play the role of a client as well as a primary node.

3.3 Design

The three phases employed by PBFT are pictorially explained in Fig. 3.1(a). In the PRE-PREPARE phase, the primary node carries out a network-wide *one-to-all* sharing of the query message while both the PREPARE and the COMMIT phase execute a network-wide all-to-all interaction among the nodes to exchange

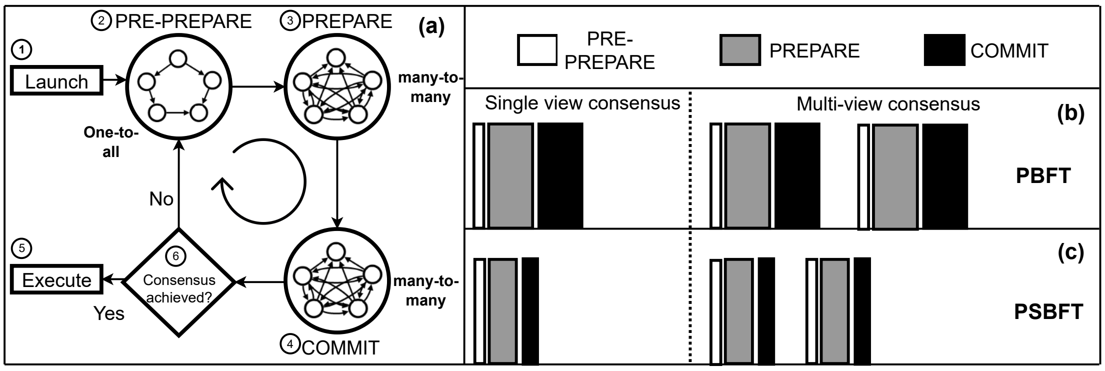

Figure 3.1: Execution of PBFT. Part (a) shows the flow diagram of a standard PBFT execution. Part (b) and (c) show the timing diagram of PBFT without optimizations and with optimizations, respectively.

the opinion regarding the message as well as check how many nodes could reach the consensus, respectively. All these phases thus involve an extensive number of message passing among the nodes. Moreover, possible change of view may happen several times which would bring many repetitions of these three phases. Under in-parallel time-slotted concurrent transmission, although communication happens in a very compact form, it's quite expected that a naive realization of PBFT may still drain a considerable amount of energy in low-power IoT/WSN-devices. To optimize the performance we make an in-depth study of the Byzantine consensus process in the context of IoT and come up with a set of observations as summarized

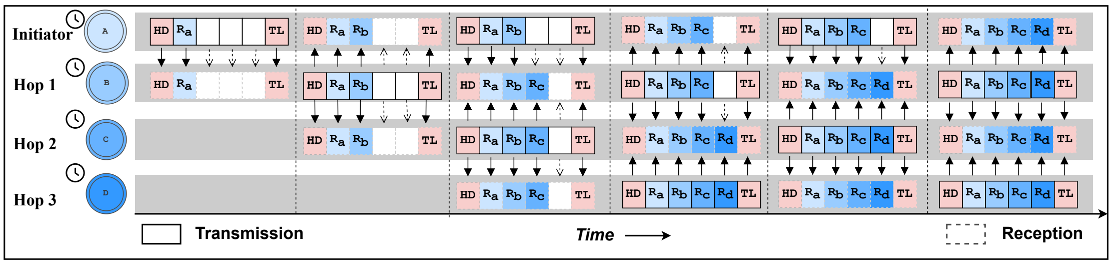

Figure 3.2: Execution of PREPARE Phase using All-to-All Data Dissemination

below. We use these findings to design our final protocol.

3.3.1 All-to-all vs many-to-many

The purpose of the data sharing in the PRE-PREPARE and COMMIT phase in PBFT is to ensure that every node in the system reaches the quorum, i.e., to enable every node to obtain the required number of alike messages. From the definition of the quorum size for Byzantine consensus, it can be observed that strict all-to-all data sharing may not be always necessary. Rather, it depends on the actual number of traitors or faulty nodes. In a system with N nodes, although PBFT can support max f traitors, where $N = 3f + 1$, the actual number of traitors present in the system may be much lesser than f, say k ($k < f$). In this situation, the necessary size of the quorum is $2k + 1$. Therefore, to reach quorum it's enough to ensure that each node receives at least $2k + 1$ alike messages.

Note that for k traitors, the number of distinct messages/opinions that may differ from each other would be maximum of k. Thus, by the pigeonhole principle, it can be said that a node would require to wait for messages from at max $3k + 1$ distinct nodes which ensures $2k+1$ out of them are from the non-traitor node, hence alike. For a random distribution of k traitors, a node may need to wait for even less as it may receive $2k + 1$ alike messages much earlier. Thus, knowledge regarding the number of traitors as well as their distribution in the system can be exploited for faster convergence. Based on this concept we simplify the PREPARE-phase.

3.3.2 Restricted behaviour of the traitors

Under traditional wired communication systems, unicast is the default communication mode. A traitor node in unicast gains enough scope to communicate differently to each of its neighboring nodes, as a response to a given query. In a wireless medium, any communication happens by default in broadcast mode. However, under asynchronous communication broadcasts are often abstracted as unicast which again can be exploited by the traitors to create confusion. In contrast, flooding protocols explicitly uses a broadcast-driven communication framework to achieve a faster spread of information and speed up convergence. However, broadcast-based communication forces the traitors to behave consistently to all their neighbors.

In addition, if a traitor node tries to share different data in consecutive or different time-slots, it creates visible disturbances in the process and hampers the performance of the protocol which enhances the chance for it to get detected. This, in turn, refrains a traitor from even tweaking with the data while forwarding. In-parallel data dissemination, thus, in general, naturally restricts the Byzantine capability of a traitor. However, note that the traitors can freely pass wrong/imperfect opinions in their own packet/slot.

Based on the above-mentioned observations, we optimize the PBFT consensus protocol as described next.

3.3.3 Protocol

The proposed protocol is depicted in Algo. 1. We design the PRE-PREPARE phase using an instance of one-to-all dissemination. The query under consideration is disseminated by the initiator node (primary) to all the other nodes in the system. Execution of one-to-all flooding, in general, happens very fast over a large network setting. Subsequently, the nodes start the PREPARE phase where all nodes share their opinion regarding the query with each other. An instance of the protocol MiniCast is used for this phase.

The primary node makes an assessment of the traitor nodes in the system and shares a value k in the query packet indicating the approx no. of the traitors. A random distribution of the traitors is assumed. Based on k, the initiator also derives the value of a vital parameter of MiniCast known as NTX and shares the same in the query packet. An appropriate value of NTX is used to restrict the execution of MiniCast for a limited duration only upto getting enough opinions to reach quorum in every node. This is detailed in the next section. During PREPARE phase, every node keeps track of how many alike messages are received (in variable $nAlike$ in Algo. 1).

In the COMMIT phase, it's enough to share a single bit of information from each node indicating whether the node could reach the quorum in the preceding PREPARE phase. In particular, a node sets a flag in case $nAlike \geq 2k + 1$. We use an aggregation protocol for this phase. Aggregation protocols originally uses a single flag bit for every node to keep track of how many nodes have contributed to an aggregation process. The execution stops when all the nodes have contributed successfully. To serve our purpose, we add one more flag bit for each node which is set in case the corresponding node reaches the quorum. The use of just two bits of contributions from each node allows the whole COMMIT phase to run very fast even when the number of nodes is large.

At the end of the COMMIT phase, every node in the network gets to know how many nodes in the system have reached the consensus (captured in the variable $nCflags$ in Algo. 1). In case $nCflags \geq 3k + 1$, a global consensus is assumed to be achieved. If consensus is not achieved, there can be two distinct reasons. It may be either due to a faulty primary node or due to insufficient spread of information caused by limited execution of MiniCast.

Since exploring the exact reason is not possible, at this stage we first repeat the PREPARE phase with NTX one more than the last used value (i.e., NTX+1). We observe the change in the COMMIT phase. If considerably more number of nodes are found to be reaching quorum with the increased NTX, it indicates wrong assessment of the traitors in the previous round. To rectify the same we repeat the process until a global consensus is reached. We save this new NTX and use it for all the rest of the rounds. In case significant improvement is not visible after increasing NTX by 1, the nodes go for changing the primary node and repeat at max $k+1$ number of times (VIEW-CHANGE) to reach consensus.

Fig.3.4(a) shows the execution of different phases of PBFT(referred to as PBFT itself later) through the naive implementation, and Fig. 3.4(b) shows a modified/optimized version of naive PBFT. We refer to our proposed version as PSBFT (*Practical Synchronous Byzantine Fault-Tolerant Protocol*). Moreover, Fig. 3.1(b) and Fig. 3.1(c) show the timing diagram of PBFT and PSBFT respectively. The execution of MiniCast for the PREPARE phase in PSBFT in a network consisting of three hops is depicted in Fig. 3.2. One representative node from each layer is considered. Transmissions in MiniCast happen in the chain of packets. The beginning and start of a chain are indicated by the packets HD and TL. It starts with the initiator node transmitting the data packet received by the first-hop nodes. The chain transmitted by the first-hop nodes triggers the transmission by the second-hop nodes and so on.

The number of times a chain is transmitted and hence, received by a node is controlled by the parameter NTX. NTX, thus, governs the degree of spread of the data from each node. In the naive implementation of PBFT we set NTX as the maximum value necessary in a network, while in PSBFT, NTX is ideally set to the minimum value which is enough for every node to reach the quorum. Appropriate tuning of the value of NTX brings a huge difference in the performance as depicted in Section 4.4.

We experiment with the implementation of MiniCast in C language on ARM architecture based sensor devices over various simulation configurations in MSP-SIM(TinyOS) as well as two Emulation environments containing 24 and 45 nodes respectively. We refer the first scenario as Emulation-1 and the second one as

Emulation-2. Figure 3.3 highlights the rise in the network coverage with NTX for different network configurations. It is observed from all these results that to achieve 100% network coverage we need to set the value of NTX above 7. But based on the number of nodes and the simulation area in MSPSIM, the value of NTX required to get up to 70% coverage is quite less. Fig. 3.3 (a)-(c) shows the results in various simulation settings. The same trend is visible in both emulation settings as shown in Figure 3.3 (d). Its also visible that, the exact relationship depends on the specific network setting. Therefore, deciding the correct value of the NTX happens to be a nontrivial task which we describe next.

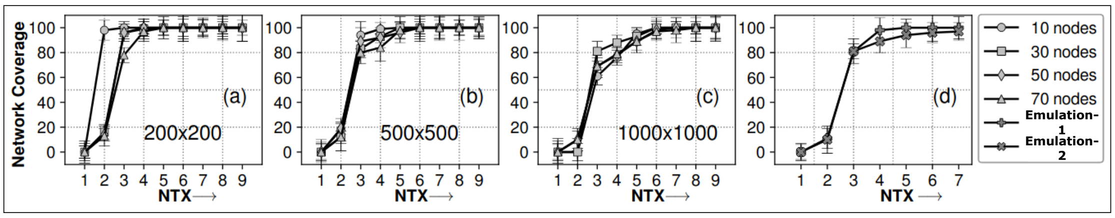

Figure 3.3: Percentage of data received by the nodes with the variation of NTX.

3.3.4 Restricted data-sharing

An iteration of a data-sharing process executes in multiple rounds of data reception. In a single reception round, any given node acquires data from a certain subset of the nodes. The maximum number of reception rounds is the same as the value of NTX. In the first reception round, a node would receive the data from its nearby nodes. In the next round onward, it receives data from nodes at higher hop-distances from it. We call this sequence of a number of data elements received in each of the reception rounds as *reception-profile*. After a sufficient number of reception rounds, a node receives data from all the nodes in the system. The reception-profile hence substantially depends on the structure of the underlying network. It also varies based on the hop-distance of a node w.r.t. the initiator of

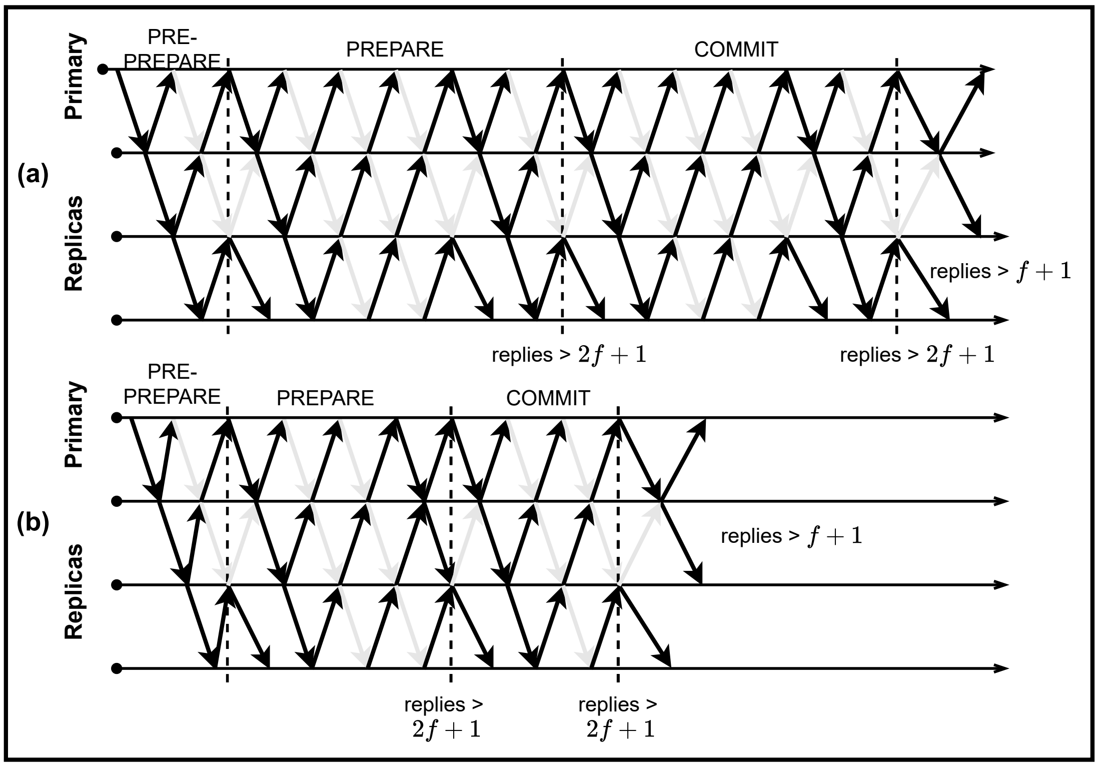

Figure 3.4: (a) **Execution of PBFT using flooding based communiction primitives:** Initially the primary/initiator node forwards the Pre-Prepare request to all nodes using one-to-all. Afterward, all phases are achieved using broadcast driven data-sharing. **Proposed Optimised execution via PSBFT:** Initially the Pre-Prepare is performed using one-to-all flooding. Afterward, PREPARE phase is using *Minicast* primitive, which eventually results in all-to-all data sharing. Moreover, the COMMIT/VIEW-CHANGE Phase is using in-network aggregation protocol.

Algorithm 1 PSBFT

```
1: LAUNCH:
2: Request: Client forwards request $Req$ to the primary $p$ node.
3: Verification: Primary node verifies the validity of $Req$.
4: Make Packet: Primary node prepares a packet $prepare$ including the query, NTX,
   and a traitor assessment $k$.
5: PRE-PREPARE:
6: Disseminate: Primary broadcasts $prepare$ to all nodes
7: for all $n_k \in N$ do                                        ▷ For every node
8:     Participate actively in execution of the instance of one-to-all data flooding pro-
       tocol to disseminate the packet $prepare$ to all the nodes in the network.
9: end for
10: PREPARE:
11: Set $ntx$=0
12: for all $n_k \in N$ do                                       ▷ For every node
13:     while $ntx \leq NTX$ do
14:         Transmission:
15:         Prepare the packet with reply.
16:         Broadcast the packet to the neighbors.
17:         Wait for reception of the packets from the neighbors.
18:         Reception:
19:         Collect the opinions from the nodes received so far.
20:         Consensus:
21:         Count number of alike answers in $nAlike$.
22:         if $nAlike \geq 3k + 1$ then
23:             Set internal state as "Reached quorum"
24:         end if
25:         ntx++.
26:     end while
27: end for
28: COMMIT:
29: for all $n_k \in N$ do                                       ▷ For every node
30:     Carry-out many-to-many data aggregation
31:     Set the flag to be '1' in case the consensus is achieved otherwise set it to '0'.
32: end for
33: VIEW-CHANGE:
34: for all $n_k \in N$ do                                       ▷ For every node
35:     Count the number of flag bits received in $nCflag$
36:     if $nCflag \geq 3k + 1$ then
37:         Set state as "Consensus achieved"
38:     else
39:         Decide the next initiator in the sequence OR increase NTX = NTX+1
40:         Go for view change: Repeat from Step 6.
41:     end if
42: end for
```

the data sharing process.

Setting a correct value of NTX in the PREPARE phase is very crucial. A larger value of NTX wastes time and energy in the nodes, while a lower value of NTX may be insufficient to reach quorum in many nodes. The knowledge of the reception-profile of each node in the system can be used to properly derive the appropriate value of NTX. However, collecting this information from all the nodes in the network would be a time-consuming task. We simplify the process based on the following observations.

Reception-profile of nodes: A dissemination process starts from the initiator node and proceeds hop-by-hop. Therefore, the time point when the process reaches a node that is located far away from the initiator, the node gets a lot of data items in comparison to the first reception at a node located close to the initiator. Thus, higher hop-distance of a node from the initiator implies a larger number of distinct messages (from different nodes) during each reception-round. Let us consider reception-profile (cumulative over reception number) of a node x at hop-i as, $< x_{i1}, x_{i2}, x_{i3}, ..., x_{intx} >$, while the same for a node y at hop-j be $< y_{j1}, y_{j2}, y_{j3}, ..., y_{jntx} >$. If $j \geq i$, we would see $< x_{i1} \leq y_{j1}, x_{i2} \leq y_{j2}, x_{i3} \leq y_{j3}, ..., x_{intx} \leq y_{jntx} >$. Based on this it can be inferred that the reception-profile of the first-hop nodes would be good enough to decide the global NTX.

Reception-profile over iterations: In MiniCast, due to TDMA based separation of packet-transmissions by different nodes, we observe that the chance for a node to fail in receiving the data at a specific slot is quite independent than others and hence, the reception-profile of a node varies quite less with the iterations. Conversely, the reception-profile data over the nodes show quite stable behavior.

Fig. 3.5 shows the comparison of the cumulative reception-profiles of certain randomly selected nodes during the execution of MiniCast in Emulation-1 and Emulation-2 environments. It is quite visible that in general with increase in hop-distance the accumulation of data at each reception count goes higher. The experiment is conducted for at 1000 iterations and the standard deviation values are reflected through the error bars in the graph. The results demonstrate quite a low variation of the reception count over different iterations.

Algorithm 2 NTX assessment

```
1: BEGIN:
2: Initiator starts the All-to-all/Many-to-Many sharing with a large NTX value.
3: for all $n_k \in N$ do                                        ▷ For every node
4:     for $i$ gets 1 to n do                                   ▷ Nodes locally store
5:         Reception_Count[$n_k$].push_back(#Nodes received)
6:     end for
7: end for
8: FOLLOW: All/Many-to-one data sharing                         ▷ Network View
9: for all $n_k \in N$ do
10:    Each node sends its Reception_Count list to Initiator
11: end for
12: Optimal NTX: Required reception/transmission count to handle $f$ failures in any
    network
13: NTX ← 0
14: for all $n_k \in N$ do                                       ▷ For every node
15:    for $i$ gets 1 to n do
16:        if Reception_Count[$n_k$][i]$\geq 2f+1$ then
17:            NTX ← $\mathbf{max}(NTX, i)$
18:        end if
19:    end for
20: end for
```

Based on the above observations and their experimental evidence we design

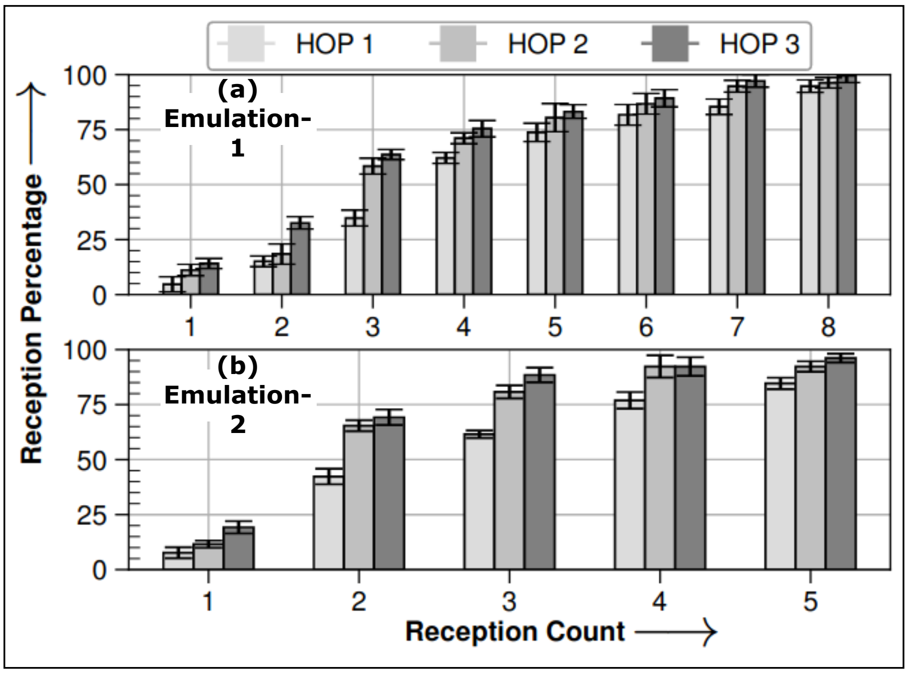

Figure 3.5: Percentage of Data Received for nodes present at successive hop distance

an algorithm (Algo. 2) to assess the NTX for the PREPARE phase. We invest two rounds of MiniCast for this part. In the first round, we set the NTX quite large based on the diameter and size of the network so that every node can derive its reception profile. In the next round, the reception-profiles from the first-hop nodes only are collected at the initiator. Note that this process is invoked only once during the bootstrapping phase and is independent of the PBFT process. Successful completion of the process enables the initiator node to calculate the correct NTX node based on the assessment of the number of traitors. In particular, for k number of traitors, the initiator calculates the NTX as the minimum value of m for which the $\sum_{i=0}^{i<m}(x_i) \geq (3k+1)$, i.e., sufficient for collecting enough messages from the surrounding to reach quorum in the nodes.

3.4 Evaluation

We implement both PBFT and PSBFT in C language. We simulate the protocols in MSPSIM (TinyOS). We also experiment with the implementations in the two emulation environments i.e., Emulation-1 and Emulation-2 composed of 24 and 45 802.15.4 compliant radio motes, respectively. The success of the PSBFT largely depends on the right selection of NTX for the PREPARE phase. Due to space limitations we do not include the detailed study of the NTX assessment part in this paper. However we use the outcome of the NTX assessment for different network setting for further experiments.

3.4.1 Metrics

To compare the performance of PBFT and PSBFT, we use the following two metrics.

Latency: It is the time taken for a process to achieve consensus. The value is calculated in each node and averaged over all the nodes and all the iterations.

Radio-on time: It is the total time necessary for a node to keep its radio-on to complete the execution of the protocol. The value is calculated in each node and averaged over all iterations and nodes. This value can also be used to understand the energy consumption of a device also.

3.4.2 Results

PBFT and PSBFT are compared under the presence of the Byzantine traitors under various network configurations through simulation as well as emulation. The number of traitors is varied from zero to the maximum number that can be supported by the algorithm (i.e., one-third of the number of nodes). Figure 3.6 shows the comparison results for a 70 node simulation for different deployment

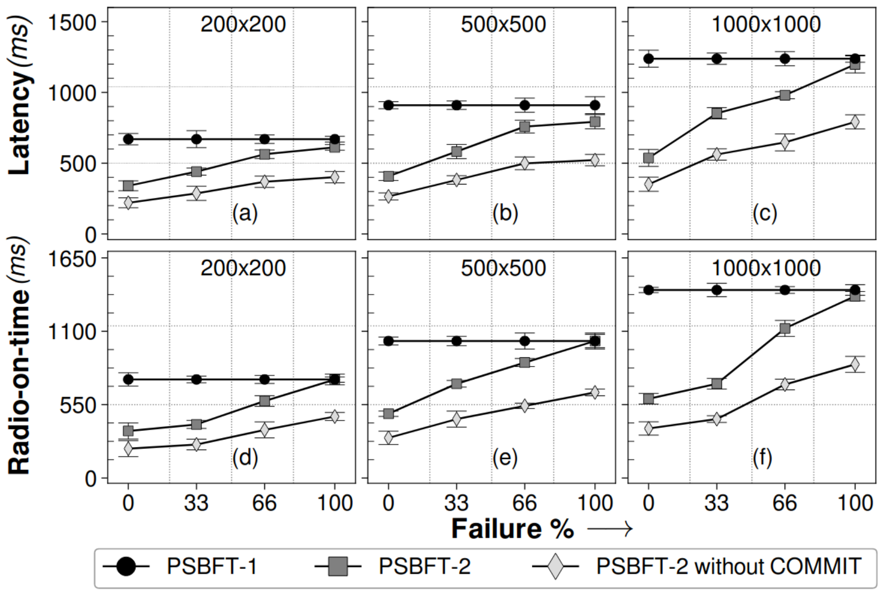

Figure 3.6: Latency and Radio-on-time for 70 node simulation over various network diameter

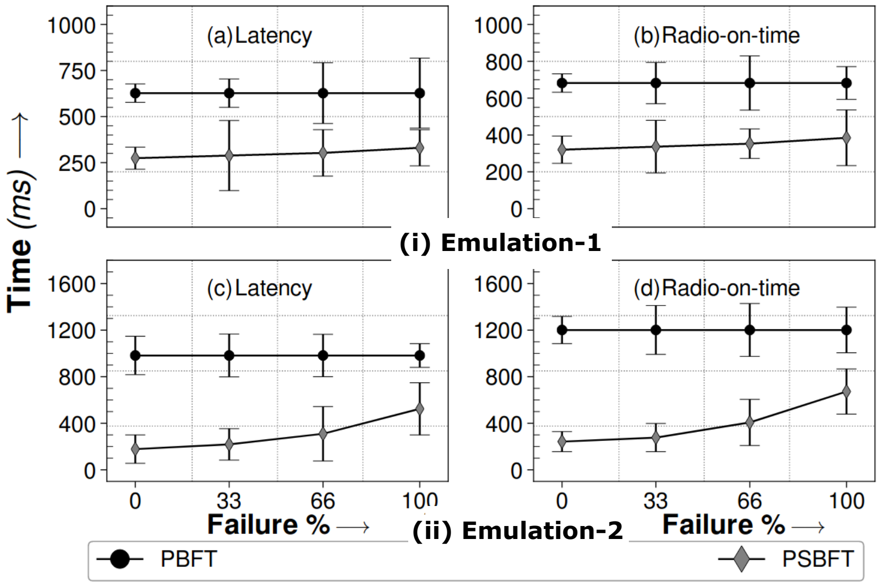

Figure 3.7: Latency and Radio-on-time for Emulation-1 and Emulation-2

areas. Figure 3.7 shows the comparison results in Emulation-1 and Emulation-2 in terms of latency and radio-on time.

PSBFT is always found to be performing much better than PBFT even when there are many traitors in the system. In summary, in a 70-node network distributed over 1000×1000 m^2 simulation area, PSBFT performs on average 55% and up to 73% faster and consumes on average 53% and up to 71% lesser radio-on time in comparison to PBFT for a wide variation in the number of traitors randomly distributed over the network. The same experiments are repeated in emulation envrionments as well. In Emulation-1, PSBFT is found to be performing on average 51% and up to 55% faster while consuming on average 48% and up to 53% lesser radio-on time in comparison to PBFT. Similarly, in Emulation-2 PSBFT performs on average 68% and up to 80% faster while consuming on average 66% and up to 78% lesser radio-on time in comparison to PBFT.

3.5 Summary

Increasing dependence over the smart-systems not only bring ease in living but also introduces the possibility of unexpected problems because of the presence of smart traitors in the system. Existing consensus mechanisms for IoT/WSN can manage only non-Byzantine faults. To the best of our knowledge, the current work is the first attempt to efficiently realize Byzantine fault resilient consensus strategy for low-power IoT/WSN systems. In general, to build resilience against Byzantine faults, the nodes need to carry out extensive data sharing to exchange opinions with each other. Therefore, the traditional asynchronous transmission-based communication mechanism does not fit well to solve this problem. In this work, we take the help of time-slotted based flooding data sharing strategies to efficiently realize the well-known Byzantine fault-tolerant consensus protocol PBFT for IoT/WSN systems. Our evaluation results show substantial improvement in the latency and energy consumption (radio-on time) in our proposed strategy over naive implementation of PBFT.

Chapter 4

Multi-party computation in IoT

4.1 Introduction

An IoT/WSN-assisted smart-systems, e.g., Wireless-Body-Area Networks [62], Smart-Health-Care [63], Intelligent-Transportation [64, 65], Smart-Grid [66], etc., functions through a massive decentralized collaboration among a large number of IoT/WSN devices. Sharing of data among the devices in such systems play a pivotal role. Due to close interaction of these systems with human-life, the data sensed by the devices may have direct relationship with private/sensitive information. For instance, in an *Advanced Metering Infrastructure* (AMI) [67] system, the real-time electricity consumption data collected by a *Smart-Meter* (SM) from a house can precisely infer the activities inside the house and hence, cause breach of consumer privacy [68]. Such privacy concerns fundamentally create a severe bottleneck in free use of IoT. However, the aforementioned privacy concerns would fundamentally create severe bottleneck in freely exploiting the technology.

To address the above mentioned issue, IoT applications mostly tend to use various forms of aggregation operation instead of directly using the raw values sensed by the edge devices. For instance in an AMI system, its enough for a control center (sink) node to know the aggregate (sum) of the electricity consumption of the individual houses over an area. However, deriving even the aggregated values in a privacy preserving manner is not straightforward. Traditional cryptographic solutions for secure communication facilities are also alone not sufficient to achieve *Privacy-Preserving-Data-Aggregation* (PPDA). There has been a significant research efforts to devise efficient solutions in general. The issue becomes more challenging considering especially under IoT-setting because where a significant fraction of the devices lacks both enough computation capability as well as energy. Traditional cryptographic measures like encryption/decryption, can only secures data in the transition and hence does not solve the problem in discussion. However, they do not provide any security when data are used in collaborative computation. Therefore, we need some other techniques apart from these which allows to carry out the same computation in decentralised fashion keeping data privacy intact. A significant fraction of the existing approaches, to achieve PPDA use *Homomorphic Encryption* (HE) [69] which enables computation of aggregation operations directly over cipher-text. This alleviates the requirement of the deciphering of the data by the intermediate forwarding nodes in a system and hence compute the partial aggregation in a privacy preserving way. HE has been employed in a quite good number of works to achieve PPDA [41, 42, 43, 44, 45]. However HE requires huge computation which cannot be efficiently supported by resource-constrained IoT-devices. In most of the works employing HE, the sink/final destination node needs to know the key to decipher the final aggregation result. This enables the sink nodes to decipher the individual cipher-text of different node. To resolve these issue, various approaches have been taken. *Data-Obfuscation* (DO) by combining random noise with the secret values has been also a common way in many prior works to hide data by each of the nodes. However, most of these works depend on some *Trusted Third Party* for sharing the random noise as well as keys which makes these strategies fundamentally weak. A set of existing works exploits collaborative computing to hide the private data in the form of *Secure-Multi-Party Computation* (SMPC)[12, 13]. *Shamir's Secret Sharing* has been used in many works to realize SMPC through sharing of the secret values in part by the nodes in the first round and then combining the data in the second round to calculate the final aggregation. However, most of these works use extensive data sharing among the nodes which makes them unsuitable for IoT-system.

In this work we propose a novel decentralized lightweight strategy based on collaborative data obfuscation which does not depend on any Trusted Third Party (TTP) or any centralized entity for any purpose and takes no help from cryptographic mechanism to achieve the goal. Most of the works on PPDA so far are either theoretical or show the concept in simulation platform. Very few of the works try to show implementation in real devices. However, all these works mostly use Asynchronous-Transmission (AT) to achieve all the data sharing requirements among the nodes. To more efficiently fulfill the requirements in this work we design the proposed strategy in a way that can exploit the recent advances in *Concurrent-Transmission* (CT) [9] based mechanisms. In a nutshell, in this work we design, implement and rigorously test a practical solution for PPDA in IoT system.

The main contributions of the proposed work are summarized below -

- A lightweight solution for PPDA for resource-constrained IoT-systems with unsecured communication links, has been designed.
- To make it self-sufficient (no use of TTP) a separate module of the protocol is also designed and analyzed for pair-wise key-exchange among all pair of nodes. We acheive the this goal using on;y single round of all-to-all data sharing communication mechanism.
- The proposed design is implemented in C language for 802.15.4 compliant

radio devices and rigorously tested through simulation and emulation platforms.

- We also implement two other state-of-the-art PPDA strategies in our framework and rigorously compare the performance with our proposed protocol.

4.2 Background

Secure Multi-Party Computation (SMPC), a sub-field of cryptography [48], deals with the goal of computing a joint collaborative function of the secret/private values for multiple parties while preserving the privacy of these values. *Shamir's Secret Sharing* (SSS) scheme [49] demonstrates a way to achieve SMPC through a clever application of *Polynomial Interpolation*. In a nutshell, a secret value is decomposed into a certain number of parts called shares and each share is communicated to different node through a secure channel. Thus, when finally, the data from all the nodes come together, the sum of all the secret values are computed while the original values and their origins remain unknown. SSS and other conventional SMPC protocols work well in case of semi-honest adversaries, but it assumes a secure communication link between any pair of nodes in the whole network as well as rely on trusted third-parties for key generation as they use asymmetric key based cryptography. Such assumptions are realistic in usual wired network, but are not suitable for wireless resource constrained IoT/WSN systems. Generally, communication plays a significant role in determining the performance of any MPC protocol as each protocol requires interactions between the participating entities, and thus, we believe that having an efficient and reliable communication infrastructure is valuable. The performance of any MPC protocol considers the number of interactions between the participating entities as one of important factor for protocol performance, and thus, we believe that having an efficient and reliable

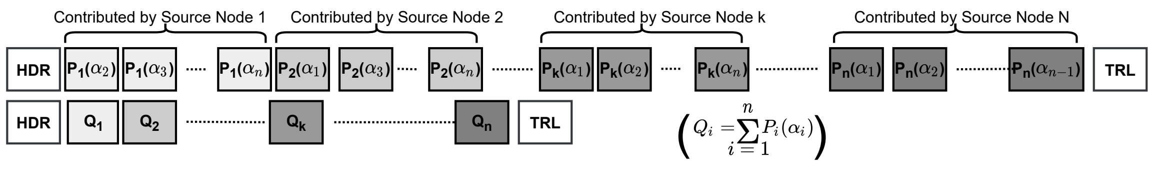

Figure 4.1: MiniCast chain structure during (a) sharing (b) reconstruction round. HDR and TRL represent the header and the trailer packets in the chain, respectively.

communication infrastructure is valuable. Under traditional asynchronous communication based approach achieving many-to-many interaction with high reliability as well as real-time data-gathering / aggregation is hard because of uncoordinated transmissions and hence repeated collision among the packets. Under such many-to-many communication storm, possibility of recurring collision also causes abruptly varying latency for a single packet to get received by different nodes in the system which enhances the chance of leakage of the information and also the number of re-transmissions [70]. Considering all these issues, in this work, we propose to use the capture effect and constructive interference based communication primitives where the transmissions are coordinated and inherently provides the supports for high reliability coupled with low-latency even for many-to-many data sharing. In particular, we select an existing many-to-many data sharing protocol MiniCast for serving as a base. In general, any other efficient strategy for many-to-many/all-to-all data sharing can be used as a base. However, we use MiniCast[17] as a protocol, since it has been shown to perform much better than well-known strategies. Thus, in a nutshell, in this work we aim *to develop a communication assisted decentralized fast and reliable mechanism to aggregate data from all the nodes without revealing their actual data to anyone else in the system.*

4.3 Design

In this work we assume a *semi-honest adversarial* model where every node in the system is supposed to follow the protocol specification correctly. However, the *honest-but-curious* or *passive* adversaries are free to learn the information from the internal states of the other nodes. In summary our proposed solution try to achieve the following:

1. **Privacy:** Preservation of privacy of individual values shared by the nodes in the network even if a set of less than $n-1$ parties collude, i.e., no-one learns the input of any other node in the network.

2. **Correctness:** If all the participating nodes are semi-honest, then the overall aggregated value is a joint function of these private values.

3. **Robustness:** The aggregated value remains intact even if some nodes drop out intentionally from the protocol after initial bootstrapping phase.

4. **Efficiency:** The protocol doesn't assume cryptographic operations except the initial Diffie-Hellman Key exchange, which is also not required for each aggregation round.

Note that a pair-wise secure data sharing mechanism cannot solve the problem. In our approach, a node never shares the original data with anyone specific. Rather, in a nutshell every node shares the data after properly obfuscating it through a masking function designed in a way so that application of the corresponding de-masking function over all the data from all the nodes only can reveal the target joint function value.

For computing a joint function over multiple entities in private manner there need to have a communication among themselves in order to perform privacy pre-

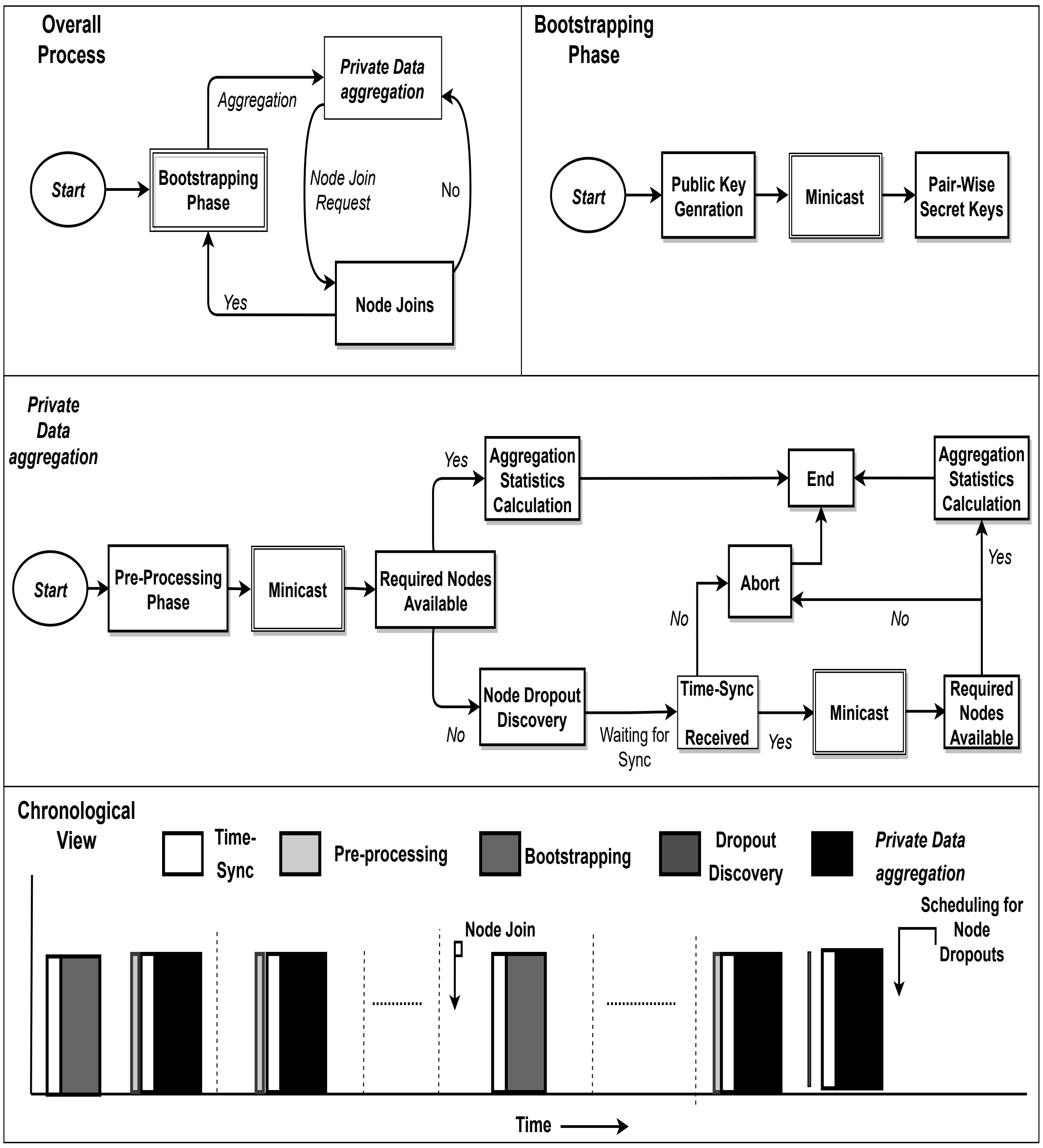

Figure 4.2: Schematic View of end-to-end execution of proposed protocol

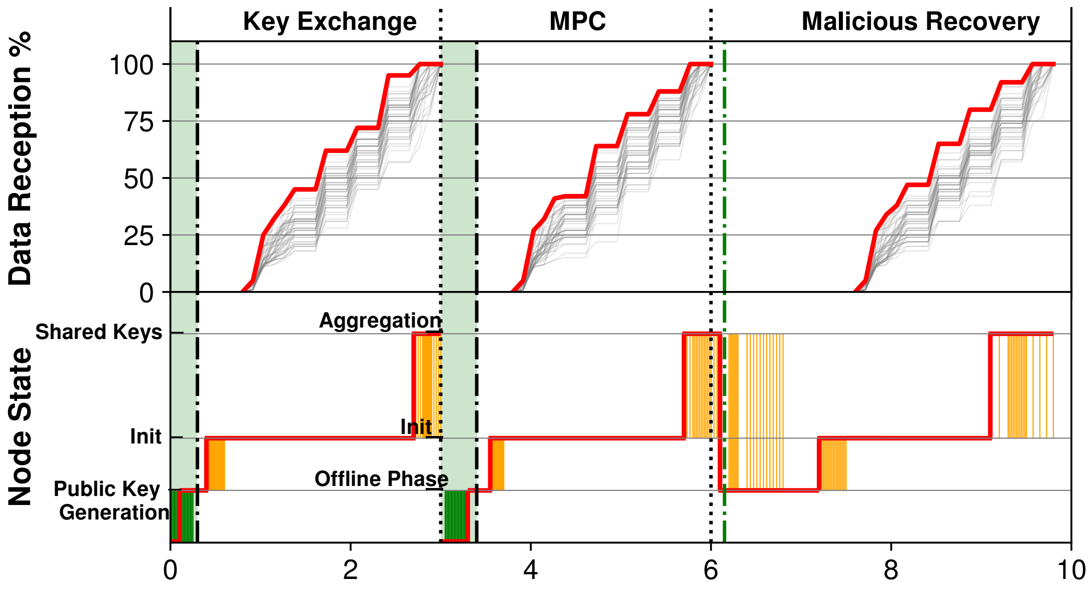

Figure 4.3: **An archetypal view of overall Aggregation Process:** The upper image depicts each node's progress (the Sync node is the thick red line), while the lower depicts their status.

serving computation. Therefore, such communication can be expensive, contributing more to the overall communication cost. To overcome this major bottleneck, we use All-to-all/many-to-many communication strategy, which plays a pivotal role in our proposed strategy. In a resource constrained wireless IoT/WSN setting, many-to-many data sharing is the most complex form of communication[9]. Under traditional asynchronous communication, convergence of a many-to-many/all-to-all data sharing takes a huge time and hence, cannot bring the real-time flavor in our goal. Therefore, we choose to use concurrent communication based flooding protocols to serve the purpose. There are several protocols that carries out efficient many-to-many data sharing with high reliability and low latency. Because of its simplicity and efficiency, we select MiniCast as our base. A brief description of MiniCast is provided below.

MiniCast: MiniCast [17] is designed based on the protocol time-sync and flooding [71] which is a one-to-all data sharing strategy. In one-to-all flooding protocol, a single node disseminates its data to all other nodes in the network. MiniCast enables propagation of one-to-all floods from each different source nodes in the network through a systematic interspersing using a packet-level TDMA mechanism. MiniCast is fundamentally a natural extension of one-to-all data dissemination to support compact time-efficient and reliable data sharing from multiple source nodes. In time-sync[71] a single packet is disseminated from the source node which is forwarded by all the nodes through continuous receptions and transmissions. To support multiple source nodes together, MiniCast enables transmission in chain of packets following a certain TDMA schedule. A transmission/reception of a full-chain of packets is referred to as a *slot* where every node is given a unique sub-slot position. As an explanation of the process, a chain of size X which is the same as the number of nodes that every node is willing to share with each other. A chain of packets is formed based on a TDMA schedule $< s_0, s_1, ..., s_i, ...s_X >$, where each node is given a unique position. The chain is transmitted following the schedule at every transmission where the data from a single node $< x_i >$ is transmitted and forwarded by the nodes in every step. As the outcome of the MiniCast process ideally all the nodes should receive the vector $< d_0, d_1, ..., d_i, ...d_X >$ when all the nodes becomes able to effectively convey the data. If for some reason some node may not be able to receive the full vector, instead receives partial data, e.g., $< d_0, d_1, s_2, ..., d_i, ..., s_{X-1}, d_X >$, where, sub-slot s_2 and d_{X-1} carries out no work.

Apart from benefit of reliability and latency, there is another crucial reason behind selection of these time-slotted concurrent transmission protocols. Its explained immediately after detailed description of the proposed strategy.

4.3.1 Privacy-Preserving Data Aggregation

To keep the protocol simple we assume a semi-honest adversarial model. Consequently, we use unsecured communication channel in MiniCast, i.e., data is transmitted in clear text and hence, each node can see the data shared by any other node. But our underlying algorithm guarantees that a node learns nothing about it until it colludes with *N-1* other nodes. This bound is very strong with regard to developing privacy critical decentralised application. Consider there are N nodes in the system where each node N_i has a private/secret value S_i. The nodes want to jointly compute a function $f(S_1, S_2, ...S_n) = S$. The nodes directly can't send their individual S_i. Therefore, a node computes a value M_i using a function f_1, called a *masking* function used for obfuscation. In the subsequent MiniCast round, N_i inserts M_i in its respective sub-slot. After a successful round of data-sharing and successful accumulation of all the shared values from all the node in the system, a function f_2 is applied over them to compute the value of the aggregation f (i.e., $S = f_2(M_1, M_2, ..., M_i, ..., M_n) = f(S_1, S_2, ..., S_i, ..., S_n)$).

We design a purely collaborative obfuscation procedure. We also do not assume the presence of any trusted third party. In our approach, in the masking process, a node first independently computes a set of values corresponding to the contribution from other nodes (r_{ij}, the contribution computed by node N_i for node N_j). Next, based on the target function f it converts these values using another function f_3, say, $qij = f_3(r_{ij})$, which are finally used in the masking process.

The de-masking procedure hence, depends on the contribution of every node which cleverly serves the goal of computation of the joint function without revealing the secret data form each node to any specific node. For instance, to mask its own secret value S_i node N_i uses the a contribution q_{ij} corresponding to each node N_j (i.e., $M_i = f_1(Si, q_{i1}, q_{i2}, ..., q_{ij}, ..., q_{iN})$). Note that de-masking can be done only when a node has received all the masked values from all the other nodes. Fundamentally, all these values pass through a function f_2 which computes the target joint function f. In this de-masking process with f_2, the effect of q_{ij} is nullified by q_{ji}, which is shared by node N_j and is brought together when N_j's shared value S_j is available while applying f_2. The functions f_1, f_2 and f_3 are designed carefully based on the target function f.

For example, if the target aggregation function is sum of all the secret values, i.e., $S = \sum_{i=1}^{N} S_i$, both f_1 and f_2 are summation functions while f_3 can be defined as $f_3(r_{ij}) = r_{ij}$, if $i < j$, and $f_3(r_{ij}) =$ - r_{ij} if $i > j$. To elaborate, M_i is computed as $S_i + \sum_{i \neq j} q_{ij}$ where $q_{ij} =$ - r_{ij}, when $i > j$ and $q_{ij} = r_{ij}$ when $i < j$. Hence, $M_i = \sum_{j=1,i<j}^{n} r_{ij} - \sum_{j=1,i>j}^{n} r_{ij}$. Combining all the M_i from all the different nodes, f can be computed as follows.

$$\begin{aligned}
\sum_{i=1}^{n} M_i &= \sum_{i=1}^{n}(S_i + C_i) \\
&= \sum_{i=1}^{n} S_i + \sum_{i=1}^{n} M_i \\
&= \sum_{i=1}^{n} S_i + \sum_{i=1}^{n}(\sum_{j=1}^{n} r_{ji} - \sum_{j=1}^{n} r_{ij}) \\
&= \sum_{i=1}^{n} S_i + (\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ji}) - (\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}) \\
&= \sum_{i=1}^{n} S_i
\end{aligned}$$

The overall data aggregation protocol is summarized later.

4.3.2 Sharing secret pair-wise keys

To enable a node to compute the masking contribution from every other node independently (i.e., r_{ij}) the nodes carry out a pair-wise a-priori agreement on secret private key, e.g., node N_i and N_j compute shared secret key as P_{ij}. Subsequently, r_{ij} is computed by feeding $P_{ij} || < \textit{seq\_no} >$ as seed to a *Pseudo Random Number*

Algorithm 3 BOOTSTRAPPING PHASE

```
Require: $p$, $g$, $\mathbf{S} = \{N_1, N_2, ...N_n\}$
Ensure: $\forall i, j \in S$ such that $i \neq j$ $\exists$ Common Secret Key $S_{ij}$
1: for $i \leftarrow 1$ to n do
2:     $Minicast\_1[i] \leftarrow g^{d_i} mod\, p$
3: end for
4: for all $N_i \in S$ do
5:     for $j \leftarrow 1$ to n do
6:         $Pub\_Key_j \leftarrow Minicast\_1[j]$
7:         $\mathbf{S}_{ij} \leftarrow (Pub\_Key_j)^{d_i}\ mod\, p$
8:     end for
9: end for
```

Generator (PRNG) function. For example, node N_i generates the contribution for node N_j, i.e., r_{ij} as follows $r_{ij} = PRNG(P_{ij}, seq\_no)$, similarly, node N_j, generates for Node N_i, i.e., r_{ji} as follows $r_{ji} = PRNG(P_{ij}, seq\_no)$. Note that here seq_no is the common sequence number used by all the nodes. This sequence number is borrowed from the underlying network communication framework. In general, the flooding based protocols execute in a periodic fashion where in every period a common sequence number is incremented by the initiator and shared by all the nodes. Thus, the same has been exploited here. It also results in enhancing the output randomness on every iteration of aggregation protocol.

In order to establish the prior-agrement on the pair-wise secret key we run an instance of all-to-all data sharing by MiniCast to do a simultaneous pair wise *Diffie-Hellman Key Exchange*[72] process.

4.3.3 Application of the proposed strategy

Here we discuss some applications our proposed approach.

Algorithm 4 PRIVACY-PRESERVING DATA AGGREGATION

Pre-Processing (Offline) Phase

```
Require: $\forall\ N_i, N_j \in S = \{N_1, N_2, N_3, ...N_k\}\ \exists\ S_{ij}$
Ensure: $\forall i \in \{1, 2, 3, ..., N\}\ \exists\ \mathbf{MASK}_i$
 1: for all $N_i \in \{N_1, N_2, ....N_n\}$ do
 2:     for $j \leftarrow 1$ to n do
 3:         if $j \in (i+1, N)$ then
 4:             $N_i.r[j] \leftarrow S_{ij}$
 5:             $N_i.c[j] \leftarrow Flip\_Bits(S_{ij})$
 6:         else
 7:             $N_i.r[j] \leftarrow Flip\_Bits(S_{ij})$
 8:             $N_i.c[j] \leftarrow S_{ij}$
 9:         end if
10:         $\mathbf{MASK}_i \leftarrow \mathbf{MASK}_i \bigotimes PRNG(N_i.r[j])$
11:                         $\bigodot PRNG(N_i.c[j])$
12:     end for
13: end for
```

Aggregation (Online) Phase

```
Require: $\forall\ N_i \in \{N_1, N_2, N_3, ...N_k\}\ \exists\ S_i\ \&\ \mathbf{MASK}_i$
Ensure: $Y = f(x_1, x_2, x_3, ..., x_n)$
 1: for $i \leftarrow 1$ to n do
 2:     $Minicast\_2[i] \leftarrow S_i \bigotimes \mathbf{MASK}_i$
 3: end for
 4: $Y_k = 0$
 5: for $i \leftarrow 1$ to n do
 6:     $Y_k = Y_k + Minicast\_2[i]$
 7: end for
```

- **Quasi-Arithmetic Mean**

  If f is a function which maps an interval I of the real line to the real numbers, and is both continuous and injective, the $\boldsymbol{f}$-mean of n numbers $x_1, \ldots, x_n \in I$ is defined as $M_f(x_1, \ldots, x_n) = f^{-1}\left(T^{-1}\left(\frac{T(f(x_1))+\cdots+T(f(x_n))}{n}\right)\right)$, which can also be written

  $$M_f(\vec{x}) = f^{-1}\left(T^{-1}\left(\frac{1}{n}\sum_{k=1}^{n} T\left(f\left(x_k\right)\right)\right)\right) \qquad \text{(Eq. 1)}$$

  where, T is transformation function which obfuscate private value x_i.
  For this to happen, f need to be injective in order for the inverse function f^{-1} to exist. Since f is defined over an interval, therefore $\frac{f(x_1)+\cdots+f(x_n)}{n}$ lies within the domain of f^{-1}.

  1. *Arithmetic Mean*:

     It measures the central tendency of the distribution and thus proves to be helpful in several statistical analyses. With reference to the aboveEq. 1, the arithmetic mean of can be calculated if $f_x = x$. In our case, $f(S_i) = S_i$, where S_i is the secret valve for every participating node.

     $$f(S_1, S_2, \cdots, S_N) = \frac{\sum_{1=1}^{N} S_\mathrm{i}}{N}$$
     $$f_1(S_i) = M_i = S_i + MASK_i \left(\bigotimes = +, \bigodot = -\right)$$
     $$f_2(M_1, M_2, \cdots, M_N) = \sum_{1=1}^{N} S_i + MASK_i \qquad = \sum_{1=1}^{N} S_\mathrm{i}$$

     $$f_3(q_{ij}) = \begin{cases} \text{PRNG}\,(S_{ij}) + \textsc{Flip\_Bits}(S_{ij}) & i \leq j \\ \textsc{Flip\_Bits}\,(S_{ij}) \text{ - } \text{PRNG}(S_{ij}) & i \geq j \end{cases}$$

  2. *Geometric Mean*:

     The geometric mean of a data set $\{a_1, a_2, \ldots, a_n\}$ is given by:

     $$\left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} = \sqrt[n]{a_1 a_2 \cdots a_n} \qquad \text{(Eq. 2)}$$

The geometrical mean also measures the central tendency, but it is more suited for data whose values are multiples in nature or exponential to each other. In such a case, the arithmetic mean fails to give the actual central tendency. We can achieve the same privacy-aware if $f(x) = log(x)$.

Apart from the above example, statistical measures, *harmonic mean, power mean, log semiring,* etc., can also be measured privately from the data generated by the sensor nodes.

- **Linear Regression Inference**:
A linear regression model assumes to have a linear relationship between the dependent variable y and k-vector set of regressors $\mathbf{x}$. We assume to have a trained linear model over a dataset of n datapoints i.e. $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors is linear. This relationship is modeled through a disturbance term or error variable ε - an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \ldots n \qquad \text{(Eq. 3)}$$

where $^\top$ denotes the transpose, so that $\mathbf{x}_i^\top \boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol{\beta}$. Often these n equations are stacked together and written in matrix notation as

$$\mathbf{y} = X\boldsymbol{\beta} + \varepsilon$$

4.3.4 Robustness

Being an solution originally for semi-honest adversarial model, we provide some robustness

Under a semi-honest adversarial model, the nodes would only try to passively learn the secret values which is already shown above to be not possible in our approach. Here we think of two possible situation which can be cause by either intentional or unintentional behaviors by the nodes. Leveraging the underlying communication mechanism we design our protocol in a way so that it can successfully efficiently defend both the situations.

Node failure after initial participation

For some reason if after initial participation in the protocol, fails to continue to complete the data sharing process to acquire the data from all the nodes completely, the protocol execution will remain unaffected in all the remaining nodes. This becomes possible because of the underlying architecture of MiniCast which allows the nodes to put their data in their respective slots and once inserted appropriately and is received by at least one 'healthy' node, the value lives in the process and eventually all the nodes can receive it and correctly compute the joint function. However, its quite true and natural that since MiniCast depends on multi-hop data forwarding, such resilience against node-failure will be limited upto a certain extent.

Non-participation in the aggregation process

We are achieving privacy preservation through collaboration. Participation of a node x in the KeyExch process can be considered to be the consent from x about its active participation in that collaborative effort. However, due to some reason, a node may not participate after agreeing. The reason maybe some malicious intention to make the computation incorrect or it may also be due to some unexpected fault. To make our system robust against such situations, we exploit the underlying communication mechanism. MiniCast uses a chain/TDMA schedule for data sharing. Thus, if a node x after agreeing does not participate in the process, its sub-slot in the chain will be empty. From this all the nodes can first infer that the computed value is wrong as the data everyone shared was calculated through f_1 considering the presence of x. From the id of the empty slot itself everyone can re-calculate their data and re-share again in an subsequent recovery MiniCast round. But, if it is done without a systematic manner, there can be following two issues. (a) If node x's non-participation was due to simple fault, it may again start participating in the subsequent recovery round and confuse the other nodes. (b) If it was a malicious step by x, it may try to participate in the recovery round[1]. To solve this issue, a message from the initiator is disseminated confirming the ids of the nodes to be ignored in the subsequent recovery round. In our approach we introduce a one-to-all dissemination [71] in between two instances of MiniCast.

4.3.5 Protocol framework

Based on the ground as depicted above, here we provide a crisp description of our design of the proposed protocol. Although the pair-wise key exchange phase (*KeyExch*) is a very important component of the proposed work, it is used only at a pre-defined interval or when needed. The overall process runs in a periodic fashion as other communication based protocols[9]. Once the pair-wise keys are established, the actual process of computation of the joint function starts from the subsequent period. To cover all the possible cases as depicted above, two back-to-back full MiniCast instances are executed with a time-sync[71] phase in the beginning of both. The first phase is used for co-ordination of overall network

[1]To keep it simple, we never change the schedule size in MiniCast.

while the second phase is used as a means of conveying the message to the nodes about the nodes that are missing in the previous round so that in the subsequent instance of MiniCast, the nodes can provide appropriate their adjustment values accordingly. The time-diagram and the flow-diagram are provided in Figure 4.2, respectively. Note that the *KeyExch* phase is executed after a certain number of computation phases are over to renew the keys. It can be also revoked when a new node joins in the system as shown the flow-diagram.

4.3.6 Time-slotted based multi-flooding vs Asynchronous Communication:

The proposed strategy in this work is actually independent of any specific communication mechanism. It can run even with asynchronous mechanism for all-to-all data sharing. However, apart from latency issue to be a significant aspect, there can be some problem with in the true preservation of privacy of the data under asynchronous setting even under semi-honest adversarial model. This may happen mainly due to the issue of the possibilty of unbounded delay in asychronous domain. For example, in some round, the data from a node x may take quite a large time to get disseminated and received other nodes. Due to expiry in the waiting time, the initiator may start the confirmation flood mentioning the ids of the node which are to be ignored. Subsequent recovery round of the all-to-all data sharing will enable the nodes to compute a correct data. However, in case the old data from node x if comes after the recovery round is over, it will allow the nodes to reveal the secret value of x which is not expected. Under concurrent transmission based implementation such situations are strictly prohibited due to highly time-bounded operations in all the implementation of the strategy. Therefore, it was one of significant motivating factor for us to adopt time-slotted based protocols for fulfilling the communication requirement of this work.

4.3.7 Scalability

Continuous Group Formation

According to the naive execution of Algo 4, every nodes incorporates a random value outputted from the SHA using pair-wise keys as seed to it. However, such technique do provide both high security and fault-tolerance despite being designed for semi-honest adversarial model. Nevertheless, as the number of participating nodes grow, the overhead also increases in terms of storing secret pair-keys and asking 100% reliability from underlying communication protocol, Minicast. However, if we don't have complete reliability, still we do guarantee consistent aggregation statistics through dropouts recovery mechanism. Nevertheless, if the network is sparse, then the chances of exchange of data between nodes sitting at two extremes becomes challenging. However, the neighboring nodes from both the nodes consistently receives the data. Moreover, such a scenario is seen with most of the concurrent transmission based protocols[9]. We exploit this fact and modify the way we can use our proposed methodology to achieve data aggregation.

In MiniCast, the parameter NTX primarily influences the degree of coverage/reliability achieved by the dissemination process. When the NTX is large enough, a node receives data from all the other nodes in the network. MiniCast, on the other hand, exhibits highly reliable and constant behaviour even at low NTX, despite failing to gain complete network coverage. Depending on the precise amount of NTX, a node successfully gets data from its neighbour within a specific perimeter. The behaviour is quite non-linear, in that a big amount of data becomes available at a node with a small increment in NTX, but it takes a considerably longer period (NTX) to achieve complete network coverage.

We use MiniCast's above-mentioned behaviour to optimise the execution of our proposed technique. In summary, we assume a low level of masking. To increase speed, not only is pair-wise key storage minimised, but the data exchange operation is also run at a low NTX value. To do this, during the bootstrapping phase, each node is supposed to keep track of which neighbours are accessible at what NTX value. Using this information, the chain is built in the sharing phase so that each node distributes evaluation values for a few known pre-determined neighbours. The procedure completes quickly and with little NTX, and then enters the rebuilding phase to finish the process.

Spatial Group Formation

Currently, our designed solution can tolerate up to n-1 collusions. However, to achieve this security level, we need to consider the participation from all the nodes. Nevertheless, our protocol design is so elegant that it can be applied to more extensive networks. For instance, we are currently considering all the nodes in a single group, and each node is incorporating the masking values for all other group nodes during the obfuscation process. However, A large network can be split into multiple groups spatially; therefore, each node can be assigned to one of the group(s). This spatial division can be done using bootstrapping phase where we run multiple time-sync floods from multiple initiators. The initiator starts the flood by mentioning the group number. Later on all the nodes who belong to the same group runs one round of minicast protocol to acheive the pair-wise secret keys among themselves. Earlier, all the nodes were keeping pair-wise keys for all the nodes in the network. However, currently each nodes needs to track the keys for all its group members.

Consequently, each node now runs Algo. 4 to incorporate masking values from each node who are in its group rather than considering all nodes. If a network of N nodes is split into M gropus with each group have nearly K nodes, then the security threshold would reduce to k-1 collusions. Moreover, the chain length in spatial division results in O(k) size. Overall, it can be seen that the same designed protocol can be applied hierarchically also to alleviate the scalability concerns.

4.4 Evaluation

4.4.1 Shamir-Secret Sharing (SSS)

SSS is a well-known approach for performing privacy-preserving compute activities in a multi-party scenario, which closely relates to the needs of massive IoT networks. It divides a secret into sub-parts known as shares. These shares are subsequently relayed to other network participants. In addition, other parties transfer their shares in a network. Finally, these shares are combined together and re-shared to compute the desired computation function. Furthermore, SSS is a threshold-based cryptography technique in which the minimum number of shares is required to reconstruct the secret. If any adversary owns less than these threshold shares, the secret is protected under the concept of perfect secrecy. Formally, SSS is a generalization of the one-time pad, which is basically SSS with a two-share threshold and two total shares. It achieves privacy-preserving data aggregation using polynomial interpolation over finite fields with information-theoretic security. SSS uses *threshold-based cryptography* to ensure the requirement of a minimum number of shares to reveal the secret value held by a node. A (t,n) secret sharing divides a secret into n shares in such a way that any t or more than t shares can reconstruct the secret; but fewer than t shares cannot reconstruct the secret. In SSS, every participating node divides its secret value into a certain number of sub-parts known as *shares*. These shares are subsequently, relayed to other participants in the network. Finally, each node combines the received shares in a defined way and computes the desired target function. Thus, at least some finite shares are necessary to reconstruct the sum of n secret values. Yet, if any adversary owns less than these threshold shares, the secret is protected under the concept of perfect secrecy. Formally, SSS is a generalization of the *one-time pad*, where the threshold is set to two shares. SSS uses polynomial interpolation over finite fields for all necessary computation. SSS$(k+1, n)$ at least at least $k+1$ shares are necessary to reconstruct the sum of n number of secret values.

In brief, the strategy works as follows, Let every participating node n_i have its secret value S_i. SSS uses polynomial interpolation over finite fields for all necessary computation. In brief the strategy works as follow. A node n_i, having a secret value S_i, first decides it's k-degree polynomial P_i with coefficients a_i, b_i,..., S_i. Thus, the secret value can be computed as $P_i(0)$. Next, every node evaluates its own polynomial at a set of n public points, α_1,α_2,...,α_k,..., α_n, as, $P_i(\alpha_1)$,$P_i(\alpha_2)$,...,$P_i(\alpha_k)$,...,$P_i(\alpha_n)$, where n is the total number of participants. Every node n_i keeps its share $P_i(\alpha_i)$ with itself and communicates the remaining $n-1$ shares to $n-1$ distinct nodes through end-to-end secure channels.Every other party in the network does the same. Thus, a node n_j receives only one distinct share of the polynomial P_i from a node n_i, through a secure channel. Node n_j would receive shares of the polynomials from other nodes too. These individual values are not enough to reconstruct any of the polynomials. Once this initial sharing is over, each node locally sums up the received values and re-shares the summed value with other nodes. Using the finally shared values, a new polynomial of degree k is formed by the *Lagrange Interpolation* technique. The constant term in this polynomial represents the desired sum of the secret values over all the nodes. These final polynomial calculations can be efficiently realized using the Lagrange Interpolation technique.

$$P_1 = a_1 x^k + b_1 x^{k-1} + \ldots\ldots + S_1 \tag{4.1}$$

$$P_2 = a_2 x^k + b_2 x^{k-1} + \ldots\ldots + S_2 \tag{4.2}$$

$$P_3 = a_3 x^k + b_3 x^{k-1} + \ldots\ldots + S_3 \tag{4.3}$$

$$\vdots$$

$$P_{n-1} = a_{n-1} x^k + b_{n-1} x^{k-1} + \ldots\ldots + S_{n-1} \tag{4.4}$$

$$P_n = a_n x^k + b_n x^{k-1} + \ldots\ldots + S_n \tag{4.5}$$

$$P_{final}(x) = \sum_{i=1}^{n} P_i(x)$$

,

$$P_{final}(0) = \sum_{i=1}^{n} S_i$$

The same can be easily written using the Lagrangian Interpolation Scheme:

$$f(0) = \sum_{j=0}^{k-1} y_j \prod_{\substack{m=0 \\ m \neq j}}^{k-1} \frac{x_m}{x_m - x_j}$$

Example: Consider an example of 5 nodes who wish to carry out the aggregation task involving an individual's secret values. Now, each one chooses its polynomial P_i and shares the respective with other parties. Technically, each node places its corresponding shares in the Minicast chain. The chain structure can be generalized to N nodes as shown in Figure 4.1. We divide the aggregation process in two phases: Sharing and Re-construction Phase. In sharing every node communicates its polynomial shares following the chain structure shown in Figure 4.1(a). Moreover, in Re-construction pahse every nodes shares its local sum following the chain structure as highlighted in Figure 4.1(b). For instance, if N nodes are willing to carry out aggregation task, then for sharing phase the chain length would be $N \times (N-1) \approx \mathcal{O}(n^2)$. However, it would be $\mathcal{O}(n)$ for re-construction phase.

4.4.2 Neighbourhood based Shamir Secret Sharing (NSSS)

The naive adoption of SSS results in O(n^2) communication time as the number of the packets that need to be communicated in the network are quadratic. However, it is so because every participating node is forwarding the its polynomial shares to all other nodes in the network. But, when the degree of polynomial is significantly less than the total number of nodes i.e. $d < N$, then such communication complexity results is high overhead despite the fact that any set of $d + 1$ nodes can reconstruct the resulting polynomial. We exploit this fact and proposed a scalable approach in order to achieve SSS in large scale IoT networks. We tweak the current technique in order to reduce the communication time from O(n^2) to O(nk), where $d \leq k \leq N$. The intution behind this optimisation is the exploit the underlying linearity property in SSS.

The initiator starts the all-to-all/many-to-many process and evaluates its d degree polynomial at k , ($k \geq d$) and places these shares into the respective slots in chain after encrypting them with AES-128[2]. When the first hop receives the chain they can take out intiator shares if they belong to them, otherwise they can evaluates their own polynomials and places their corresponding shares in the chain and transmits the modifies chain. This process keeps on repeating for several iterations defined by a parameter NTX. NTX controls the percentage of data flow in the network. As in this step every node intends to share its data mostly with its neighboring nodes. Therefore we need significantly less value of NTX compared to the case of naive execution of SSS. Each node in this phase use d degree polynomial. This phase would result in chain length of $O(nk)$. After the former sharing process, the nodes locally decrypts the shares belong to them and locally sum these shares.

[2]However, we do not need encryption if nodes chooses k degree polynomial with k neighbours as it will generate $k + 1$ shares, but it shares only k shares and keeps 1 share with itself. So, it is impossible to reconstruct its polynomial using any set of collusions using these k shares.

Now, as all the nodes starts with same degree polynomial and also evaluates it same k public points. Therefore, if the nodes shares their aggregated shares then it would eventually becomes the shares of

$$P_{final}(x) = \sum_{i=1}^{n} P_i(x)$$

, wherein

$$P_{final}(0) = \sum_{i=1}^{n} S_i$$

. This reconstruction phase would result in the chain length of $O(n)$. Our reconstruction phase is all-to-all data sharing phase, therefore all the nodes can reconstruct the final aggregation value by interpolating the final polynomial from these shares in final chain. We implement the proposed strategy in C language for 802.15.4 compliant radio devices and rigorously tested through simulation and emulation platforms. We simulate the same in MSPSIM and ns-2 as well as carry out experiments with the same in two emulation settings using sensor nodes having ARM, e.g., Emulation-1 and Emulation-2., which contain 24 and 48 devices, respectively. We vary the network size and area of deployment for our experiments.

4.4.3 Parameters and Metric

We use the following metrics to measure the efficiency and robustness of our proposed strategy.

- **Latency:** It is the time taken for a node to get the consistent result of the desired statistical measure. We assume it to be an end-to-end latency as it considers both online and offline executions of the protocol. At network level it indicates the average latency over all the nodes.

- **Radio-on time:** Total time for which the node keeps its radio on for the data sharing operation and hence, depends on the MiniCast runs. It depends on the value of NTX. It is also averaged over all the iterations of the protocol.

We specifically define two variations of these metrics viz. sync_(latency/ROT) and all_node_(latency/ROT). It basically means how much time is taken by the controller node and all nodes to get the final results. Aside from these two indicators, we also use the metrics *Data Reception Percentage*(DRP) and *Intermediate Measure*(IM). *DRP* highlights the data received from the percentage of participating nodes whereas, *IM* shows the value of the desired statistical measure at each subsequent packet reception. The number of nodes either fails to turn on the radio after initial transmission or intentionally dropped out from the desired interaction round are termed as Faulty percentage.

We run each experiment for at least 1000 iterations and computer the metrics as an average over all the iterations and the nodes. The error bars in the results show the standard deviations over the iterations.

In particular, we first show the analysis of our proposed strategy and later compare its performance with SSS, NSSS and PPMP [46].

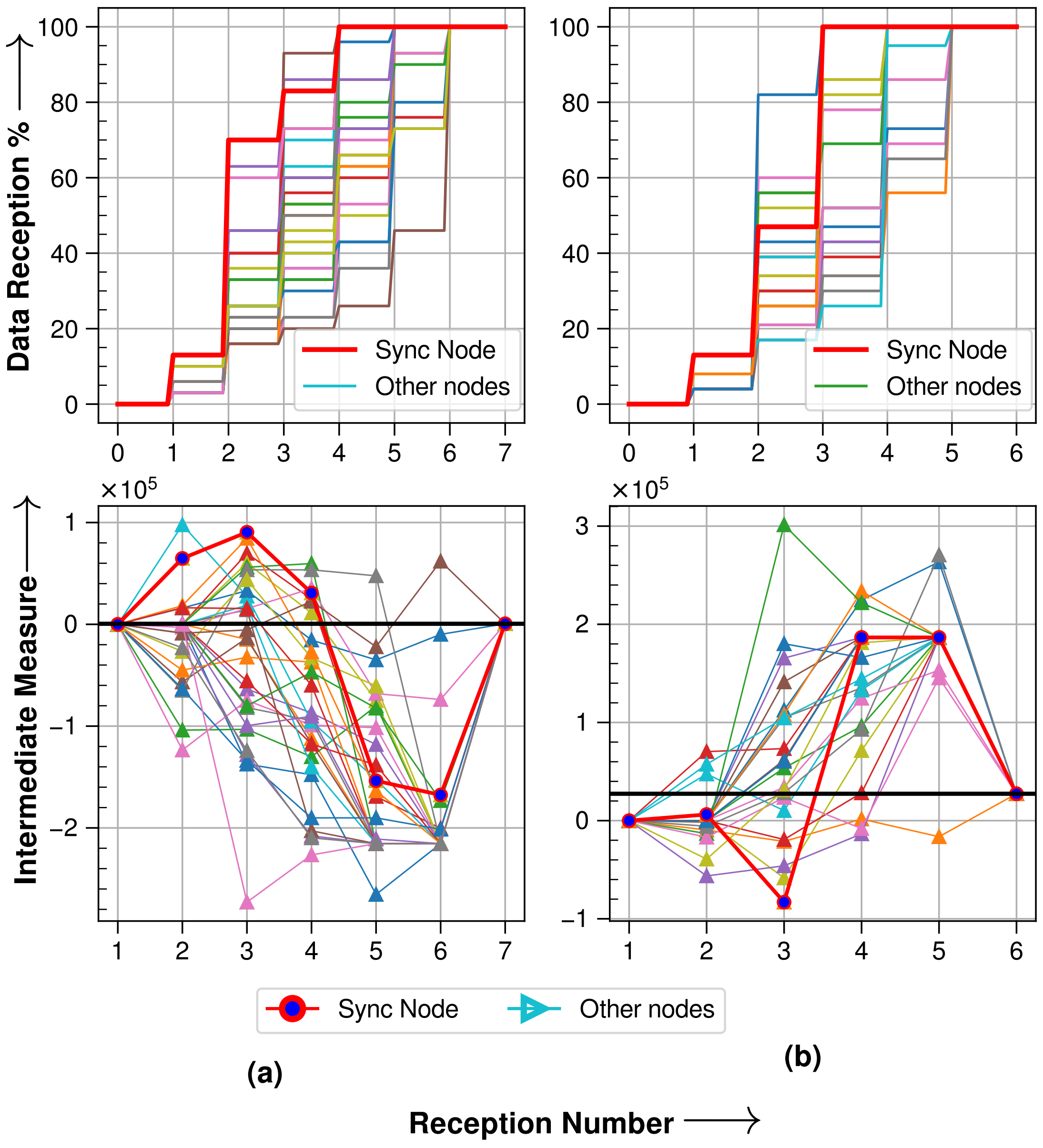

Figure 4.4: Intermediate Sum and network coverage in Emulation-1 and Emulation-2 environments

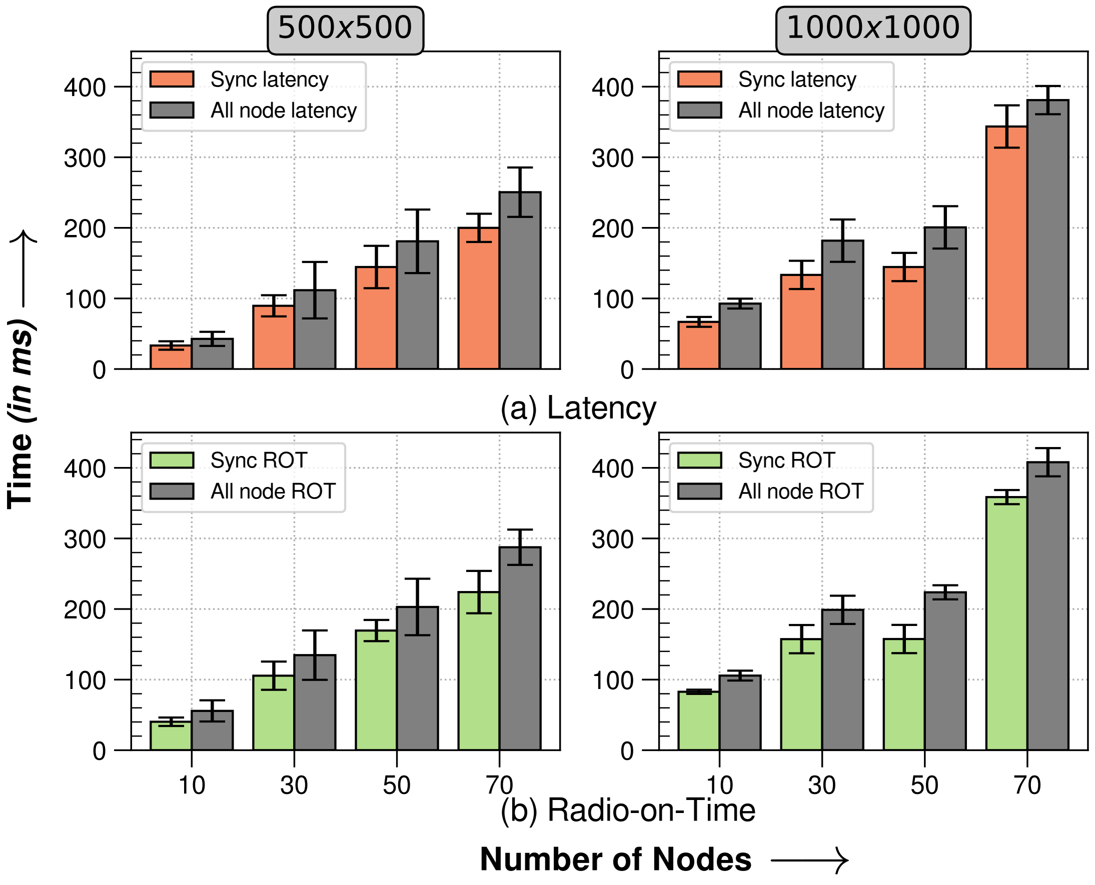

Figure 4.5: **Normal Execution:** The increment in total number of nodes*(N)* and the network diameter *(mxn)* directly influence both Latency and Radio-on time (in ms). The sync latency/ROT is measured for keeping in mind the state of only Sync node whereas, Overall Latency/ROT is the time for the network-wide calculation of the final result.

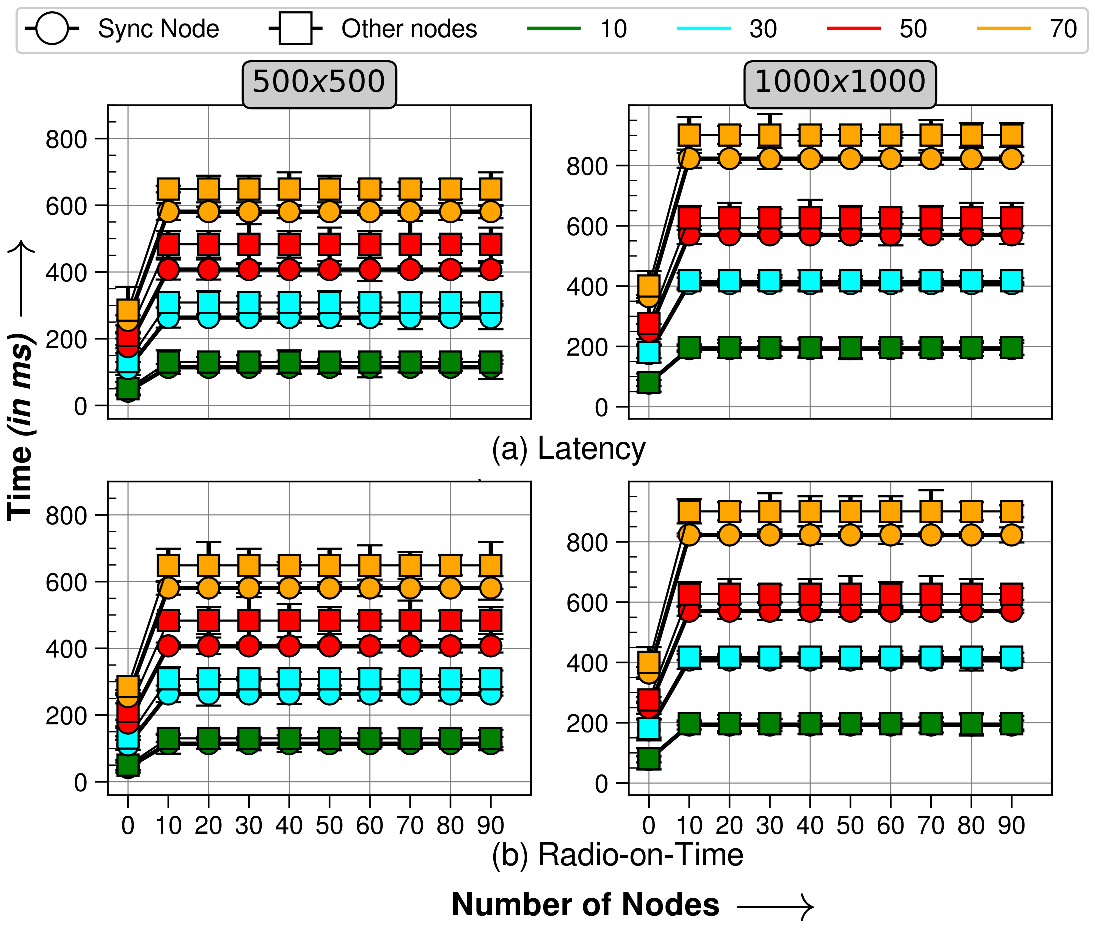

Figure 4.6: **Malicious Execution:** The increment in total number of nodes*(N)* and the network diameter *(mxn)* directly influence both Latency and Radio-on time (in ms). The sync latency/ROT is measured for keeping in mind the state of only Sync node whereas, Overall Latency/ROT is the time for the network-wide calculation of the final result.

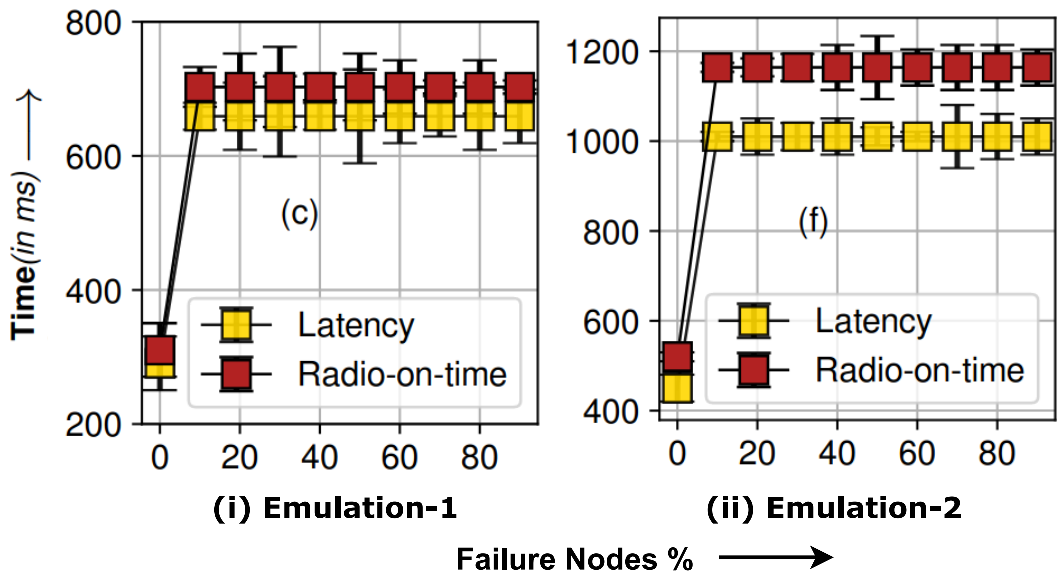

Figure 4.7: Malicious Execution in Emulation Settings

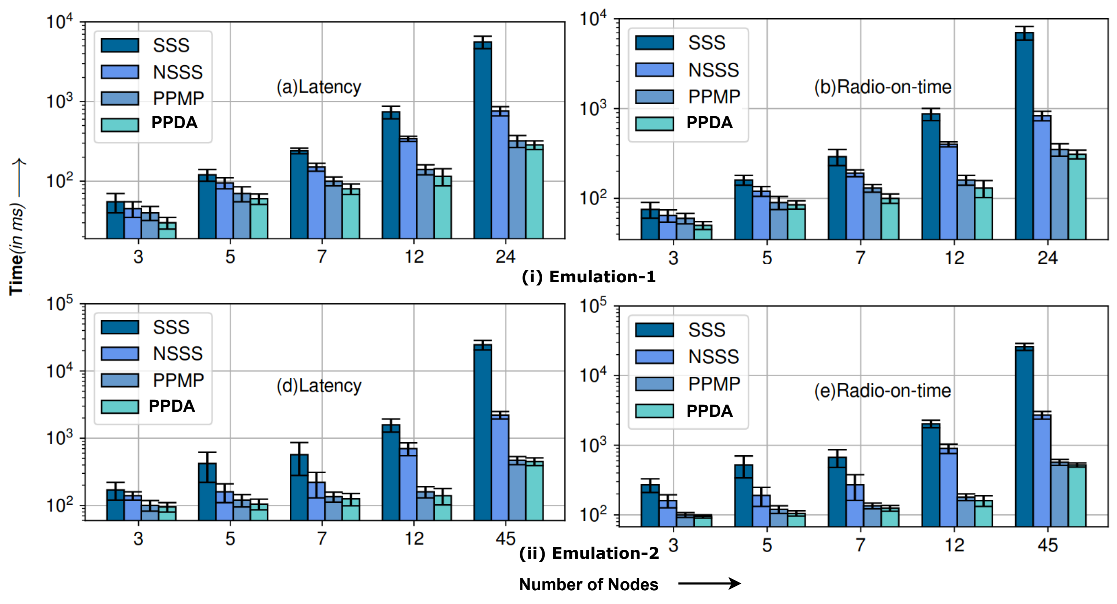

Figure 4.8: Comparison of our proposed protocol with other state-of-art strategies

4.5 Summary

In this work, we propose an efficient and robust privacy-preserving data aggregation in low-power IoT-systems with the help of concurrent transmission. As per the knowledge of the authors, this is the first attempt in this direction. Moreover, the work brings a set of useful insights. The proposed mechanism shows promising results for low-power devices in terms of both overall completion time of the process as well as the energy consumption. Use of any encryption technique consumes a huge space in the packets which ultimately inflates the energy consumption heavily in the nodes. Howoever, we avoid the the usage of any such cryptographic strategy that is always preferable in resource-constraint settings. Moreover, message passing complex- ity of the base strategy used here is O(n^2) where n is the number of source nodes. With such huge message passing even efficient CT based strategies fail to bring a scalable solution. Thus, to solve the issue, we propose an $O(n)$ solution. We also utilise the unique features of our underlying communication to make our data aggregation robust against node dropouts and data privacy. In this current work, we work in semi-honest adversarial model, therefore we would like to transform the same developed mechanism for active adversaries and considers it as immediate future work in this direction.

Chapter 5

Conclusion

While in operation, IoT-based smart systems must give adequate assurances such as security, privacy, resilience, etc. The absence of either of these undoubtedly poses several risks to both users and the network itself. However, meeting such standards is not easy because an practical IoT network consists of several devices, as shown in,3.1 and behaves differently from typical wired networks. The key challenges are how devices can efficiently interact with one another to achieve any stated set of goals in the face of internal or external prospective adversaries. As a result, ensuring the security guarantees for a network comprised of heterogeneous IoT devices is difficult. However, we believe such characteristics are necessary and need to be fulfilled for critical applications. Therefore, we consider all requirements and develop our protocols with these objectives in mind. We specifically created protocols for computing aggregate statistics on the data of participating IoT nodes while maintaining privacy. In addition, we provide a service to make IoT networks reliable/trustworthy by enabling computation to occur in the face of Byzantine failures. To make it time and energy-efficient, we introduced a couple of optimizations over the naive strategy based on the observation that we do not need the highest degree of privacy protection or fault tolerance for practical cases.

Chapter 6

Future Works

6.1 Security in Flooding based communication

In the current state of this work, we assume some restrictions on the malicious behaviour of participating nodes. We do it because of weaknesses in our underlying communication primitives. Flooding based concurrent communication-based protocols, completely rely on the strict schedules for transmission/reception of packets. Thus, any disturbance in the time in any particular node or group of nodes can completely make the protocols fall apart from the desired communication process. The victim nodes may send/receive the packets at unexpected time slots. This would not only waste time but also drain the battery power without doing any useful work. While under asynchronous communication, there are many solutions present for secure communication, the same for counterpart is largely missing. Earlier works tries to make the fundamental one-to-many flooding protocol secure, also does not guarantee the prohibition of injection of false data by some legitimate/authorized node in the network. Moreover, a set of malicious nodes may coordinate with each other to get a subset of a network flooded with false data despite the use of the proposed security measures. The proposed one-to-many solution becomes infeasible in case of many-to-many flooding as any particular malicious node can target several legitimate nodes and then modifies their data before transmitting which in turn results in an inconsistent network state. We try to address these issues for many-to-many flooding-based protocols in both the MAC layer and Network layer using concepts from both symmetric and asymmetric key cryptography. The raw use of available cryptographic algorithms poses a great challenge in these tiny devices due to the lack of on-device computation resources. The use of cryptography also poses an extra challenge by introducing extra overhead in the allocated slot time. This extra time will get amplified in case of many-to-many as compared to one-to-many. Thus, our aim here is the development of optimized algorithms that satisfies both data integrity and doesn't disturb the overall time with a great margin. The final goal is to develop end-to-end secure versions of both one-to-many and many-to-many data communication protocols. On the same side, we did a preliminary study on these data perturbation activities and have seen the following impact of it in network.

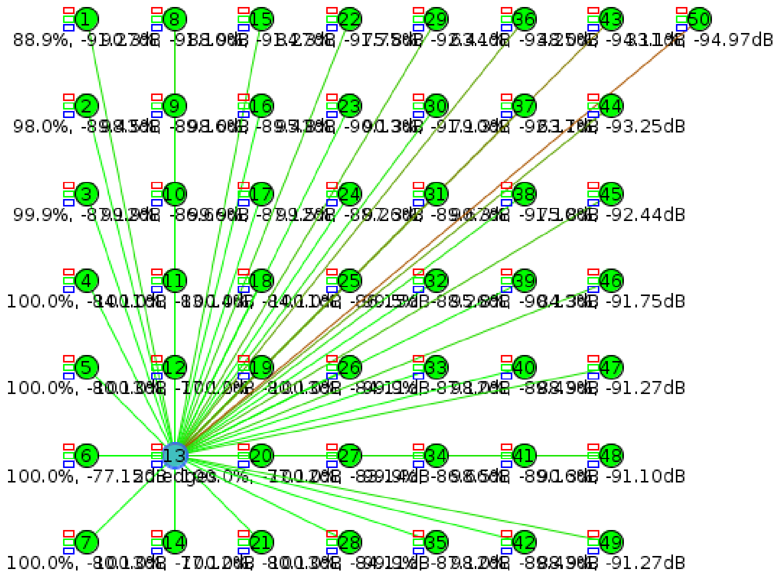

Figure 6.1: A grid layout of 50 nodes with network area 500×500 m^2

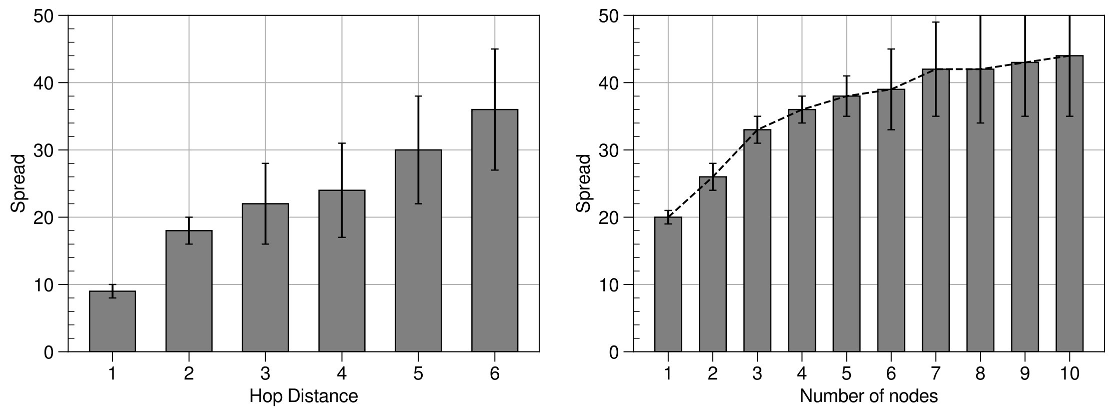

Figure 6.2: False data spread in an 50 node network with single and multiple attackers, respectively

6.2 Privacy-Preserving data aggregation in Active adversarial model

Our proposed solution for private data analytics uses Multi- party computation (MPC) to ensure semi-honest security. However, we are curious how such rich computation tasks can be attained in malicious environments. We are interested in using approach of proof systems to counter malicious behaviors during secure computation. We are curious in knowing how these proof systems can be designed for a Multi-party setting amidst arbitrary failures and incomplete communication networks. Moreover, we would like to explore the theoretical foundations of MPC to build efficient protocols that can provide optimal security with a minimum performance overhead. Additionally, we would love to study IoT systems utilising private computetion techniques from the adversarial perspective. We believe it would help us learn how to provide provable security guarantees after examining underlying implementation weaknesses to reason about the security, question assumptions of deployed algorithms, and under- stand potential model threats. As a result, it would help us design robust Internet-of-Things or Wireless-Sensor-Networks.

Dissemination of Research Results

1. Himanshu Goyal, Sudipta Saha. Multi-party computation in IoT for Privacy Preservation - Accepted in 42nd IEEE International Conference on Distributed Computing Systems (ICDCS), 2022, Bologna, Italy.

2. Himanshu Goyal, Sudipta Saha. LiPI: Lightweight Privacy-Preserving Data Aggregation in IoT - Under Review, IoT Journal.

3. Himanshu Goyal, Sudipta Saha. Practical Synchronous Byzantine Consensus for Internet-of-Things - Under Review.

4. Himanshu Goyal, Sudipta Saha. DivConMPC: Divide and Conquer based Privacy Preserving Multi-Party Computation in IoT - To be submitted in the conference publication.

Bibliography

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of things security and forensics: Challenges and opportunities," 2018.

[3] F. B. de Oliveira, *On Privacy-Preserving Protocols for Smart Metering Systems: Security and Privacy in Smart Grids.* Springer, 2016.

[4] E. L. Quinn, "Privacy and the new energy infrastructure," *Available at SSRN 1370731*, 2009.

[5] S. Tan, D. De, W.-Z. Song, J. Yang, and S. K. Das, "Survey of security advances in smart grid: A data driven approach," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 397–422, 2016.

[6] M. T. Moghaddam and H. Muccini, "Fault-tolerant iot," in *International Workshop on Software Engineering for Resilient Systems*, pp. 67–84, Springer, 2019.

[7] M. A. Uddin, A. Stranieri, I. Gondal, and V. Balasubramanian, "A survey on the adoption of blockchain in iot: Challenges and solutions," *Blockchain: Research and Applications*, vol. 2, no. 2, p. 100006, 2021.

[8] J. Lu and K. Whitehouse, *Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks.* IEEE, 2009.

[9] M. Zimmerling, L. Mottola, and S. Santini, "Synchronous transmissions in low-power wireless: A survey of communication protocols and network services," *ACM Comput. Surv.*, vol. 53, Dec. 2020.

[10] S. Tonyali, K. Akkaya, N. Saputro, A. S. Uluagac, and M. Nojoumian, "Privacy-preserving protocols for secure and reliable data aggregation in iot-enabled smart metering systems," *Future Generation Computer Systems*, vol. 78, pp. 547–557, 2018.

[11] A. Ara, M. Al-Rodhaan, Y. Tian, and A. Al-Dhelaan, "A secure privacy-preserving data aggregation scheme based on bilinear elgamal cryptosystem for remote health monitoring systems," *IEEE Access*, vol. 5, pp. 12601–12617, 2017.

[12] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[13] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pp. 162–167, IEEE, 1986.

[14] B. Zhang, G. Liu, and B. Hu, "The coordination of nodes in the internet of things," in *2010 International conference on information, networking and automation (ICINA)*, vol. 2, pp. V2–299, IEEE, 2010.

[15] M. Conard and A. Ebnenasir, "A practical self-stabilizing leader election for networks of resource-constrained iot devices," in *2021 17th European Dependable Computing Conference (EDCC)*, pp. 127–134, IEEE, 2021.

[16] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. Alexander, "Rpl: Ipv6 routing protocol for low-power and lossy networks," tech. rep., 2012.

[17] S. Saha, O. Landsiedel, and M. C. Chan, "Efficient many-to-many data sharing using synchronous transmission and tdma," in *DCOSS, 2017.*

[18] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, 1998.

[19] L. Lamport, "Paxos made simple," *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, December 2001.

[20] L. Lamport, "Fast paxos," *Distributed Computing*, 2006.

[21] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *USENIX, 2014*, 2014.

[22] J. C. Corbett, J. Dean, and E. et al., "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, 2013.

[23] E. Androulaki, A. Barger, and B. et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *EuroSys, 2018.*

[24] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *2017 4th international conference on advanced computing and communication systems (ICACCS)*, pp. 1–5, IEEE, 2017.

[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system."

[26] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," 2012.

[27] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI, 1999* (M. I. Seltzer and P. J. Leach, eds.), USENIX Association, 1999.

[28] A. Ahmad, M. Saad, and A. Mohaisen, "Secure and transparent audit logs with blockaudit," *Journal of network and computer applications*, vol. 145, p. 102406, 2019.

[29] L. Gerrits, C. N. Samuel, R. Kromes, F. Verdier, S. Glock, and P. Guitton-Ouhamou, "Experimental scalability study of consortium blockchains with bft consensus for iot automotive use case," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 492–498, 2021.

[30] S. Latif, Z. Idrees, Z. e Huma, and J. Ahmad, "Blockchain technology for the industrial internet of things: A comprehensive survey on security challenges, architectures, applications, and future research directions," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 11, p. e4337, 2021.

[31] S. R. Maskey, S. Badsha, S. Sengupta, and I. Khalil, "Alicia: Applied intelligence in blockchain based vanet: Accident validation as a case study," *Information Processing & Management*, vol. 58, no. 3, p. 102508, 2021.

[32] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 1980.

[33] P. Chen, D. Han, T.-H. Weng, K.-C. Li, and A. Castiglione, "A novel byzantine fault tolerance consensus for green iot with intelligence based on reinforcement," *Journal of Information Security and Applications*, vol. 59, p. 102821, 2021.

[34] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer pbft consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2020.

[35] H. Qushtom, J. Mišić, X. Chang, and V. B. Mišić, "A scalable two-tier pbft consensus for blockchain-based iot data recording," in *ICC 2021-IEEE International Conference on Communications*, pp. 1–6, IEEE, 2021.

[36] L. Zhang and Q. Li, "Research on consensus efficiency based on practical byzantine fault tolerance," in *2018 10th International Conference on Modelling, Identification and Control (ICMIC)*, pp. 1–6, IEEE, 2018.

[37] X. Xu, G. Sun, and H. Yu, "An efficient blockchain pbft consensus protocol in energy constrained iot applications," in *2021 International Conference on UK-China Emerging Technologies (UCET)*, pp. 152–157, IEEE, 2021.

[38] S. Abbasian Dehkordi, K. Farajzadeh, J. Rezazadeh, R. Farahbakhsh, K. Sandrasegaran, and M. Abbasian Dehkordi, "A survey on data aggregation techniques in iot sensor networks," *Wireless Networks*, vol. 26, no. 2, pp. 1243–1263, 2020.

[39] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the internet of things: A systematic review of the literature and recommendations for future research," *Journal of Network and Computer Applications*, vol. 97, pp. 23–34, 2017.

[40] S. Ozdemir and Y. Xiao, "Secure data aggregation in wireless sensor networks: A comprehensive overview," *Computer Networks*, vol. 53, no. 12, pp. 2022–2037, 2009.

[41] R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot," *IEEE access*, vol. 5, pp. 3302–3312, 2017.

[42] J. Zhang, Y. Zhao, J. Wu, and B. Chen, "Lvpda: A lightweight and verifiable privacy-preserving data aggregation scheme for edge-enabled iot," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4016–4027, 2020.

[43] J. Jose, M. Princy, and J. Jose, "Peppda: Power efficient privacy preserving data aggregation for wireless sensor networks," in *2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN)*, pp. 330–336, 2013.

[44] Y. Liu, W. Guo, C.-I. Fan, L. Chang, and C. Cheng, "A practical privacy-preserving data aggregation (3pda) scheme for smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1767–1774, 2019.

[45] C.-I. Fan, S.-Y. Huang, and Y.-L. Lai, "Privacy-enhanced data aggregation scheme against internal attackers in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 666–675, 2014.

[46] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation," in *2013 Proceedings IEEE INFOCOM*, pp. 2634–2642, 2013.

[47] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "Pda: Privacy-preserving data aggregation in wireless sensor networks," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 2045–2053, 2007.

[48] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pp. 160–164, 1982.

[49] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, p. 612–613, nov 1979.

[50] W. Jia, H. Zhu, Z. Cao, X. Dong, and C. Xiao, "Human-factor-aware privacy-preserving aggregation in smart grid," *IEEE Systems Journal*, vol. 8, no. 2, pp. 598–607, 2014.

[51] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology - EUROCRYPT 2006*, (Berlin, Heidelberg), pp. 486–503, Springer Berlin Heidelberg, 2006.

[52] J. He, L. Cai, P. Cheng, J. Pan, and L. Shi, "Distributed privacy-preserving data aggregation against dishonest nodes in network systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1462–1470, 2019.

[53] D. He, N. Kumar, S. Zeadally, A. Vinel, and L. T. Yang, "Efficient and privacy-preserving data aggregation scheme for smart grid against internal adversaries," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2411–2419, 2017.

[54] T. Dimitriou and A. Michalas, "Multi-party trust computation in decentralized environments," in *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, 2012.

[55] L. Chen, R. Lu, and Z. Cao, "Pdaft: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications," *Peer-to-Peer networking and applications*, vol. 8, no. 6, pp. 1122–1132, 2015.

[56] V. A. Memos, K. E. Psannis, Y. Ishibashi, B.-G. Kim, and B. Gupta, "An efficient algorithm for media-based surveillance system (eamsus) in iot smart city framework," *Future Generation Computer Systems*.

[57] A. P. Plageras, K. E. Psannis, C. Stergiou, H. Wang, and B. Gupta, "Efficient iot-based sensor big data collection–processing and analysis in smart buildings," *Future Generation Computer Systems.*

[58] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE Transactions on Industrial Electronics.*

[59] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, 1982.

[60] "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, 2018.

[61] M. Van Steen and A. Tanenbaum, "Distributed systems principles and paradigms," *Network*, vol. 2, p. 28, 2002.

[62] T. Wu, F. Wu, J.-M. Redouté, and M. R. Yuce, "An autonomous wireless body area network implementation towards iot connected healthcare applications," *IEEE Access*, vol. 5, pp. 11413–11422, 2017.

[63] S. Tyagi, A. Agarwal, and P. Maheshwari, "A conceptual framework for iot-based healthcare system using cloud computing," in *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pp. 503–507, 2016.

[64] T. M. Bojan, U. R. Kumar, and V. M. Bojan, "An internet of things based intelligent transportation system," in *2014 IEEE International Conference on Vehicular Electronics and Safety*, pp. 174–179, 2014.

[65] H.-T. Wu and G.-J. Horng, "Establishing an intelligent transportation system with a network security mechanism in an internet of vehicle environment," *IEEE Access*, vol. 5, pp. 19239–19247, 2017.

[66] Y. Saleem, N. Crespi, M. H. Rehmani, and R. Copeland, "Internet of things-aided smart grid: Technologies, architectures, applications, prototypes, and future research directions," *IEEE Access*, vol. 7, pp. 62962–63003, 2019.

[67] H. Mohammed, S. Tonyali, K. Rabieh, M. Mahmoud, and K. Akkaya, "Efficient privacy-preserving data collection scheme for smart grid ami networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[68] S. Desai, R. Alhadad, N. Chilamkurti, and A. Mahmood, "A survey of privacy preserving schemes in ioe enabled smart grid advanced metering infrastructure," *Cluster Computing*, vol. 22, no. 1, pp. 43–69, 2019.

[69] C. Gentry, *A fully homomorphic encryption scheme.* Stanford university, 2009.

[70] S. Ji and Z. Cai, "Distributed data collection and its capacity in asynchronous wireless sensor networks," in *2012 Proceedings IEEE INFOCOM*, pp. 2113–2121, 2012.

[71] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *IPSN, 2011.*

[72] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.