Distributed Secure Image Regeneration in CyberPhysical Systems

Thesis to be submitted in partial fulfillment of the requirements for the degree

of

Dual-degree (B.Tech + M.Tech) in Electrical Engineering with specialization in Signal Processing and Instrumentation

by

Shailesh Mishra
17EE35014

Under the guidance of

Prof. Sanand Dilip Amita Athalye

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Electrical Engineering
Indian Institute of Technology,
Kharagpur
India - 721302

CERTIFICATE

This is to certify that we have examined the thesis entitled **Distributed Secure Image Regeneration in CyberPhysical Systems**, submitted by **Shailesh Mishra** (Roll Number: *17EE35014*) a postgraduate student of **Department of Department of Electrical Engineering** in partial fulfillment for the award of degree of Dual-degree (B.Tech + M.Tech) in Electrical Engineering with specialization in Signal Processing and Instrumentation. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the Post Graduate Degree for which it has been submitted. The thesis has fulfilled all the requirements as per the regulations of the Institute and has reached the standard needed for submission.

Supervisor

Department of Department of Electrical Engineering
Indian Institute of Technology,
Kharagpur

Place: Kharagpur
Date: December 17, 2022

ACKNOWLEDGEMENTS

I would like to thank Prof. Sanand for providing me with this amazing opportunity to collaborate with him where we have been able to develop a distributed system for image reconstruction.
I would also like to thank my parents for their constant support, without which it would have been impossible for me to complete this work.

Shailesh Mishra
IIT Kharagpur
Date: December 17, 2022

ABSTRACT

Cyberphysical Systems (CPSs) have seen wide integration in numerous fields of science and technology because of their ability to acquire and process a large number of data from various sources simultaneously. Among the various types of data processed by CPSs, image is one of the most prevalent. The standard model of CPSs analyzing images consists of a raft of sensors obtaining image data and a *central entity* that processes the images sent to it by the sensors. Deployment of numerous sensors has made the cumulative size of the data acquired by sensors, which in turn makes it extremely difficult for a central entity to manage. In addition, images acquired by CPSs are privacy-sensitive, which if tampered with, can cause various issues. This phenomena is very much possible in a centralized architecture where data is sent to a single entity. Thus, a distributed secure solution is needed to mitigate these issues.

Therefore, in this research work, we propose two distributed frameworks, DIRAS and DSITR, for transmitting and regenerating images in a secure and privacy-preserving manner. Being a distributed system, DIRAS and DSITR solve the issue of scalability of CPSs. A novel leader-based consensus algorithm has been deployed in both DIRAS and DSITR to reach consensus among the nodes. DIRAS deploys RPCA to remove the noise from the transferred image. DIRAS integrates matrix completion to regenerate images even during the case of packet drops. Since DIRAS does not deploy public-key infrastructure (PKI), the bandwidth consumed in this framework is relatively less. However, the absence of encryption makes the packets vulnerable to being tampered with. DSITR attempts to overcome these issues. DSITR deploys PKI for secure transmission of images between images which makes it near impossible to pollute an image. DSITR uses matrix completion extensively so that the monitor nodes regenerate the image from lesser amount of data. This reduces the bandwidth consumed considerably. In addition, the architecture of DSITR makes it more easily deployable than DIRAS. However, the usage of PKI makes DSITR dependent on certificate authorities for providing valid public keys to the nodes.

Hence, DIRAS and DSITR come with their own benefits and issues. Therefore, in this report, we describe the architecture of both the models, and present the implicative results which give a better picture of where these models can be deployed.

Keywords: *Cyberphysical System (CPS), Image, Distributed Systems, Matrix Completion, Image Regeneration, Security, Privacy, Robust Principal Component Analysis (RPCA), Encryption*

Contents

List of Figures

Chapter 1

Motivation

Cyberphysical Systems (CPSs) combine physical processes, computational processes and communications networks aiming to enhance the overall functioning of systems [8]. CPSs have found great usage in various domains such as power grids, healthcare, supply chain and various other fields. Another sister technology of CPS is the *Internet of Things* (IoT), which connects all the devices over the internet so that the devices can interact with each other to reach a common goal [7]. Around 20 billion IoT devices will be connected in 2022, with that number expected to rise to 41 billion by 2025 [30, 21]. Due to the abundance of data acquired by billions of sensors and the cumulative processing power of the billions of devices, CPS and IoT have revolutionized the way various systems are handled. Along with the IoT, CPSs have played an integral role in improving the efficacy of the existing processes.

CPSs and IoT acquire various types of data, process them and take actions according to the implications obtained from the processing. CPSs acquire and process data such as image, audio, temperature, humidity, voltage and many more. Among all the above-mentioned types of data processed by CPSs, images are one of the most important due to the abundance of information carried by them. With the recent advancements in image capturing ability of devices and computer vision [22, 4, 36], images have become an integral form of data in our lives. Recently, analysis of image has become a key component for proper functioning of various sectors of society such as: agriculture [37], medical systems [16], remote sensing [13], robotics [25] and many other fields. Therefore, image data holds great value in various fields.

However, the generation of terabytes of image data by CPS and IoT has made the management of data extremely difficult. The sharing and storage of such a huge amount of data has become a prevalent issue in CPSs, which has been exploited by adversaries extensively. Firstly, in the conventional architecture, CPSs that process images, deploy a central processing unit. This architecture is not scalable, i.e., the processing unit won't be able to handle the increase in number of sensors. With the steep rise in application of CPS in various domains, this scenario is imminent. Secondly, if the processing unit fails or is compromised, then the whole framework would fall. Therefore, there's the issue of *single point of failure*. Next, the images, acquired by the sensors from their surroundings, contain sensitive information. Consider sensors deployed in a hospital or a military base. If compromised, such images can lead to fatal consequences. Lastly, the communication channel is also vulnerable, i.e., an adversary can sniff packets from the packets that are being shared between the sensor and the computing unit which is a classic example of the man in the middle (MITM) attack [33]. If this attack is successful, then the attacker can easily tamper with the image. This can lead to false data injection attacks (FDIA) [3]. In another scenario, an adversary can execute a denial of service (DoS) [27] attack, where the connection to a specific node is completely blocked by the adversary. Here, the attacker sends large volumes of data to the victim node, thus preventing any other communication for the victim node. Therefore, there are numerous issues that exist in the frameworks that involve communication of images from sensors to computing units.

To overcome the above-mentioned issues, we propose *DIRAS*, a distributed and robust solution for reconstructing images in CPSs. DIRAS reconstructs images in a distributed manner which can be further used for analysis. DIRAS consists of multiple nodes for computation (which have been called as monitor nodes) that mitigates the issue of centralization, and thus, the issue of single point of failure. In the case of DIRAS, the sensors split the images into chunks and distribute it to the sensors. Splitting of data improves the privacy of data. DIRAS deploys an efficient, random consensus algorithm along with robust principal component analysis (RPCA) [11] for image reconstruction, even in the case of false data injection attack. DIRAS incorporates numerous design schemas to mitigate other various kinds of attacks.

Moreover, DIRAS does not deploy public-key infrastructure (PKI) which makes it independent of any certificate authorities. This prevents any dependency on third party. However, by not using encryption and digital signatures, DIRAS becomes more vulnerable to FDIA. Due to this reason, the final image generated in DIRAS has higher chances of being polluted. In addition, in the case of DIRAS, a sensor node is connected to all the monitor nodes. This feature makes the deployment of DIRAS difficult because in a CPS, a sensor is normally a device with quite less computation power. Therefore, the design of DIRAS needs some improvements to make image regeneration more feasible and efficient.

To mitigate the issues in image regeneration in DIRAS, we propose DSITR which incorporates PKI for secure transmission of images. Just like DIRAS, DSITR is distributed and scalable. To improve the performance of the framework, DSITR incorporates network sharding. DSITR deploys the same random consensus algorithm that has been used in DIRAS. Moreover, in case of DSITR, a sensor node is connected to only a small set of monitor nodes which makes the design more easily deployable. However, by including PKI, the amount of data that needs to be sent in each packet increases (each packet would consist of the actual message, the digital signature, and the hash). This increases the total bandwidth consumed in the whole framework. Therefore, DIRAS and DSITR provide different merits and demerits.

Overall, DIRAS and DSITR provide novel distributed secure solutions for regenerating images. The major contributions of this work are:

1. Two novel distributed frameworks consisting of sensor nodes and monitor nodes to reconstruct images. One deploys RPCA and matrix completion [24] to improve the robustness of image reconstruction. The other deploys PKI and matrix completion for image regeneration that makes the overall process more secure.

2. A light-weight, randomized leader selection algorithm for reaching a consensus among the monitor nodes.

3. An efficient data splitting mechanism by sensors to enhance data privacy and reducing the overheads.

4. Preliminary results that prove the benefits of using the frameworks

Rest of the report has been organized as follows: Section 2 outlines the relevant research that has been conducted in this field. Section 3 provides necessary background knowledge for this research. Section 4 elucidates the design of the system, various algorithms used and the mechanisms deployed. Section 6 discusses the implementation of the developed system. Section 7 describes the results from the implementation and compares the proposed model with the existing models. Section 8 elaborates on the implications obtained from the evaluations and the performance of the framework against various attacks. Section 9 concludes the report.

Chapter 2

Literature Review

The works that have been presented in this report span four different research fields which are: security in CPSs, distributed optimizations, image regeneration and secure image transmission. Here, we study the existing work that has been done in these four fields.

2.1 Security in CPSs

Numerous research works have addressed the security issues that exist in CPSs. In [31], the authors have presented control-theoretic approaches to cyberphysical security. First of all, the models of CPS, monitors and attacks have been described. *Detectability* and *identifiability* of attacks for these models have been defined. Then, they discuss the detection and identification limitations from system and graph-theoretic perspectives. They have also discussed monitor design problem and provided a case study on coordinated attacks against power networks. They have used power systems networks and water networks as example for study in the paper. In [34], the authors have proposed two algorithms for state reconstruction from sensor measurements which are corrupted with sparse attack. In [20], malicious state attacks in a remote state estimation has been considered. Here, a smart sensor node transmits data to a remote estimator equipped with a false data detector. The authors have presented an optimal linear deception attack on sensor data where the adversary can successfully inject data and remain undetected by the false data detector. In [17], the authors have proposed a technique for exact reconstruction of the discrete state of a switching system, when only the continuous output is accessible and the discrete output is not available. They have also investigated the scenario where the continuous input and output signal is adulterated by malicious attacks. All these works have presented formidable solutions for attacks.

2.2 Distributed Optimizations

In [42], the authors elaoborate on various distributed optimization algorithms. In distributed optimization of multi-agent systems, the agents collaborate with each other to minimize a global function which is the sum of local objective functions. In [6], the authors have presented a decentralized technique to solve the equation $Ax = B$ in the case of network of agents. In [29], the authors have provided an overview of distributed methods for optimization in networked systems. A distributed architecture can help to overcome the issues of scalability, single point of failure and privacy breaches that exist in the case of a centralised model.

In addition, in the field of distributed computing, there are multiple algorithms that have been proposed to reach consensus. First of all, there is Paxos [26] which presents an algorithm where a proposer proposes a value and a common value is accepted by an acceptor. Paxos is used for message sharing where the nodes want to reach a common point. A recent technology that has also received great attention is *blockchain*. Bitcoin [28] is the most famous cryptocurrency and the most widespread application of blockchain. Bitcoin deploys the Proof-of-Work (PoW) to reach a consensus. Here, the nodes solve a computationally expensive problem to become the miner. The miner is the node in the blockchain that adds a new block to Bitcoin. Although the application of Paxos and PoW is different, there is one similarity between these two algorithms. They work on the basis of a leader selection who executes the main task of maintaining the framework. This idea has also been used in DIRAS and DSITR to reach a consensus.

2.3 Image reconstruction

Image reconstruction has received great attention recently. It has been used extensively in the field of tomography [40]. Image reconstruction methods are being used for reconstructing 3D images from various projections of the image. The work done in [39] discusses proposes machine learning algorithms for image reconstruction in tomography. In [9], the authors have presented the application of deep learning algorithms for fluorescence image reconstruction. The authors in [38] have proposed a deep learning algorithm for tomographic image reconstruction.

2.4 Secure Image Transmission

Secure image transmission has been studied extensively for long. In [1], the authors have presented a technique where they have used block-based elliptic curve public key encryption as the first stage of encryption and then, they have used XOR of the first stage as the second stage of encryption. The results presented in the work depict the efficiency of the proposed method. In [2], the authors have proposed a chaos-based fractal encryption scheme for secure image transmission. On the other hand, the authors of [5] have proposed a blockchain-based method for secure image transmission.

All the methods proposed for image reconstruction are executed by powerful machines (personal computers) at a particular location. The same goes for the solutions described in the secure image transmission. Such a framework cannot be used in CPS because the devices are computationally less powerful. DIRAS, the solution that we have proposed here, considers first of all the security in CPS and tries to mitigate the effects of attacks. Next, DIRAS is a distributed framework and uses an algorithm which is inspired from the work in [28]. Moreover, the distributed optimization using Least Squares Solution, as presented in [29], provides a completely distributed solution but the solution provided by this is not accurate enough. The matrix generated after this optimization has each element equal to the element of the actual matrix divided by the total number of elements (the element will be normalized). Thus, although the system will reach a consensus, the deviation will still be very high. Due to this reason, we have deployed a leader-based algorithm that helps in reaching consensus quickly and regenerating images fast. DIRAS uses RPCA and matrix completion as its core component for reconstructing mmatrices. In the next section, we describe these two technologies.

Chapter 3

Background

In this section, we describe the technologies which form an integral part of DIRAS. Robust Principal Component Analysis has been used for extracting a low-rank matrix from the matrix that is obtained after collecting the image from other nodes. On the other hand, matrix completion has been used for solving the issue that arises when an adversary simply drops packets and does not allow the packet to reach the destination node. These two algorithms have been elaborated in this section.

3.1 Robust Principal Component Analysis

Robust Principal Component Analysis (RPCA) is the algorithm for obtaining the low rank matrix and the sparse component of the matrix when the matrix is corrupted [11, 41]. It is the modification of the long established Principal Component Analysis (PCA) to make it work even in the case of gross corruption of the data. First of all, we have defined what is PCA and then, elaborate about RPCA.

Suppose we have a data matrix which is the sum of a low rank matrix and a sparse matrix. PCA is the method via which we can obtain the low rank matrix and the sparse component from the data matrix. This is possible only under certain assumptions and can be achieved by solving a convex optimization problem called *Principal Component Pursuit*.

PCA is widely used for data analysis and dimensionality reduction problems. However, its performance reduces greatly when the data is *grossly* corrupted[1]. Therefore, RPCA has been developed for making the process of PCA robust.

[1]Gross errors are the ones which are generally large with respect to the data.

Many RPCA mechanisms have been proposed in literature using multivariate trimming [19], alternating minimization [23], and random sampling techniques [18]. However, these algorithms do not provide a polynomial-time complexity and hence, can be inefficient in case of large matrices. The RPCA described in [11] is the improved version of the other algorithms and yields the low-rank matrix from a highly corrupted matrix.

Hence, in the case of DIRAS, we have used the RPCA described in [11] and we describe the performance of the algorithm in Section 7.

3.2 Matrix Completion

Matrix Completion will be used in DIRAS when RPCA alone will not be able to provide efficient results. This would occur when multiple packets will be unavailable for the node that will reconstruct the image. Therefore, matrix completion is a very important part of DIRAS to improve its security.

In [12], the authors have provided a method for completing a matrix with minimal entries. They have proposed that nuclear-norm minimization (NNM) subject to data constraints, which is a convex optimization problem, needs to be solved to complete a data matrix. Their results prove that matrix completion provides satisfactory output in case of small noise in the data. The work done by the author in [32] provides a method to complete matrix by minimizing the nuclear norm of the hidden matrix. In [24], the authors have presented a method called OptSpace to improve on the work presented in [12] and [32]. We have also deployed the nuclear-norm minimization technique for matrix completion. In [35], the authors have proposed alternating gradient descent (ASD), which is an iterative method for regenerating matrix.

Next, we explain how DIRAS and DSITR have been designed and how these two technologies mentioned here help in robust image reconstruction.

Chapter 4

DIRAS: Distributed Image Reconstructor for Adversarial Scenario

In this section, we describe the architecture of DIRAS and the algorithms deployed in it. DIRAS is a distributed framework for regenerating images, even in the case of attacks. DIRAS deploys defense mechanisms against false data injection attack, DoS attack and the scenario where the adversary drops a packet. DIRAS also deploys image splitting for improving privacy of information. All the features have been elucidated of DIRAS have been elucidated in this section. First of all, the assumptions made while designing DIRAS have been described. Next, the components of DIRAS have been outlined. Then, the various mechanisms and algorithms used in DIRAS have been explained. Table 4.1 provides the various terminologies used in this section to describe the various components of DIRAS.

4.1 Assumptions

The essential assumptions made while designing DIRAS are as follows: (i) The processing nodes, that regenerate the image, are connected over a *peer-to-peer* (p2p) network. In other words, all the processing nodes are connected with each other; (ii) The p2p network is synchronous; (iii) All the processing nodes are trusted, i.e., a node that receives data from the monitor does not behave maliciously; (iv) The processing nodes have enough computational power to run algorithms and regenerate the image; and (v) Sensors have enough computational power to split matrices.

Table 4.1: Definition of the terminologies used

Terminology	**Definition**
S_i	$i-th$ sensor node
SN	Number of sensor nodes
I_i	Image generated by $i-th$ sensor node
$(x_i(I_i), y_i(I_i))$	Coordinates corresponding to the $i-th$ sensor node for the image I_i
M_j	$j-th$ monitor node
MN	Number of monitor nodes
(x_j, y_j)	Coordinates corresponding to the $j-th$ monitor node
Δ	Epoch time (in seconds)
δ	Time (in seconds) a leader waits before regenerating the image
$C(I_i)$	Set of chunks of the image I_i
$C_j(I_i)$	Chunk of the image I_i sent to the $j-th$ monitor node
I_i_ID	Identifier of the image generated by the sensor
M_j_ID	Identifier of the monitor node
$L(I_i)$	Leader for the image I_i
β_j	Count of the number of images regenerated by the $j-th$ monitor node

4.2 System Architecture

Here, we discuss the building blocks of DIRAS. Figure 4.1 depicts the high-level architecture of DIRAS.

4.2.1 Sensor Nodes

These are the part of CPSs that acquire the image data from the surroundings. After acquisition of images, the sensor nodes split the images into chunks and send it over to the monitor nodes. Splitting helps in two ways: (i) splitting an image leads to improved privacy because an adversary cannot acquire enough information from chunks of an image; (ii) splitting reduces the bandwidth consumption on a particular communication channel. Furthermore, the sensors can record images from any source

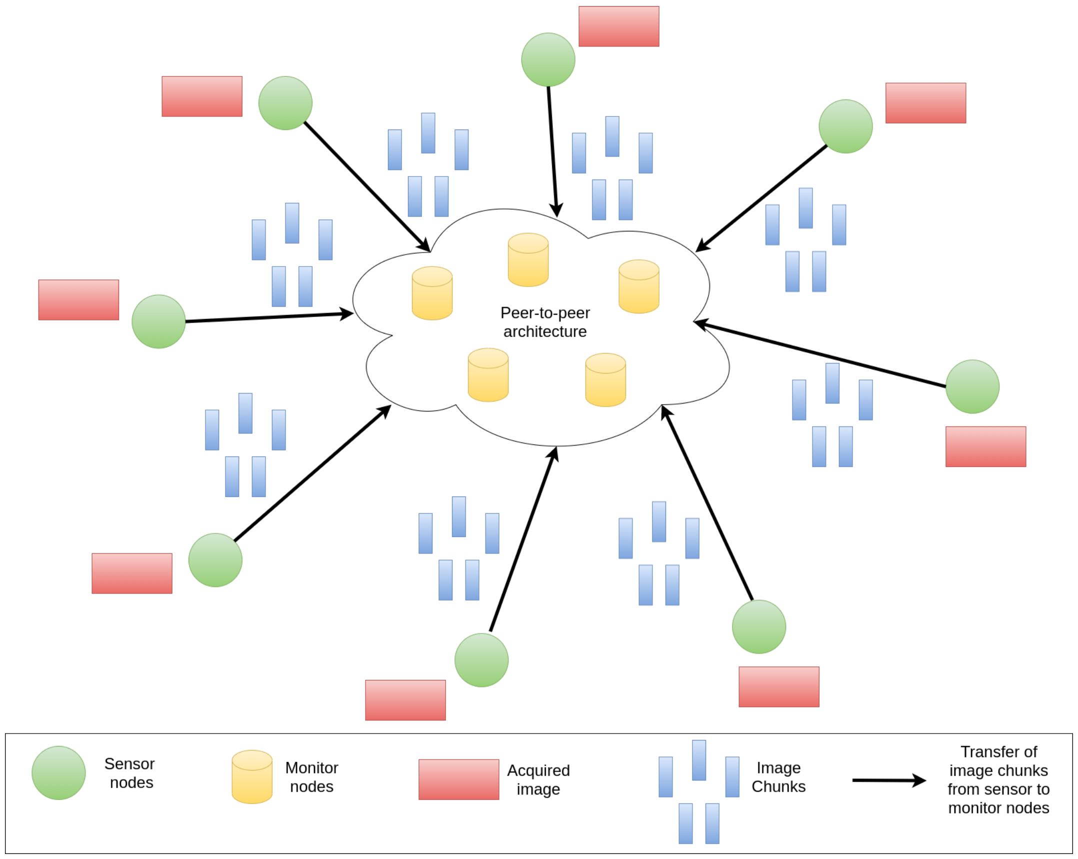

Peer-to-peer architecture
Sensor nodes
Monitor nodes
Acquired image
Image Chunks
Transfer of image chunks from sensor to monitor nodes

Figure 4.1: High level architecture of DIRAS

such as from a factory or a hospital, thus making DIRAS platform independent.

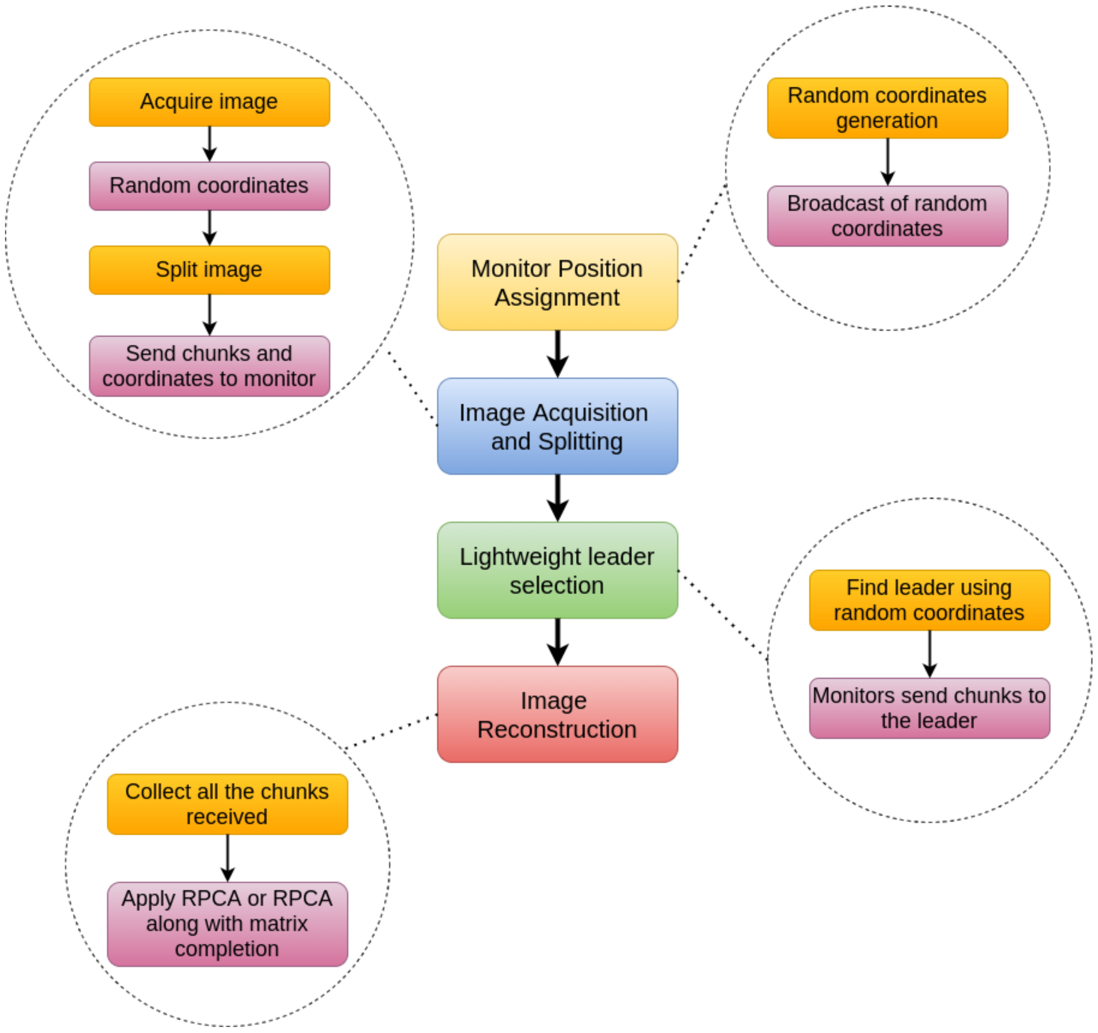
4.2.2 Monitor Nodes

These are the processing units of the CPSs. They are the nodes that receive chunks of acquired images from the sensors and regenerate the whole images. The images regenerated by the monitor nodes can be used further for other processing and analysis. These analyses are beyond the scope of this work.

4.2.3 Peer to peer(p2p) network

The monitor nodes are connected over a p2p network. This is to ensure that any monitor node can share a packet with any other monitor node. This is a reasonable assumption because considering the advent of the IoT, most of the processing devices around us are connected to each other. Thus, making such an assumption would not make a huge difference to the current architecture.

4.3 Functionality



Acquire image
Random coordinates
Split image
Send chunks and coordinates to monitor
Random coordinates generation
Broadcast of random coordinates
Monitor Position Assignment
Image Acquisition and Splitting
Lightweight leader selection
Image Reconstruction
Find leader using random coordinates
Monitors send chunks to the leader
Collect all the chunks received
Apply RPCA or RPCA along with matrix completion

Figure 4.2: Various steps involved in DIRAS

Here, the various design schemas and algorithms used in DIRAS for its functioning, and making it efficient and robust have been described. Figure 4.2 depicts the various functionalities involved in DIRAS. First of all, DIRAS is a distributed framework for regenerating image. Being a distributed system, the monitor nodes in DIRAS need to reach a consensus for regenerating images efficiently. We do this by using a lightweight consensus algorithm. For each image, a leader is selected randomly who is responsible for aggregating all the chunks of an image and then, regenerating it. This design ensures lower overhead and does not overburden any particular monitor node. The details of this algorithm have been discussed below.

4.3.1 Monitor Position Assignment

This is an important step in the functioning of DIRAS. In this step, the monitor nodes generate 2-D coordinates (x_j, y_j) with each coordinate being a (pseudo)random integer (from now on we refer a psuedorandom number as a random number). (x_j, y_j) is generated by the monitor nodes for every epoch time Δ. It should be noted that the sensor nodes do not know about the coordinates of the monitor nodes. Thus, changing of the coordinates (x_j, y_j) appears to be superfluous to the design. In spite of this, the monitor nodes keep changing the set of coordinates for each Δ. It is so as to improve the security of DIRAS against DoS attack (explained in the section 8). After generating the coordinates, the monitor nodes broadcast these packets to all other monitor nodes. The structure of the packet sent by a monitor node is $< M_j\_ID, (x_j, y_j) >$.

After receiving the packets from all the monitor nodes, the monitor nodes simply store the coordinates of all other nodes in its memory. The position assignment is an important step for the leader selection.

4.3.2 Image Acquisition and Splitting

This is the task carried out by the sensors of DIRAS. The sensors acquire images from their surroundings continuously. After acquiring an image I_i, a sensor generates a 2-D coordinates $(x_i(I_i), y_i(I_i))$ with each of the coordinates being a random number. The coordinates being a function of I depicts that a sensor node generates unique coordinates for every image acquired. The image acquired by the monitor will have the R, G and B components. Thus, the dimension of the acquired image matrix would be $m \times n \times 3$ (where m is the number of rows in the image and n is the number of columns in the image). Before splitting the image, the R, G and B components are stacked vertically, i.e., the order of the new matrix becomes - $3m \times n$. After stacking the image vertically, a sensor splits the image into chunks row-wise. We have carried out the splitting row-wise but it can be done column-wise (if done column wise, then the stacked matrix would be $m \times 3n$) or by any other way that can be decided by the network operator. After splitting the image, the sensor form packets whose structure is: $< I_i\_ID, M_j\_ID, C_j(I_i), (x_j, y_j) >$.

All the components of the packet have been explained in Table 4.1. The packet depicted here is for a single chunk of image sent by the $i - th$ sensor node to the $j - th$ monitor. After receiving this packet, the monitor nodes select the leader for the image.

4.3.3 Lightweight Random Leader Selection Algorithm

The most important component in a distributed system is to reach a *consensus*, i.e., all the nodes in the p2p network should be in the same state. This is a challenge in any distributed system. In the case of DIRAS, we achieve this by selecting a leader for every image. The leader is selected by using a simple method which has been described below.

1. Each monitor node finds the distance between the coordinates generated by every monitor node and the coordinate sent by the sensor node by using the formula: $D_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

2. Then, the monitor node evaluate the monitor node in the p2p network that has the minimum D_{ij} for the image.

3. The one with the minimum D_{ij} is the leader for the image.

4. Every other monitor node sends its $C_j(I_i)$ to the leader and the leader can then regenerate the image whose structure is $< I_i\_ID, M_j\_ID, C_j(I_i) >$.

The leader selection algorithm is not a new concept in distributed systems. It has been used extensively in blockchains [28]. In standard blockchains, the leader (which is called the miner) is chosen on the basis of a cryptographic puzzle. The one who solves the puzzle becomes the miner of a block and adds a block to the blockchain. However, the algorithm used in the case of [28] is resource intensive and requires a lot of time and power. Therefore, we have introduced a lightweight and random algorithm to select a leader for regenerating an image.

This algorithm appears to be a little centralized with respect to a particular image. But if we consider a long duration of time, this algorithm is decentralized because the leader is selected on the basis of four random coordinates. Thus, if considered honest behavior from all the nodes in the network, this algorithm is decentralized. Another choice would be to design a completely decentralized algorithm where each monitor node regenerates the image. In this case, after receiving the packet from the sensor node, each monitor node would broadcast the packet to all other nodes in the network. However, this mechanism would increase the overheads greatly making the system very inefficient. Thus, the algorithm in DIRAS, which is distributed with respect to time, provides an optimal distributed solution. We have compared the performance of DIRAS with the above-mentioned decentralized model in section 7. Next, we discuss the regeneration of the image by a monitor node after it receives all the packet.

4.3.4 Image Regeneration

After receiving the first packet corresponding to an image from a source, each node starts a timer. This timer is integrated so that a leader does not wait indefinitely for regenerating an image. Each leader waits for a time δ after receiving the first packet. After the time δ is over, the leader starts the regeneration process. The leader builds a matrix whose dimension is $3m \times n$. Therefore, the leader first breaks the matrix into 3 matrices of size $m \times n$. After this, it has two paths for regeneration: (i) *The leader receives all the packets*: In this case, the leader first of all combines all the chunks and then, applies RPCA to extract the low-rank matrix. The low-rank matrix is the image that will be used by other systems for analysis; (ii) *The leader receives some of the packets*: This means that the packet sent by either the monitor node or the sensor node gets dropped. In this case, the leader combines all the chunks and makes the rows equal to zero which it hasn't received. Then, it applies RPCA on the matrix that it gets after combining all the rows. The next step is to apply matrix completion algorithms on the matrix that has been obtained after applying RPCA. The matrix obtained after matrix completion is the regenerated image that will be used for further analysis.

This is how the whole image regeneration algorithm works in DIRAS. Next, we discuss load balancing, a mechanism to improve DIRAS.

4.3.5 Improvement in DIRAS: Load Balancing

DIRAS uses a very simple algorithm for selecting the leader which generates the final image. This algorithm uses random numbers which makes it non-deterministic and efficient. However, the randomness also distributes the tasks among the monitor nodes randomly. Therefore, there will be multiple scenarios where one node may have to regenerate more images than the others. This will induce latency in the functioning of the network because more traffic will be there at some M_j. To prevent this scenario, we have added the feature of load balancing to improve DIRAS. The load balancing algorithm is again a simple yet effective one (we prove is efficacy in the results). Next, we discuss the load balancing algorithm.

Recall that the leader was chosen based on the algorithm:

$$\begin{aligned} L(I_i) = j, &\quad \text{s.t.} \quad \min_j \quad D_{ij} \\ \text{where} \quad D_{ij} = \sqrt{(x_i(I_i) - x_j)^2 + (y_i(I_i) - y_j)^2}, &\quad \text{j} = 1, ..., \mathbf{MN} \end{aligned}$$

, where all the coordinates are random numbers. To mitigate the issue of uneven load distribution, we have introduced β_j, which is the total number of images regenerated by a monitor node. For load balancing, all the monitor nodes keep a track of β_j of all the monitor nodes in the network. After the integration of load balancing, the leader selection algorithm becomes,

$$\begin{aligned} L(I_i) = j, &\quad \text{s.t.} \quad \min_j \quad F_{ij} \\ F_{ij} &= D_{ij} + \beta_j \times \max_j \quad D_{ij} \\ \text{where} \quad D_{ij} = \sqrt{(x_i(I_i) - x_j)^2 + (y_i(I_i) - y_j)^2}, &\quad \text{j} = 1, ..., \mathbf{MN} \end{aligned}$$

The term $\beta_j \times \max_j \quad D_{ij}$ has been added for load balancing. This term ensures that the monitor nodes, which have regenerated more images already, don't have to regenerate the incoming images. Thus, this load balancing algorithm ensures that the computation burden is distributed evenly.

4.4 Limitations in DIRAS

Although DIRAS helps in regenerating images in a distributed manner and reduces noise in the regenerated image, there are some limitations which make its deployment less feasible:

1. DIRAS does not use PKI while transferring information from one node to another. Due to this reason, adversaries can acquire whole of the information carried by a packet. In addition, this makes the image data more vulnerable to tampering.

2. Moreover, all the sensor nodes in DIRAS need to be connected to all the monitor nodes. In general, sensor nodes are designed to only acquire data and send it to the few of their neighbours. Connecting them to hundreds or thousands of nodes would overburden sensor nodes. This could prove to be detrimental to the overall functioning of the network. Thus, this property makes it difficult to deploy DIRAS in the real world scenario.

3. Moreover, in DIRAS, there are some chances of the failure of a leader. This can lead to the loss of an image. The case of failure of the leader hasn't been dealt with in DIRAS.

4. The quality of image regenerated using DIRAS may not be of good quality. This is because the data is sent row-wise to the nodes. If a packet is dropped, then nodes lose row(s). In such a scenario, the matrix completion algorithm does not perform well (presented in the results).

To overcome the above-mentioned limitations in DIRAS, we have designed DSITR, a sharded network architecture for secure image transmission and regeneration. In the next chapter, we discuss the design and working of DSITR.

Chapter 5

DSITR: Distributed Secure Image Transmittor and Regenerator

In this section, we describe the architecture and working of DSITR. DSITR aims to overcome the shortcomings of DIRAS. DSITR incorporates PKI for encrypting packets transferred in the framework. This ensures that the image data, being transmitted in the network, is not tampered with. In addition, the monitor nodes network in DSITR is sharded, i.e., the monitor nodes network is fragmented into multiple parts. The functioning of each shard does not depend on the functioning of another shard. Each shard regenerates its own image. Due to this reason, there are multiple replicas of an image in different parts of the network. This is what makes DSITR fault tolerant. DIRAS has the requirement of a sensor being connected to all the monitor nodes in the network. As discussed earlier, the could prove to be a major issue while deploying DIRAS. To overcome this issue, DSITR only needs the sensor nodes to be connected to a small set of nodes. This makes the architecture conducive to real-world deployment. In addition to all these differences with respect to DIRAS, DSITR has many common features with DIRAS. For instance, the sensor nodes split the image into smaller chunks and then, send it to the various monitor nodes. Chunking of image ensures that communication channels are not overburdened. Just like DIRAS, DSITR deploys the same leader selection algorithm and load balancing algorithm. By including the formidable features of DIRAS and solving the issues of DIRAS, we have tried to make DSITR a robust and accurate image regenerator. First, we describe the network architecture of DSITR. Then, we describe the various functionalities of the model. Table 5.1 provides the various terminologies used in this section to describe the various components of DIRAS.

5.1 Assumptions

The essential assumptions made while designing DSITR are as follows: (i) The processing nodes, that regenerate images, are connected over a *peer-to-peer* (p2p) network within their shards. In other words, all the processing nodes are divided into subgroups and the nodes; (ii) The p2p network is synchronous, i.e., the processing nodes work according to a global clock; (iii) All the processing nodes are trusted, i.e., a node that receives data from the monitor does not behave maliciously; (iv) The processing nodes have enough computational power to run algorithms and reconstruct the image; (v) Sensors have enough computational power to split matrices; and (vi) There is a trusted certificate authority (CA) that provides public keys and private keys to the nodes in the network.

5.2 System Architecture

Here, we discuss the architecture of DSITR. Figure 5.1 depicts the high-level architecture of DSITR for a single sensor node. The diagram shows the architecture for a single sensor node. A sensor node is connected to all the shards in network, however, it is connected to only some nodes within each shard. Each shard has a leader which we call *Admin* node here. The admin node is the leader of each shard and coordinates the working of its respective shard. We describe each component of DSITR and also, discuss how its architecture is different from DIRAS.

5.2.1 Sensor Nodes

These nodes play the same role as the sensor nodes in the case of DIRAS. They acquire images, split them into chunks and send the chunks to the monitor nodes. The difference between the sensor nodes in DIRAS and DSITR lies in the way they are connected to the monitor nodes. In the case of DIRAS, a sensor node was connected to all the monitor nodes. On the other hand, in case of DSITR, a sensor node is connected only to a small set of monitor nodes. It does increase the amount of information carried by a packet, thus making a packet more vulnerable to information leakage. To mitigate this issue, we have incorporated data encryption, that makes it near impossible for an adversary to draw any relevant information from the

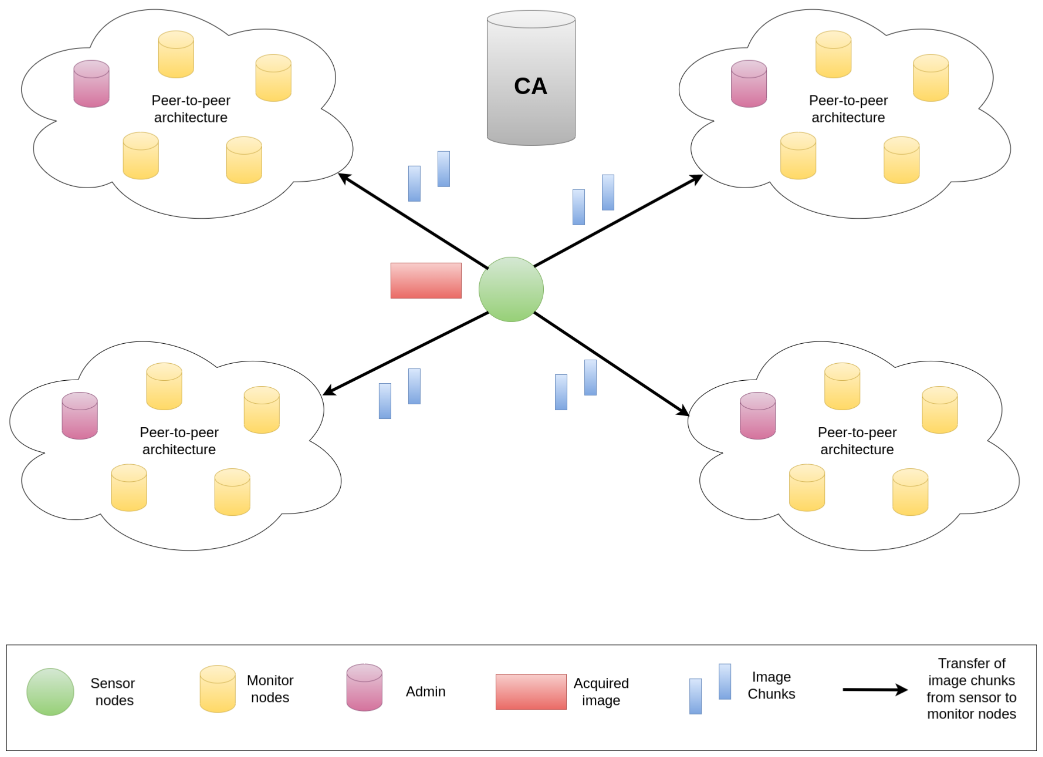

CA
Peer-to-peer architecture
Peer-to-peer architecture
Peer-to-peer architecture
Peer-to-peer architecture
Sensor nodes
Monitor nodes
Admin
Acquired image
Image Chunks
Transfer of image chunks from sensor to monitor nodes

Figure 5.1: High level architecture of DSITR

packets being transferred in the network. We discuss this more thoroughly in the functionalities subsection.

5.2.2 Monitor Nodes

These nodes are the same as the monitor nodes in the case of DIRAS. They receive chunks from the sensors and regenerate the whole image. The difference in this architecture is the way these nodes are connected with each other and the flow of information in their network. We discuss this thoroughly in this section.

5.2.3 Sharded peer to peer(p2p) network

The monitor nodes are connected over a sharded p2p network. The reason for incorporating sharding within the framework is to improve the fault tolerance of the framework. Each shard regenerates its own version of image. Thus, even if a large section of network fails, images are regenerated by the network. Within each shard, the monitor nodes are connected over a p2p network. This again makes the design of DSITR better because the monitor nodes need not be connected to a large set of nodes, which was the case in DIRAS.

5.2.4 Admin Node

Admin nodes are the leaders of each shard. Admin nodes are responsible for coordinating all the nodes within a shard and resolving any possible dispute. Admin nodes acquire the final image regenerated within their shard and share it with the admin of other shards to obtain the final image that is going to be used for further analysis. Admin nodes play a key role in the functioning of the overall framework. In addition, to this admin node, there are 2 other monitor nodes in each shard, which can play the role of an admin, in case of any failure.

5.2.5 Certificate Authority (CA)

Certificate Authority (CA) are bodies that provide public keys and private keys to the nodes in DSITR. Unless signed by the CA, a key is not considered valid. The CA plays a key role in the security aspect of DSITR because encryption forms the baseline for the functioning of DSITR.

Next, we discuss the working of DSITR.

5.3 Functionality

Here, the various design schemas and algorithms used in DSITR for its functioning have been described. Figure 4.2 depicts the various functionalities involved in DSITR. Just like DIRAS, DSITR is a distributed framework for transmitting image data securely and then, regenerating the whole image. DSITR uses the same lightweight consensus algorithm which was used in the case of DIRAS, where a leader is selected for each image. To reduce the bandwidth being consumed in the framework, we rely on matrix completion. DSITR works on the principle that the whole of the image data need not be transferred by the sensor to obtain relevant information. In other words, DSITR uses the concept of *approximate computing* to regenerate the image from some part of the whole image data. This does not provide the exact image but it does provide enough accuracy for functioning of the framework. In addition, DSITR deploys PKI for encryption and digital signature. This conceals the actual image data and also gets rid of the requirement of an algorithm for removing noise such as RPCA in the case DIRAS. The details of each step have been explained below. We have not explained the design features of DSITR that overlap with DIRAS, which otherwise would have made the report verbose.

5.3.1 Monitor Position Assignment

Just like DIRAS, this is an important step in the functioning of DSITR. The purpose of this step and its working is exactly same as that in the case of DIRAS. The only difference lies in the fact that the monitor nodes encrypt the coordinates, digitally sign their message, and broadcast the coordinates only to the nodes within their shard. After generating the coordinates, the monitor nodes encrypt the message using the receiver's public key and then, sign the message using their private key. On receiving a packet, a monitor node first verifies if the message is from the appropriate user. After verifying the signature, the monitor node decrypts the message to obtain the coordinates. The structure of the packet sent by a monitor node is $< M_j^k\_ID, en(x_j, y_j), Sign, Hash, TS >$. Just like DIRAS, the position assignment, in the case of DSITR, is an important step for the leader selection.

5.3.2 Image Acquisition and Splitting

This is the task carried out by the sensors of DSITR. Again, just like the monitor position assignment step, this step in DSITR is similar to the corresponding step in DIRAS. There are four subtle differences between DIRAS and DSITR in this step.

1. The first difference is that the DSITR is designed for black and white image as of now, while DIRAS has been designed for colorful images. This has been done so that we can obtain better performances in image regeneration.

2. The second is the use of encryption and digital signature while sending out packets to the monitor nodes.

3. The third difference is in the way the image is broken into small chunks. In the case of DIRAS, the image was split into chunks row-wise and the rows were sent to the monitor nodes. In this scenario, if any packets were dropped, then the quality of image regenerated was of poor quality. Therefore, in the case of DSITR, the pixels of an image are picked randomly, and then sent to the

monitor nodes. Since the elements are randomly picked, the index of the pixel is also sent along with the pixel. This increases the packet overhead slightly.

4. The fourth and the most major difference lies in the fact that a sensor node needs to send only some part of the image data to the monitor node network. This mitigates the effect of bandwidth consumption due to the inclusion coordinates of the pixels and thus, reduces the packet overhead to some extent.

After splitting the image, a sensor forms packets whose structure is:

$$< I_i\_ID, M_j^k\_ID, en(C_j(I_i)), en((x_j, y_j)), Sign, Hash, TS >$$

All the components of the packet have been explained in Table 5.1. The packet depicted here is for a single chunk of image sent by the $i - th$ sensor node to the $j - th$ monitor of the $k - th$ shard. After receiving this packet, the monitor nodes select the leader for the image.

5.3.3 Lightweight Random Leader Selection Algorithm

This algorithm is exactly same as that in the case of DIRAS. The only difference lies in the fact that the monitor nodes need to verify the signature and decrypt the message in the packet. Therefore, next we discuss the image regeneration mechanism used in DSITR which is quite different from the method used in DIRAS.

5.3.4 Image Regeneration

Just like DIRAS, after receiving a packet from a source, each node starts a timer so that a leader does not have to wait indefinitely for regenerating an image. It should be noted that the leader node would get the packets from only the monitor nodes of its own shard. After receiving the elements and their respective indices, the leader node begins the regenerating process. After time δ, the leader node applies matrix completion to obtain the pixels of the image that haven't been sent by the sensor to the shard. A leader applies nuclear norm minimization for obtaining the completed matrix of the image. However, when the image size is large, nuclear norm minimization takes a lot of time for optimization and in some cases, may not even regenerate image after tens of minutes. This is because nuclear norm minimization requires a node to solve a convex optimization problem which makes it time consuming for large matrices.

For such scenarios, the nodes will need iterative methods like alternating steepest descent (ASD) algorithm for matrix completion. ASD can handle larger images with larger number of pixels. Therefore, DSITR can shift to ASD in case of larger images. DSITR offers two choices to the sensor nodes - (i) flexible image size, where the monitor nodes need to switch between NNM and ASD based on the image size; (ii) hardwired image size where the nodes have been programmed to use either NNM or ASD all the time. This design choice has been left for the network designer. After image regeneration using matrix completion, the leader nodes send the image to the admin for further improvement in image quality.

5.3.5 Inter-shard communication

This is the final step and can be used based on the type of image being dealt with in the framework. This step is for improving the quality of final image that has to be used for further processing or analysis. After regenerating an image, the *leader* sends the regenerated image to the admin of the shard in which the leader is present. After receiving the image from the leader monitor node, the admin node finds the admin that is going to be the leader for the final image. The admin nodes use the same leader selection algorithm as they have the coordinate of the sensor node that has generated the image and their own set of coordinates which they generated during the monitor position assignment phase. After receiving all the RI_i^k, the leader among the admin nodes during the inter-shard communication regenerates the final image using the formula:

$$Final\, Image_i = \frac{\sum_{k=1}^{\mathbf{SHN}} RI_i^k}{\mathbf{SHN}}$$

By averaging over all the images being regenerated in the framework, the quality of final image gets better. We prove the benefit of this in the section 7. It should be also noted that this step involves consumption of extra bandwidth. Therefore, if a framework does not require high accuracy, then the system designer can remove this step. There is a chance that the leader monitor node changes the coordinates to cause DoS attack within this inter-shard communication. To counter this scenario, load balancing can be integrated with the inter-shard communication as well. This would ensure that none of the admin node is overburdened.

5.3.6 Load Balancing

DSITR deploys the same load balancing algorithm to improve its working. In addition to using load balancing for selecting leaders for regenerating images, DSITR uses load balancing for inter-shard communication as well. This ensures that the none of the node's channel is bloated. If a monitor node is found to be sending packets to the nodes which are not leaders, then this issue is mentioned to the admin node so that the point of adversarial activity can be detected.

In the next section, we elaborate on the implementation of DIRAS and DSITR so as to evaluate its performance.

Table 5.1: Definition of the terminologies used

Terminology	**Definition**
S_i	$i-th$ sensor node
SN	Number of sensor nodes
I_i	Image generated by $i-th$ sensor node
$(x_i(I_i), y_i(I_i))$	Coordinates corresponding to the $i-th$ sensor node for the image I_i
A_k	Admin node of the $k-th$ shard
SHN	Number of shards in the framework
M_j^k	$j-th$ monitor node of the $k-th$ shard
MN	Number of monitor nodes in a shard
ϕ_i^k	Number of monitor nodes to which $i-th$ sensor node is connected in the $k-th$ shard
(x_j^k, y_j^k)	Coordinates corresponding to the $j-th$ monitor node of the $k-th$ shard
Δ	Epoch time (in seconds)
δ	Time (in seconds) a leader waits before regenerating the image
$C(I_i)$	Set of chunks of the image I_i
$C_j^k(I_i)$	Chunk of the image I_i sent to the $j-th$ monitor node of the $k-th$ shard
I_i_ID	Identifier of the image generated by the sensor
S_i_ID	Identifier of the $i-th$ sensor. In the case of DSITR, identifier of a sensor node is its public key
$M_j^k\_ID$	Identifier of the $j-th$ monitor node of the $k-th$ shard. In the case of DSITR, identifier of a monitor node is its public key
P	Number of pixels sent to each shard
$L^k(I_i)$	Leader for the image I_i in the $k-th$ shard
RI_i^k	Regenerated version of I_i in the $k-th$ shard
β_j^k	Count of the number of images reconstructed by the $j-th$ monitor node
$en(m)$	Encryption of message m
$Sign$	Digital Signature of the corresponding message in the packet
$Hash$	Hash of the message in the packet
TS	Timestamp at which the packet is sent

Chapter 6

Implementation

In this section, we describe the implementation of both the systems. For each of them, we have separated the implementation of the system into four parts: (i) communication network; (ii) image regeneration framework; (iii) load balancing analysis; and (iv) privacy analysis.

6.1 DIRAS

First of all, we discuss the implementation of DIRAS.

6.1.1 The Communication Network

DIRAS involves sharing of data among monitor nodes and transfer of packets from sensor nodes to the monitor nodes. Therefore, there's a communication network established. The first implementation is done so as to study the scalability of the network, i.e., the performance of the network when the number of sensor nodes and monitor nodes increase. This implementation was done on ns-3 network simulator. First of all, the sensor nodes and monitor nodes were created in the simulator. The monitor-monitor and sensor-monitor connections are established. Then, monitor nodes transfer packets to each other for completing the step of *monitor position assignment*. Then, the sensor nodes generate images (each node generates five images), split the image into chunks and send the chunks to the respective monitor nodes. Lastly, monitor nodes find the leader and send the chunks to the leader. After developing the code for this, the simulator was run for two variations in the network.

6.1.1.1 Varying Sensor Nodes

In this case, we study the performance of DIRAS with the variation in number of sensors in the network. Here, we have also studied the performance of DIRAS with respect to the standard network architectures:

1. *Centralized model:* There is only one monitor node (server) and the sensor nodes send the whole image to a server. The centralized architecture is inspired from the conventional client-server architecture of network systems.

2. *Decentralized model:* Here, there are multiple monitor nodes and all the monitor nodes regenerate all the images. This model is inspired from the architecture used in distributed optimization [29]. However, there isn't any optimization running on any node. The decentralization implies that all the monitor nodes regenerate all the images. In this framework, a sensor node splits an image into chunks and send the chunks to the respective monitor nodes. The monitor nodes, then, broadcast their chunk to all other monitor nodes to ensure that all the monitor nodes regenerate the image.

The centralized and decentralized models are classical models which form the extreme cases of network systems. On one hand, the centralized model has less overheads but is prone to single point of failure. On the other hand, the decentralized framework has higher overheads but is fault-tolerant. DIRAS falls in the sweet spot of these two extreme cases and aims to provide low overheads and fault-tolerance.

The simulations for this part of implementation have been run with the following parameters: **MN** $= 1$ for centralized model, and **MN** $= 10$ for decentralized model and DIRAS; **SN** $= n$, where $n = \{10, 15, 25, 40, 50, 75, 100, 125, 150, 175, 200, 225, 250, 300\}$; $dimension(I_i) = 100 \times 100 \times 3$; and number of images generated by each sensor $= 5$.

6.1.1.2 Varying Monitor Nodes

Here, we study the effect of increasing the number of monitor nodes in DIRAS. The simulations for this part of implementation have been run with the following parameters: **MN** $= n$; where $n = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$; **SN** $= 100$; $dimension(I_i) = 100 \times 100 \times 3$; and number of images generated by each sensor $= 5$.

Figure 6.1: Image used for analysis in DIRAS

We have studied the packet overhead and network delay induced in the network because of the increase in monitor nodes. Recall that a sensor chunks an image according to the number of monitor nodes in the network. Thus, the packet overhead and the latency will increase with the increase in monitor which isn't appropriate for a CPS. However, it should be noted that an increase in chunks implies a decrease in the information carried by a chunk. Thus, the privacy of data is improved, i.e, if an adversary is able to intercept a chunk, then the adversary would be able to acquire less information when the number of monitor nodes is high. Thus, there is a trade-off between privacy and overheads. Due to this reason, we have also studied quality of privacy provided by DIRAS and quantified it.

It should be noted that there is no step in this implementation where the image of a sensor is polluted. Therefore, in this part of the implementation, we do not intend to study the image regeneration. Rather we wish to obtain the performance of the network in terms of a communication framework because the implementation described here exactly emulates the packet transfer mechanism as described in the system design. Therefore, this part of the implementation furnishes us with the study of the overheads and delays in the network due to the communication and processing except the regeneration part.

6.1.2 The Image Regeneration Framework

Since DIRAS aims at regenerating images, it is necessary to study the efficiency of DIRAS in regards to the ability to generate accurate images. In this implementation, we have studied this parameter of performance.

6.1.2.1 RPCA

In the first part of this implementation, we consider only false data injection attack, and there is no packet drop attack. This implementation has been done in *python-3.8.8* using the *NumPy* module. We have used the image depicted in figure 6.1 (which is a $200 \times 200 \times 3$ matrix), and then, introduce random sparse noise into it. The image with the induced noise is the image that is obtained after the leader obtains all the chunks from all other monitors. The false data injected is a matrix with random entries. The entries of the noise matrix had a maximum value set for its entries. We increased the maximum value of this error and evaluated the performance of the image regeneration algorithm. Thus, we have emulated the scenario where an adversary adds random noise to any chunk. After obtaining the noisy image, we apply RPCA to it to reduce the sparse noise. This gives us an idea of the degree of noise our regeneration method can handle. In this study, we have analyzed the performance of RPCA with respect to various image processing denoising algorithms which include: (i) mean blur; (ii) median blur; (iii) Gaussian blur; (iv) bilateral Filter [10]; and (v) Wiener filter[14]. We have studied the performance of the algorithms using the following formula: $Dist = E(M_{regenerated} - M_{actual})$, where $E(M)$ is the Euclidean norm of matrix M, $M_{regenerated}$ is the regenerated matrix after applying the denoising algorithm, and M_{actual} is the actual matrix (the matrix obtained from the figure 6.1). We calculated this for R, G and B components of the image ($Dist_R$, $Dist_G$ and $Dist_B$). Finally, the distance between the actual and regenerated image is:

$$Distance = \sqrt{\frac{Dist_R^2 + Dist_G^2 + Dist_B^2}{3}}$$

6.1.2.2 RPCA and Matrix Completion

Next, we study the performance of our algorithm in the case where an adversary drops an packet, i.e., the leader does not receive all the packets of a particular image, and noise is also added to the image.. In this case, the leader tries to regenerate the image by using matrix completion algorithm. This part of the implementation has also been done in *python-3.8.8* using the *NumPy* and *cvxpy* modules. To start with this implementation, we used the image in figure 6.1. Next, we add some random noise to the image. This random noise is same as the one described above. Then, we randomly remove some rows of the image. Note that by doing this, we have emulated the exact attack scenario - where an adversary is in the middle of a monitor and a sensor or in the middle of two monitors, and the attacker simply acts as a sink by not allowing the packet to pass to the receiving monitor. If the leader does not receive all the packets corresponding to an image in δ, the leader assumes that the packet has been dropped. First of all, the leader makes all the elements equal to zero for the rows it hasn't received. Then, it applies RPCA to remove the noise from the image. In the final step, the leader applies matrix completion algorithm to obtain the rows of the image that it hasn't received. This implementation helps us to study the efficiency of our image regeneration algorithm in case of the combined attack of addition of false data injection and dropping of packets.

6.1.3 Load Balancing Analysis

Here, we study the performance of the load balancing analysis proposed here. Here, we study the improvement in the leader selection algorithm. Firstly, the normal leader selection algorithm (without the load balancing) was implemented for 10 monitor nodes and 1000 images, i.e., we studied which node is decided as the leader based on the leader selection algorithm for each image. Then, we calculated how many times each monitor node acts as the leader. Next, we implement the leader selection algorithm using the load balancing and evaluate the number of times each node has to act as the leader. This implementation was also done in *python-3.8.8* and gives an understanding of the improvement provided by the load balancing algorithm.

6.1.4 Privacy Analysis

Here, we have analyzed the privacy provided by DIRAS. DIRAS deploys splitting of images into chunks, which ensures that an attacker cannot acquire a substantial amount of information if the attacker is able to read packets of a communication channel. Therefore, here, we study the variation of amount of information obtained by an adversary with the number of rows of the matrix intercepted. We vary the number of rows of the matrix obtained by the attacker and make the other rows of the matrix equal to zero. Then, we apply matrix completion to obtain the whole matrix from the intercepted rows. From here, we get an understanding of the amount of information revealed by chunks of image. This amount of information revealed is obtained using the Euclidean norm of the matrix obtained by the difference of the actual matrix and the matrix obtained after matrix completion.

Next, we will discuss the implementation of DSITR.

6.2 DSITR

6.2.1 The Communication Network

Just like DIRAS, DSITR has numerous interactions over the network. Due to this reason, we need to study the performance of DSITR with the change in number of network parameters. The network parameters include: (i) no. of sensor nodes; (ii) no. of nodes to which a sensor node is connected in a shard; (iii) no. of shards; (iv) size image; (v) no. pixels sent to each shard. Just like DIRAS, the networking part of DSITR has been built using ns-3 network simulator. We also integrated *CryptoPP* with ns-3 to implement encryption and digital signature. We have not compared the performance of DSITR with any other classical model because of its difference from any classical model. We varied various parameters to obtain the performance of DSITR. Consider the table 5.1 while considering the parameters for this subsection.

6.2.1.1 Varying Sensor Nodes

Here, we vary the number of sensor nodes and keep the rest of the parameters the same. The parameters chosen for this simulation are as follows: $\mathbf{SHN} = 4$, $\mathbf{MN} = 50$, $\phi^k = 4$, $dimension(I_i) = 200 \times 200$, $\mathbf{P} = 15000$ and $\mathbf{SN}$ is varied as:

$$[10, 15, 20, 25, 40, 50, 75, 100]$$

6.2.1.2 Varying Connections Per Node Per Shard

Here, we vary the number of monitor nodes to which a sensor node is connected to per shard and keep the rest of the parameters the same. The parameters chosen for this simulation are as follows: $\mathbf{SHN} = 4$, $\mathbf{MN} = 100$, $\mathbf{SN} = 50$, $dimension(I_i) = 200 \times 200$, $\mathbf{P} = 15000$ and ϕ^k is varied as:

$$[1, 2, 3, 4, 5, 6, 10, 15, 20, 25, 30, 40, 50]$$

6.2.1.3 Varying Shards

Here, we have varied the number of shards and the number of monitor nodes per shard in a way that the total number of monitor nodes in the framework remains constant at 200. The parameters chosen for this simulation are as follows: **SN** = 50, $\phi^k = 4$, $dimension(I_i) = 200 \times 200$, **P** = 15000 and **SHN** is varied as:

$$[1, 2, 4, 5, 8, 10, 20, 25]$$

and, **MN** = 200/**SHN**.

6.2.1.4 Varying Image Size

Here, we have varied the image size and the number of pixels sent to each shard, keeping rest of the components the same. The parameters chosen for this simulation are as follows: **SHN** = 4, **MN** = 50, **SN** = 50, $\phi^k = 4$, **P** = 15000, ϕ^k, $dimension(I_i) = n \times n$, where n is varied as:

$$[25, 50, 75, 100, 150, 200, 250, 300, 350, 400, 450, 500]$$

, and **P** $= 0.4 \times dimension(I_i)$.

6.2.1.5 Varying Number of Pixels Per Shard

Here, we have varied only the number of pixels per shard. The parameters chosen for this simulation are as follows: **SHN** = 4, **MN** = 50, **SN** = 50, $\phi^k = 4$, $dimension(I_i) = 200 \times 200$, **P** is varied from 2000 to 4000 with a jump of 2000.

6.2.2 The Image Regeneration Framework

Next, we study the image regeneration framework in DSITR. Here, there is almost no chance of image getting tampered with by an adversary. Thus, we only study how well the image regeneration works with lesser amount of data and how well it can handle the scenario of packet drop. We have also studied the improvement in the image quality after the inter-admin communication. Just like DIRAs, we have used the figure 6.2 for testing:

Figure 6.2: Image used for analysis in DSITR

6.2.3 Load Balancing Analysis

The load balancing analysis for DIRAS and DSITR are exactly the same. Therefore, we have implemented this part only once and depicted the result in one place.

6.2.4 Privacy Analysis

The privacy analysis for DIRAS and DSITR are very similar. Therefore, we have implemented this part only and depicted the result in one place.

These implementations help to study the scalability and ability of DIRAS and DSITR to regenerate images in case of adversarial activities. The implications obtained from the implementations have been discussed in the next section.

Chapter 7

Evaluation

In this section, we explain the results that we have obtained from the implementation.

7.1 Communication Network

Here, we describe the performance of network component of DIRAS and DSITR.

7.1.1 DIRAS

7.1.1.1 Varying sensor nodes

Here, we vary the number of sensor nodes keeping the number of monitor nodes constant (refer to 6.1.1.1).

Figure 7.1 depicts the packet overhead in centralized model, DIRAS and decentralized model. Packet overhead depicts the bandwidth consumed by the whole network. As the graph suggests, the packet overhead in the case of decentralized model is very high with respect to the other two, which makes it unsuitable for deployment. Centralized model has the least packet overhead because it doesn't involve any inter-monitor communication (as there is only one monitor node). DIRAS has a packet overhead which is not much higher than the the centralized model.

Figure 7.2 depicts the delay in centralized model, DIRAS and decentralized model. Delay depicts the delay in transmission of packets in the network. The x-axis depicts the number of sensor nodes deployed and the y-axis is the delay in the network. As the graph suggests, the delay in the case of decentralized model is much higher than the other two, which makes it unsuitable for deployment in CPS. Centralized model has the least network delay because it involves only transmission of packets from sensors to a single server. DIRAS has a network delay which is not much higher than the the centralized model.

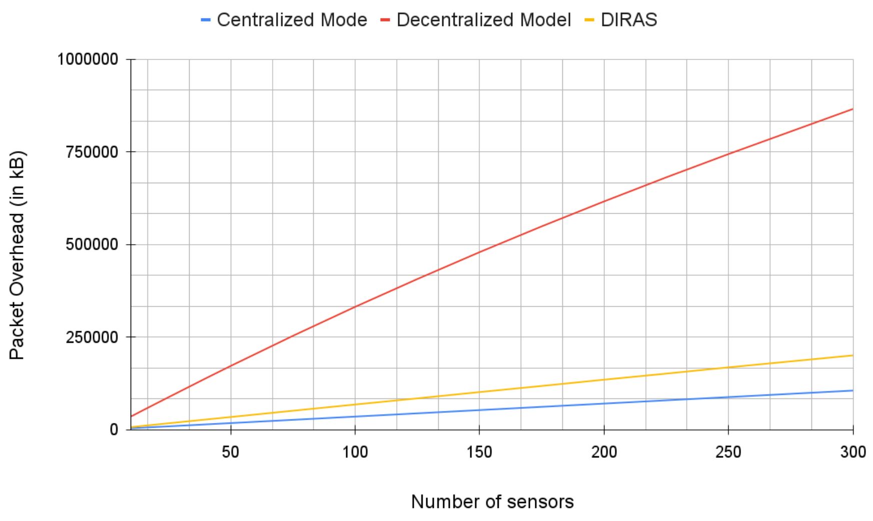


Figure 7.1: Variation in packet overhead with change in number of sensor nodes and comparison with centralized and decentralized models

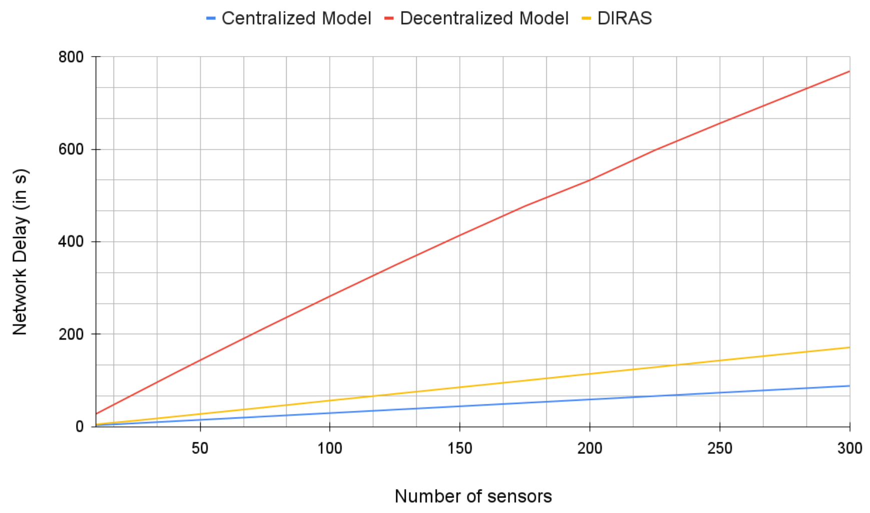


Figure 7.2: Variation in delay with change in number of sensor nodes and comparison with centralized and decentralized models

As the figures 7.1 and 7.2 suggest, DIRAS falls in between the centralized and decentralized models in terms of the performance of the network system.

7.1.1.2 Varying monitor nodes

Here, we vary the number of monitor nodes keeping the number of sensor nodes constant (refer to 6.1.1.2).

Figure 7.3 and figure 7.4 depict the packet overhead and network delay in DIRAS with the increase in number of monitor nodes. With the increase in number of monitor nodes, the overhead and delay will also increase and the major increase in the overheads and the delay will be due to the monitor position assignment because the total number of images generated in the network is constant. Thus, as expected, the graphs depict that the overhead and delay increase with increase in the monitor nodes. However, it should be noted that when the number of monitor nodes becomes ten-fold, the packet overhead and the network delay doesn't increase that greatly which is depicted by the lower slopes of the lines. Thus, increasing the number of monitor nodes doesn't greatly affect the performance of the network component. On the other hand, the increase in monitor nodes is beneficial for the network in terms of privacy (this has been discussed later).

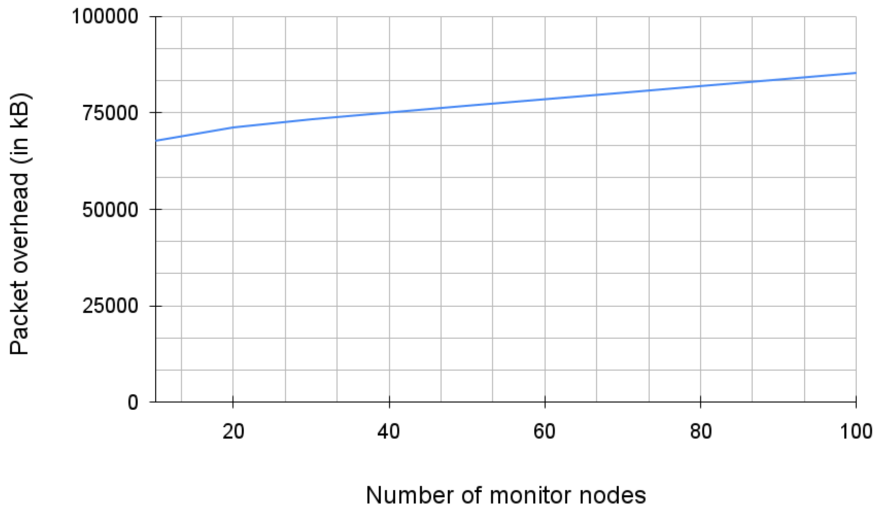


Figure 7.3: Variation in packet overhead with change in number of monitor nodes

7.1.2 DSITR

In the case of DSITR, we have not compared its working with any of the model because of its difference from classical models. Therefore, in this section, we only study the

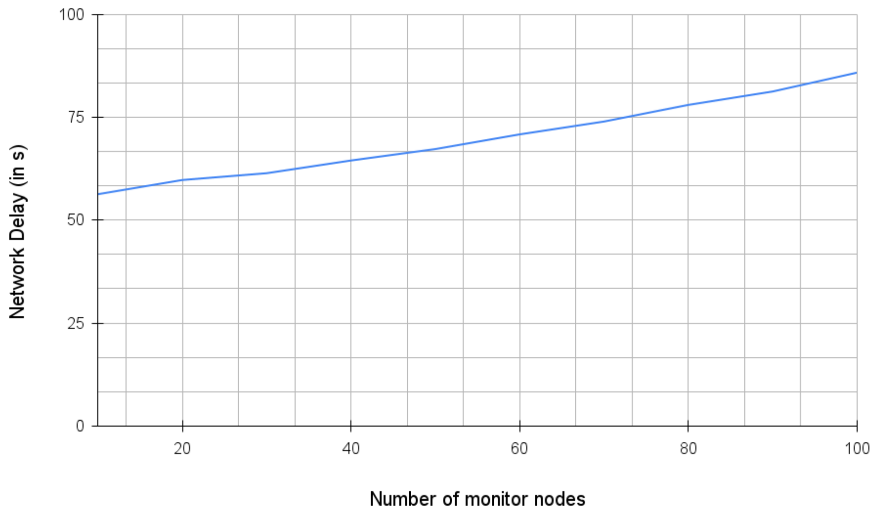


Figure 7.4: Variation in delay with change in number of monitor nodes

variation of various quantities by changing various parameters in the framework.

7.1.2.1 Varying sensor nodes

First of all, we vary the number of sensor nodes in the framework by keeping every other parameter constant.

Figure 7.5 and figure 7.6 depict the packet overhead and network delay in DSITR with the increase in number of sensor nodes respectively. As the figure suggests, the packet overhead increases gradually as the number of sensors increase in the framework. It should be noted that overheads in case of DSITR is much larger than that in the case of DIRAS. This was expected because incorporating encryption and increasing the number of replication increases the overheads. On the other hand, the latency does not vary that greatly from that in the case of DIRAS.

7.1.2.2 Varying number of connections

Here, we vary the number of monitor nodes (per shard) to which a single sensor node connected to in the framework and we keep every other parameter constant.

Figure 7.7 and figure 7.8 depict the packet overhead and network delay in DSITR with the increase in number of connections made per shard respectively. There is one interesting pattern here. Until 2 connections per shard, the overhead and latency are

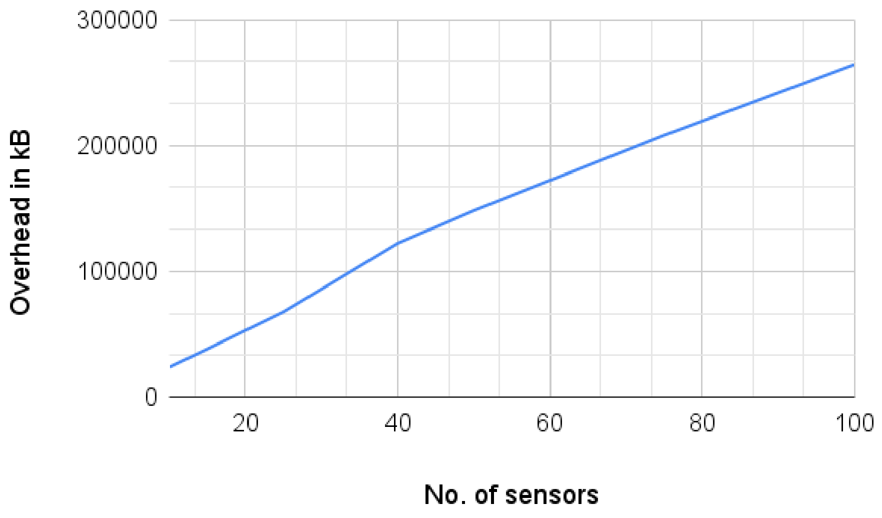


Figure 7.5: Studying the impact of number of sensor nodes on packet overhead.

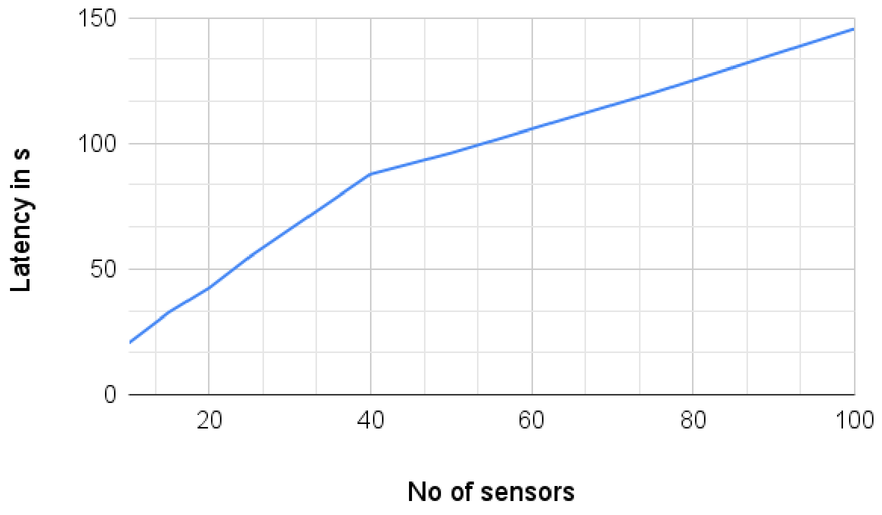


Figure 7.6: Studying the impact of number of sensor nodes on network delay.

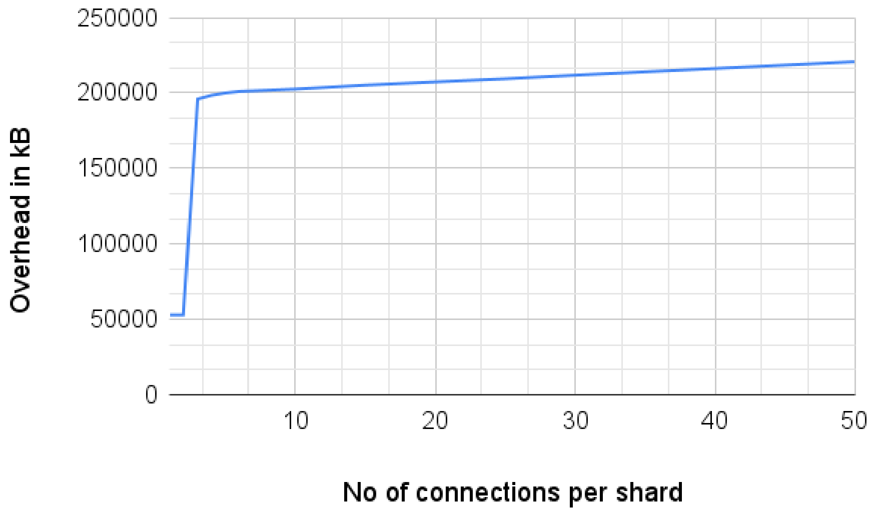


Figure 7.7: Studying the impact of number of connections on packet overhead.

pretty low. But at 3 connections per shard, there is a steep jump in the overhead and latency. After 3 connections, the increase in overheads and latency is very gradual.

7.1.2.3 Varying number of shards

Here, we vary the number of shards and the number of monitor nodes per shards and we keep every other parameter constant. We have varied those parameters in a way that the total number of monitor nodes in the framework remains constant.

Figure 7.9 and figure 7.10 depict the packet overhead and network delay in DSITR with the increase in number of shards respectively. As the graph suggests, the packet overhead and network delay increase as the number of shards increase in the framework. This is obvious because the number of connections per shard is constant. Therefore, as the number of shards increase, the number of replicas of the image regenerated in the framework increase. Thus, there is an increase in both the quantities.

7.1.2.4 Varying image size

Here, we vary the number of pixels in an image acquired by a sensor and we keep every other parameter constant. The number of pixels transferred to each shard is 0.4 times the number of pixels in an acquired image.

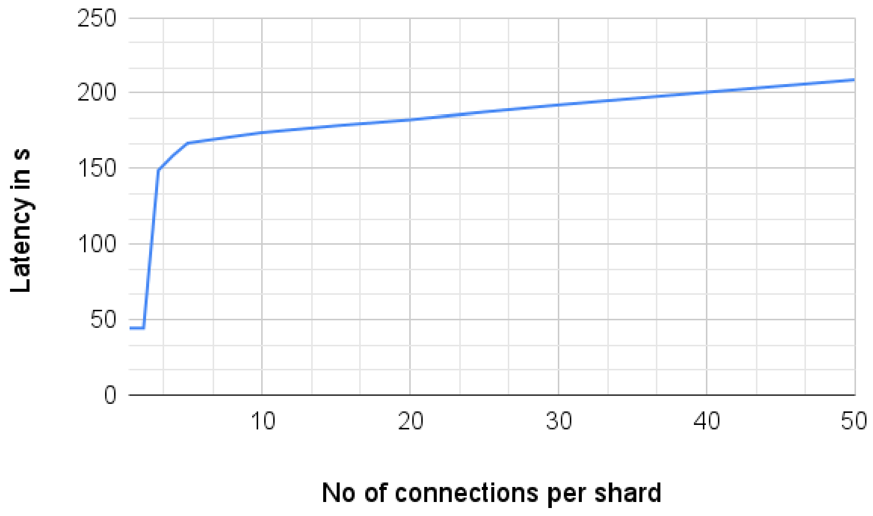


Figure 7.8: Studying the impact of number of connections on network delay.

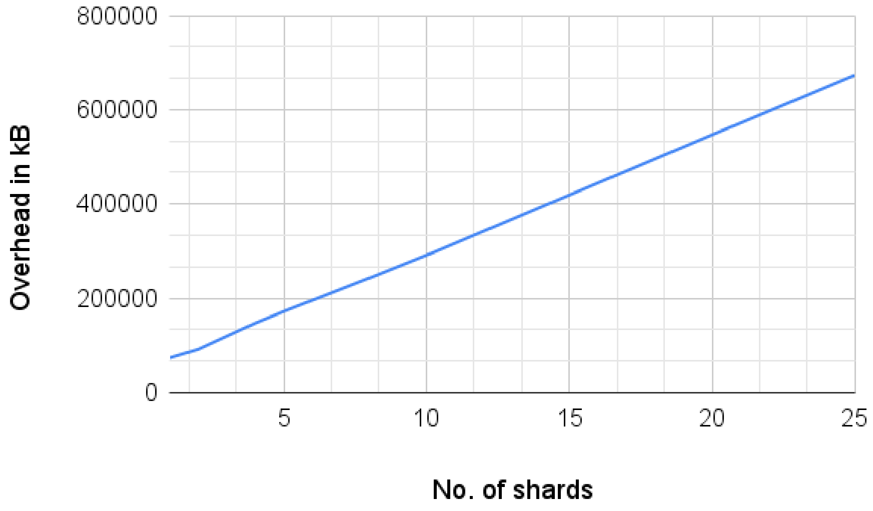


Figure 7.9: Studying the impact of number of shards on packet overhead.

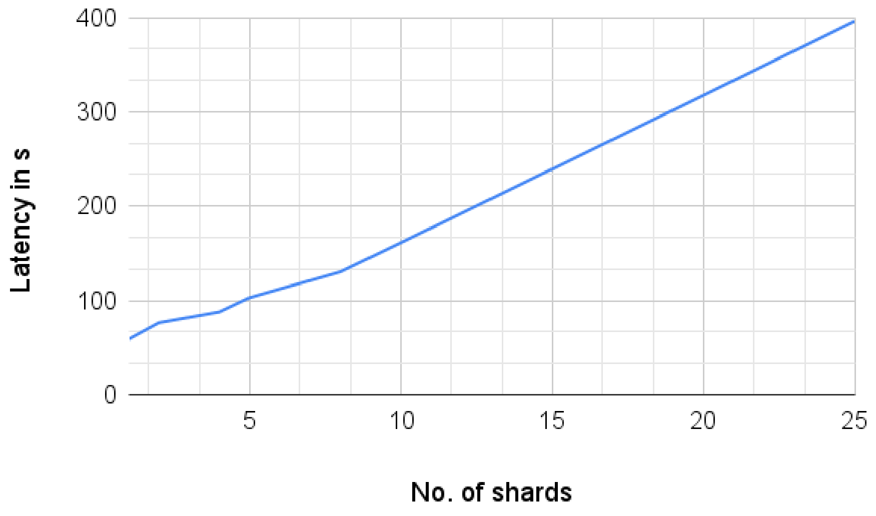


Figure 7.10: Studying the impact of number of shards on network delay.

Figure 7.7 and figure 7.8 depict the packet overhead and network delay in DIRAS with the increase in number of pixels in an image. As the graph suggests, with the increase in number of pixels, the packet overhead and network delay increase. One peculiar behavior was observed here. Until 40000 pixels (200 rows and 200 columns), the overhead and latency is almost constant. This behavior is because of the values that we have chosen for the simulation.

7.1.2.5 Varying number of pixels per shard

Here, we vary the number of pixels sent to each shard and we keep every other parameter constant.

Figure 7.13 and figure 7.14 depict the packet overhead and network delay in DIRAS with the increase in number of pixels transferred to each shard respectively. As expected, both the quantities increase with the increase in number of pixels being transferred. We can again observe a peculiar behavior here. Until half the image size, i.e., 20000 pixels, the increase in packet overhead and latency is almost negligible. After 20000 pixels, the overhead and latency increases rapidly. This behavior can be again attributed to the values that have been chosen for the simulation.

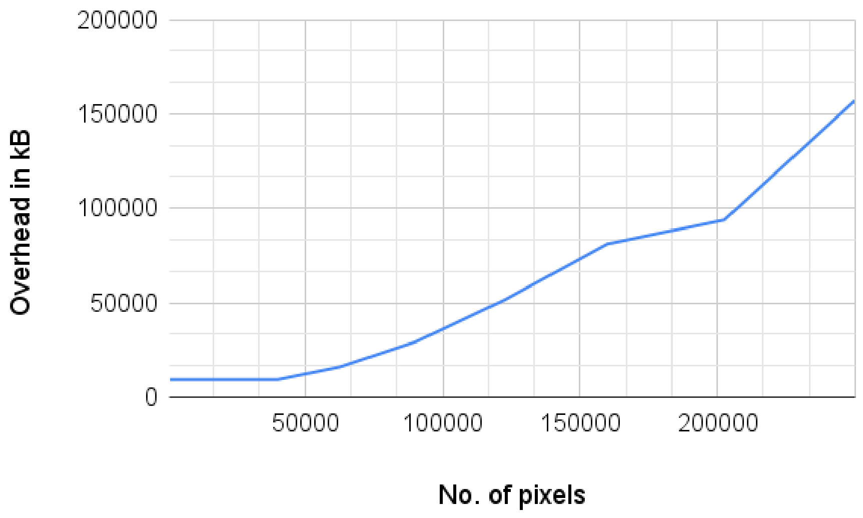


Figure 7.11: Studying the impact of image size on packet overhead.

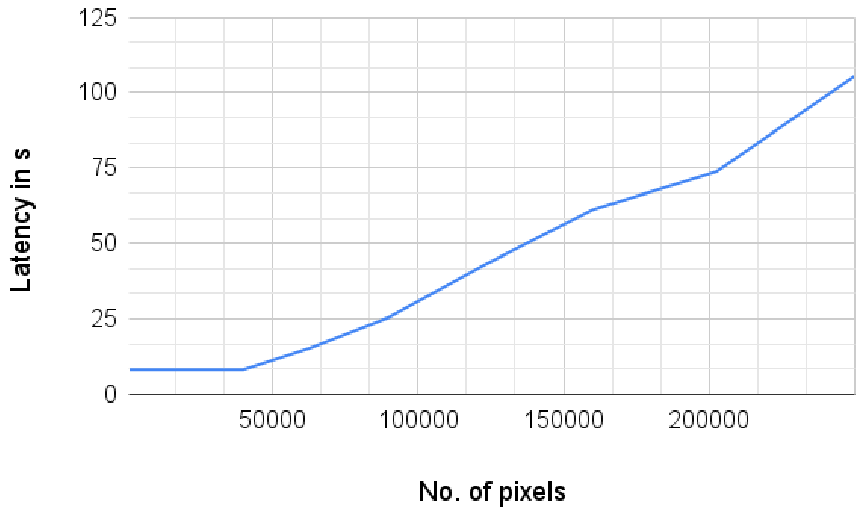


Figure 7.12: Studying the impact of image size on network delay.

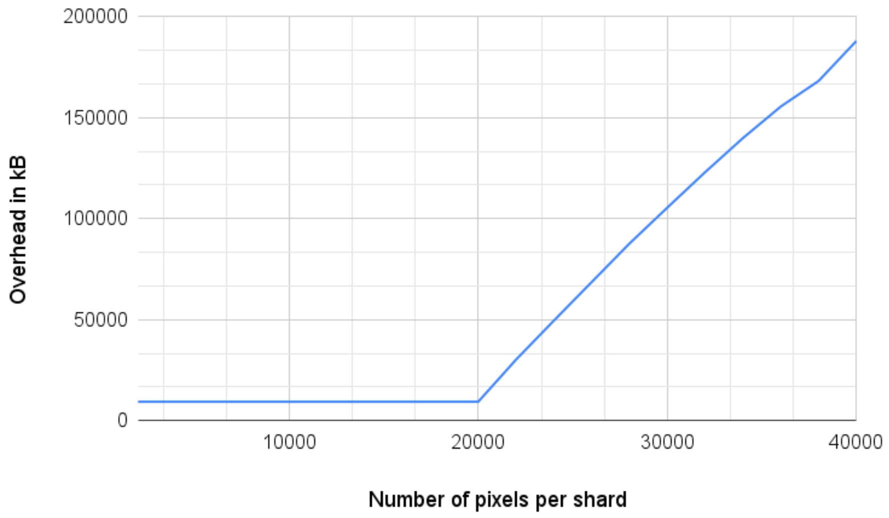


Figure 7.13: Studying the impact of number of pixels transferred to each shard on packet overhead.

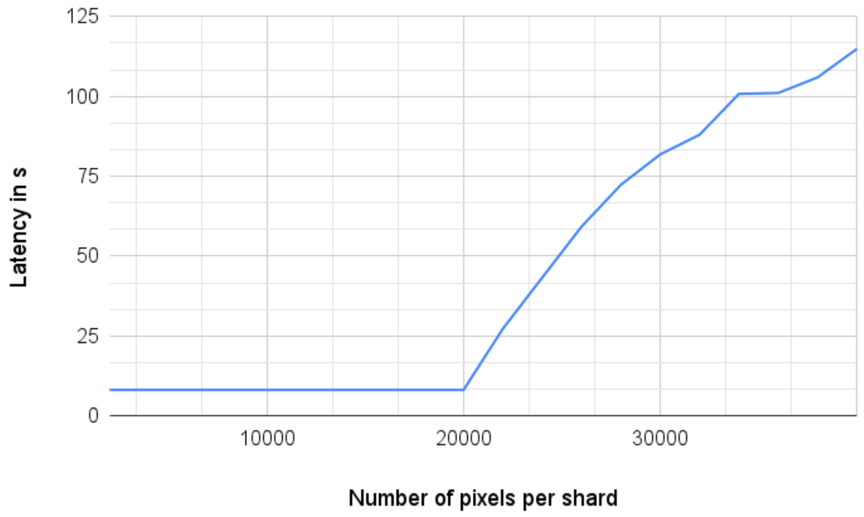


Figure 7.14: Studying the impact of number of pixels transferred to each shard on network delay.

7.2 Image Regeneration Framework

Here, we study the performance of the image regeneration component of DIRAS AND DSITR.

7.2.1 DIRAS

7.2.1.1 RPCA

Figure 7.15 depicts the actual image, image with the noise and the image regenerated from the noisy image after applying RPCA. As the figure depicts, RPCA removes the noise to a great extent. However, there is a certain degree of blurring in the regenerated image. Therefore, RPCA does not preserve all the features of the image. One reason for this inefficiency is the image that has been used not being a low-rank matrix.

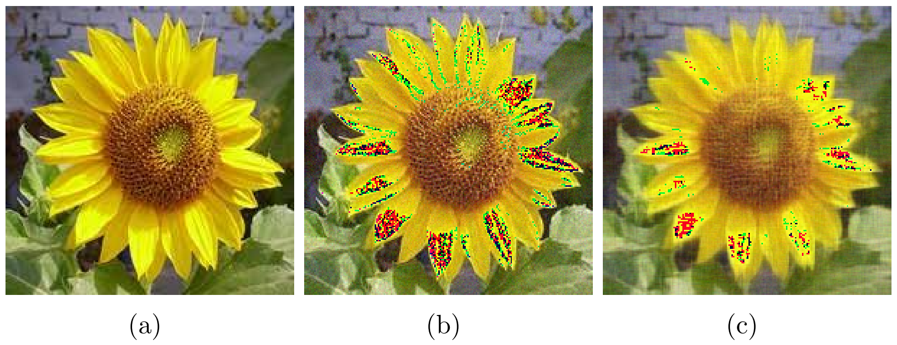


Figure 7.15: (a) Actual Image (b) Image with noise (c) regenerated image

Figure 7.16 depicts the performance of RPCA with the increase in magnitude of the noise being added to the image. We have also compared the performance of RPCA with other standard filters. The Euclidean distance between the actual image and the regenerated image reduces as the error is increased. Thus, all the filters perform better as the magnitude of error increases and RPCA performs better than all of them. However, the regenerated image obtained from all the filters vary greatly from the actual image.

7.2.1.2 RPCA and matrix completion

Figure 7.17 depicts the performance of DIRAS in case of packet drop and false data injection. As expected, the Euclidean distance of the regenerated image and the

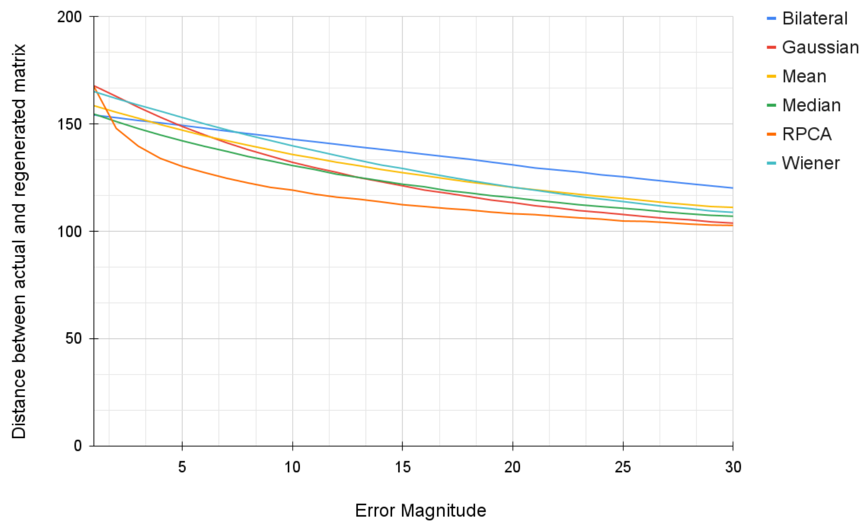


Figure 7.16: Performance of DIRAS in case of false data injection and its comparison with other filters

regenerated image increases with the increase in number of packets dropped. As the graph suggests, there is a small saturation in the Euclidean distance after 85 rows of the matrix have been dropped.

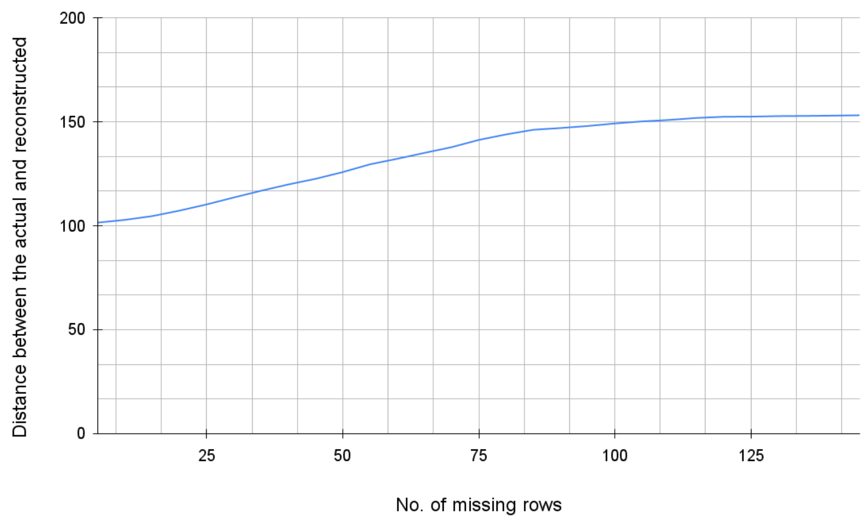


Figure 7.17: Performance of DIRAS in case of packet dropping and false data injection

7.2.2 DSITR

Next, we discuss the performance of DSITR in case of image regeneration.

Figure 7.18 depicts the performance of DSITR in terms of quality of image regenerated with the number of pixels used for regenerating the image. Here, we have used NNM for matrix completion. It can be seen that the image regenerated using 25000 pixels is very close to the actual image. Thus, our concept of using less amount of data for obtaining the near-actual data works efficiently as depicted in the figures.

Figure 7.19 depicts the distance of regenerated matrix with the change in number of pixels used for regenerating the entire image. As it can be seen, the distance between the actual matrix and the regenerated matrix is very small for number of pixels 15000. We have also compared the performance of ASD and NNM. As expected, NNM provides a more accurate matrix than ASD. However, ASD is faster than NNM in execution. This is depicted by the figure 7.20 where we can see that ASD is always faster than NNM.

Figure 7.21 depicts the impact of inter-shard communication for improving the final image quality. It is evident that the quality of image increases considerably when the regenerated images are averaged over. By averaging, DSITR removes the deviations that lie in individual images.

Next, we quantify the improvement achieved by averaging in terms of distance between the actual matrix and regenerated matrix. As depicted by figure 7.22, the distance between the matrices reduces on averaging. It should be also noted that averaging has more in case of lesser number of pixels than in the case of greater number of pixels.

7.3 Load Balancing Analysis

Figure 7.23 depicts the improvement provided by load balancing. Without the load balancing, some monitors have less computation burden while the others have more. On the other hand, the load balancing algorithm ensures that each of monitor has equal computation burden in terms of regenerating a whole image.

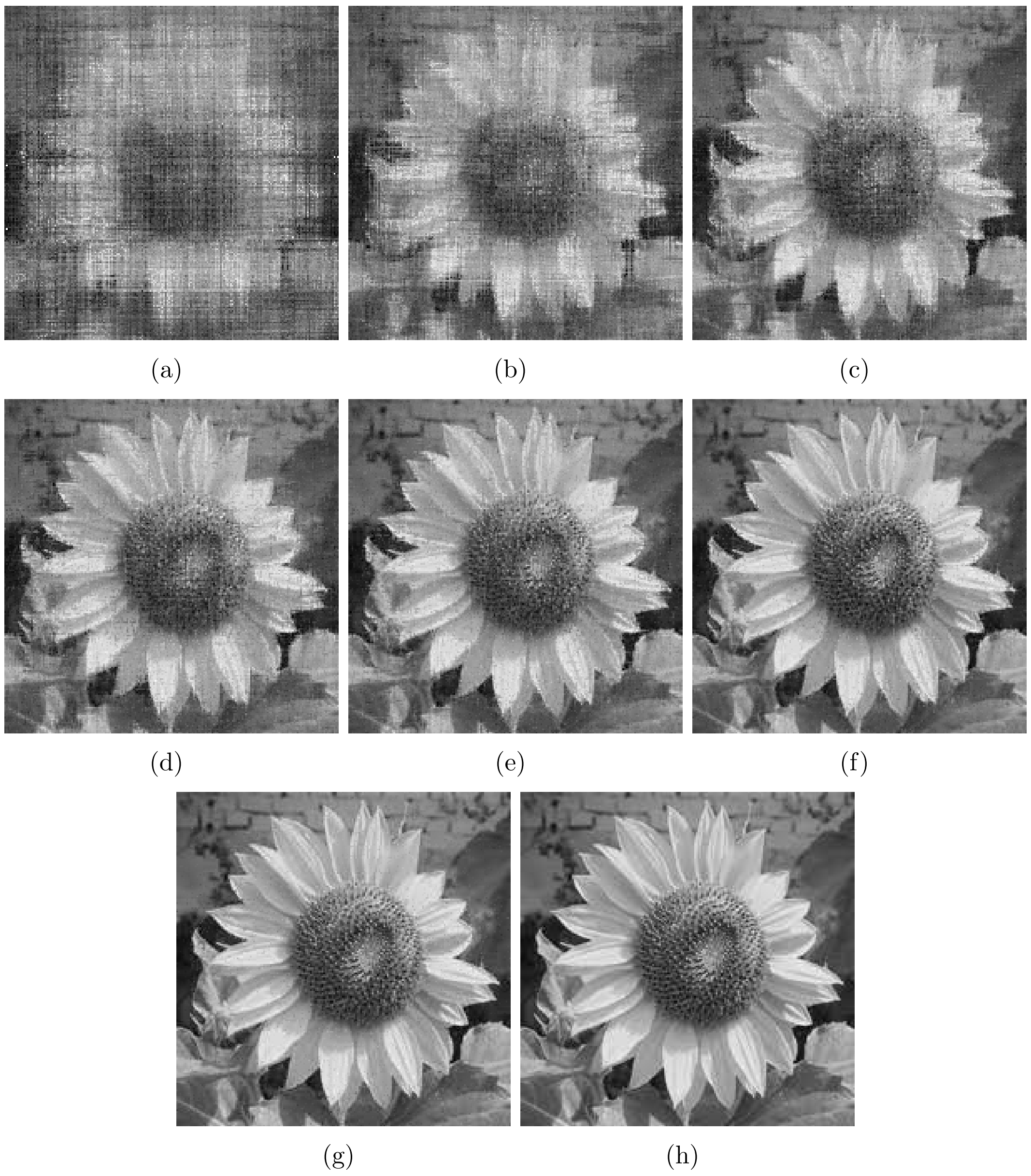


Figure 7.18: Image regenerated from: (a) 5000px (b) 10000px (c) 15000px (d) 20000px (e) 25000px (f) 30000px (g) 35000px (h) 40000px(actual image)

7.4 Privacy Analysis

Figure 7.24 depicts the amount of information revealed with the number of packets intercepted. It is evident that the Euclidean distance between the actual regenerated matrix from matrix completion and the actual matrix is quite large when the num-

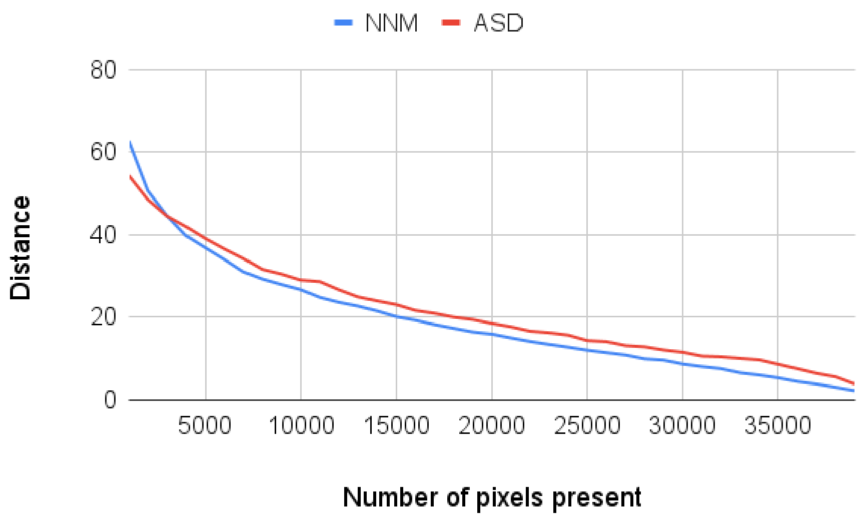


Figure 7.19: Distance of matrix regenerated with the change in number of pixels used for matrix completion

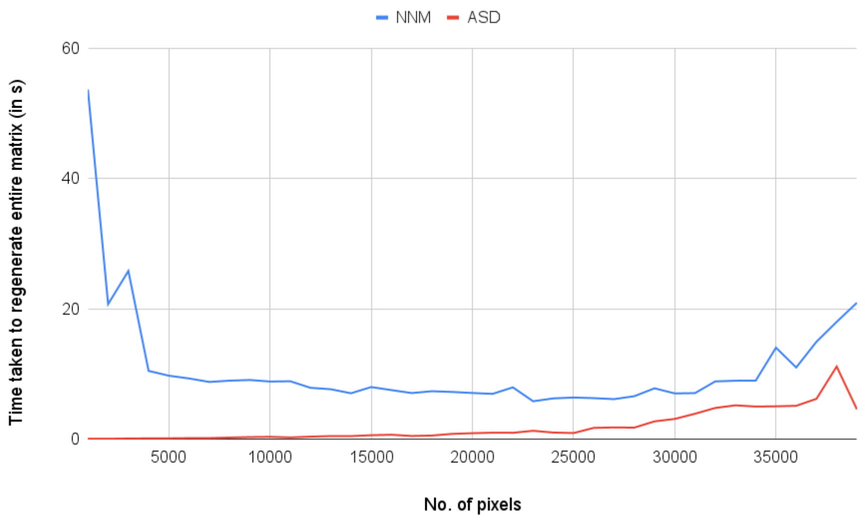


Figure 7.20: Time taken to regenerate whole matrix from some set of pixels

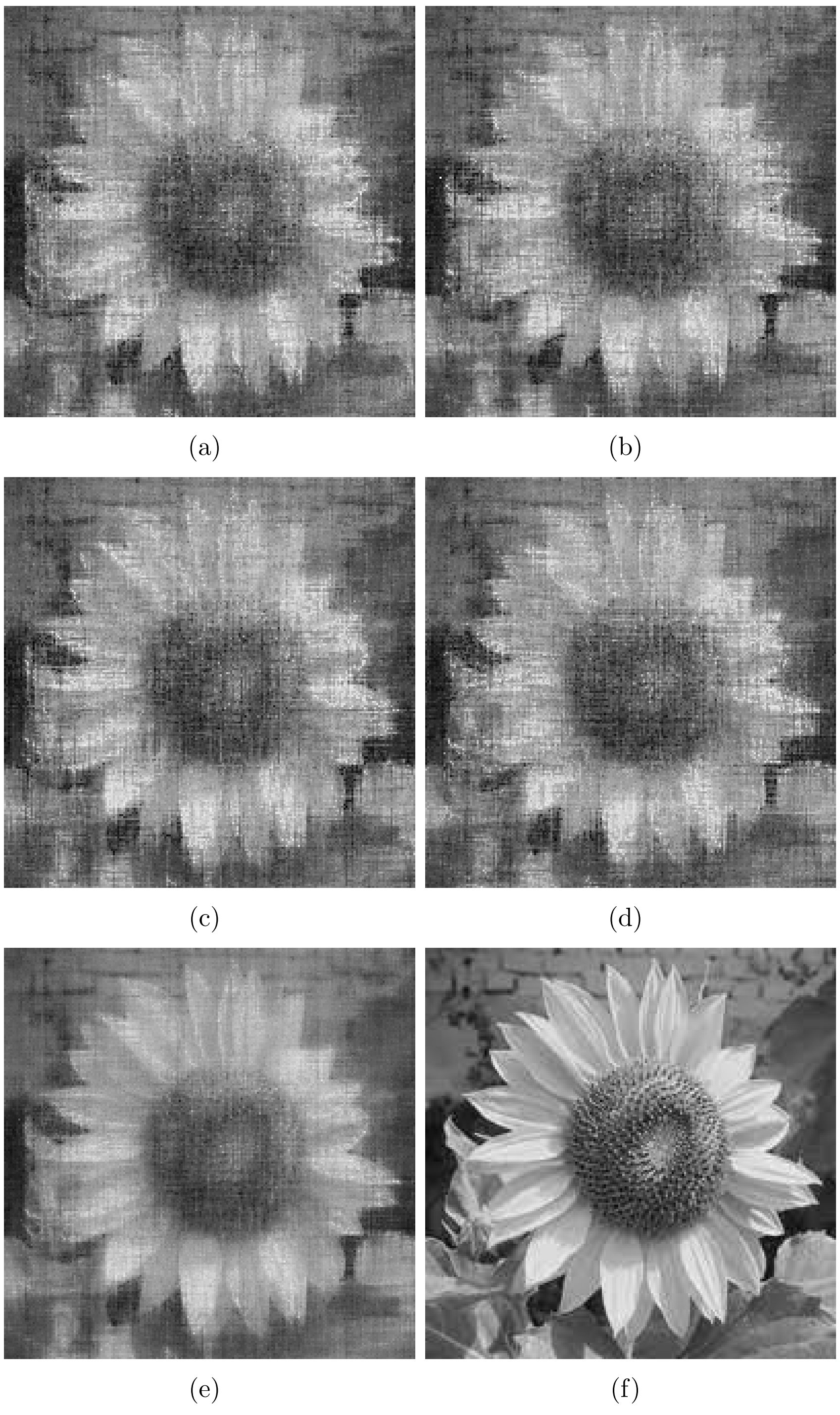


Figure 7.21: Impact of inter-shard communication: (a), (b), (c), (d): Image regenerated from 10000 pixels; (e) Image obtained after averaging the above four images; (f) Actual image

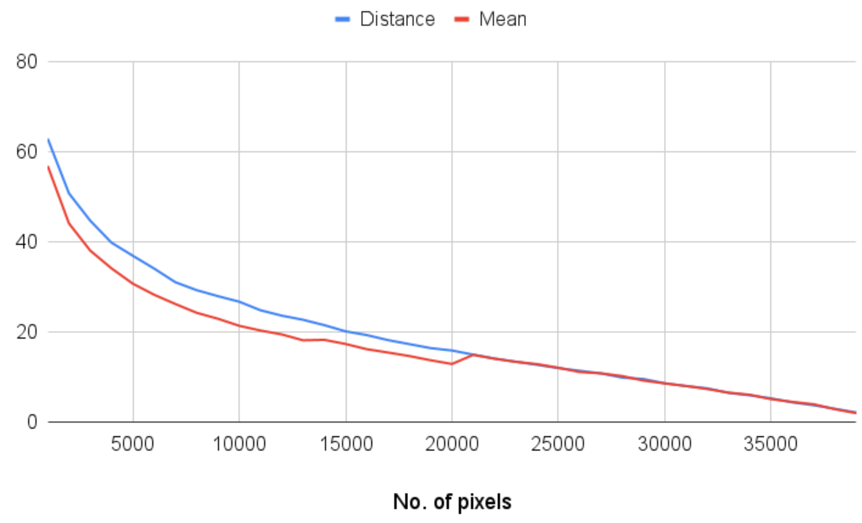


Figure 7.22: Improvement in the quality of image by inter-shard communication

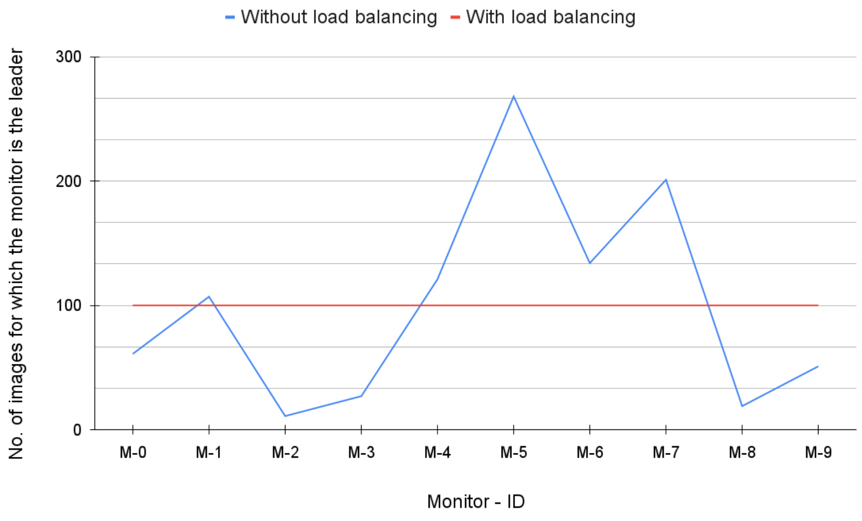


Figure 7.23: Improvement in performance because of load balancing based on the number of images for which each monitor node has been chosen as the leader

ber of rows intercepted is very small. An adversary does not have a great deal of information until he has intercepted around 500 rows. Thus, DIRAS is resistant to privacy leakage as an attacker would have to get control over numerous channels to get substantial information about the image.

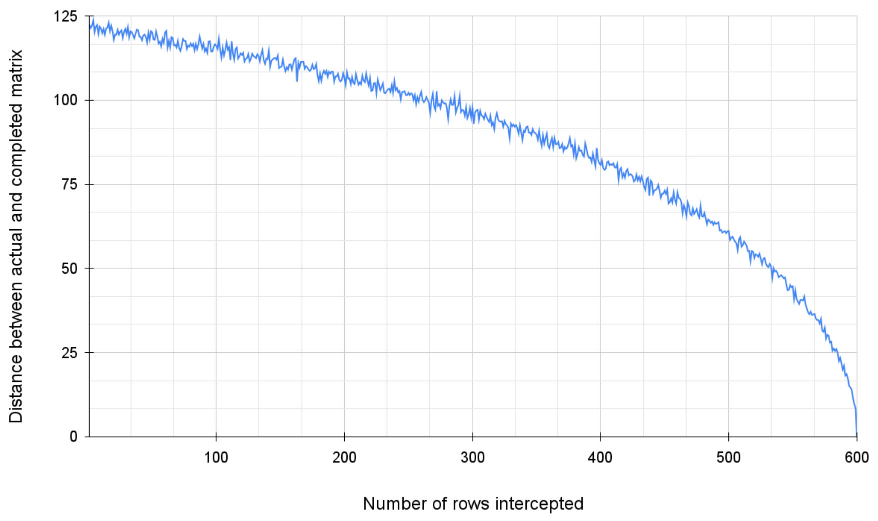


Figure 7.24: Privacy provided by DIRAS based on distance between the actual matrix and the matrix regenerated after applying matrix completion

Chapter 8

Discussion

In this section, we discuss the various implications provided by the results in Section 7. We have classified this discussion into scalability, privacy and security of DIRAS and DSITR.

8.1 Scalability Analysis

As results suggest, the overhead and network delay in DIRAS is comparable with the centralized model. Considering the benefits provided by DIRAS in terms of privacy and security, DIRAS can prove to be a better choice over a centralized architecture.

In addition, splitting of data helps in enhancing the scalability of the framework. This is because a sensor node does not have to transfer its entire image, which is generally a large matrix. Such transfer of large matrices would not only bloat the connections of the sensor node but also they will make the communications between monitor nodes slower.

On the other hand, the magnitude of overheads and latency for the same of number of sensor nodes is higher in case of DSITR as compared to DIRAS. However, this is compensated by the security provided by it.

8.2 Privacy Analysis

Splitting of messages has helped us in increasing the privacy of the data. Figure 7.24 suggests that an attacker needs to acquire a large number of rows to gain enough information of an image. The Euclidean norm of the matrix obtained from the difference between the reconstructed and actual matrices is around 90 for 400 rows intercepted.

This is difficult for an adversary to achieve. For example, consider a scenario each chunk sent by the sensor has 5 rows. Then, the adversary would need to intercept packets from 80 communication channels, which is very difficult computationally.

Another point that should be noted is the trade-off between bandwidth and privacy. Figure 7.3 and figure 7.4 depict that the bandwidth consumed and delay increase with the increase in the number of monitor nodes. However, with the increase in number of monitor nodes, the size of chunks reduces and thus, the information revealed by each chunk reduces. Thus, the network designer needs to decide the decide the number of monitor nodes based on the requirement of privacy.

Moreover, we have assumed that the adversary uses the same technique used by us for matrix completion. The attacker may use a better matrix completion algorithm for acquiring the entire image.

DSITR provides a very high level of privacy because of encryption. Due to encryption, it is nearly impossible for an adversary to acquire any information from the image. Even if we consider the worst case, where the adversary gets the keys of a node, even then it is difficult to acquire enough information because of data splitting.

8.3 Security Analysis

Here, we discuss the various attacks DIRAS defends against and how.

8.3.1 False Data Injection Attack

In this case, the adversary is in between two nodes and tampers with the data within the packet. Note that the adversary can be between a sensor node and a monitor node or two monitor nodes. The adversary changes the data such that the image reconstructed deviates greatly which leads further wrong analysis. We have mitigated the effects of this attack by deploying RPCA that separates the sparse noise from the low-rank matrix. However, as the results depict, the reconstructed matrix has a great deal of blurring and thus, the reconstructed image varies greatly from the actual image. Thus, we may need to improve the algorithm or look for new methods for the same.

This attack is impossible in DSITR.

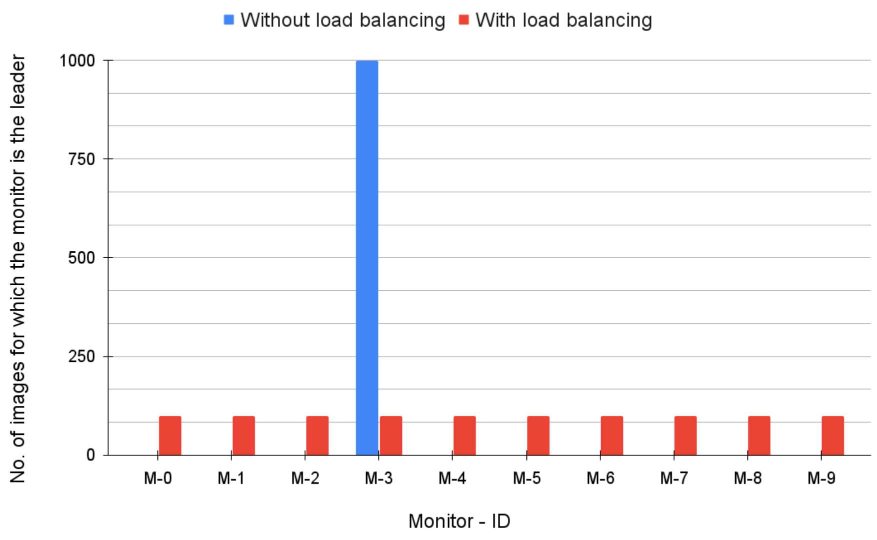


Figure 8.1: Mitigation of effect of DoS attacks by load balancing

8.3.2 Packet Drop Attack

In case of this attack, the attacker is between any two nodes and does not allow packets to pass from source node to destination. We call this the *Packet Drop Attack*. In the case of this attack, the whole image cannot be reconstructed which is really detrimental to the functioning of the systems. DIRAS reduces the effect of this attack by deploying matrix completion algorithm which helps greatly.

By changing the way data is sent in the network, i.e., by sending random pixels instead of rows of an image, DSITR has achieved much better performance than DIRAS. Even if an adversary drops some packets, the monitor node would still receive randomly distributed data points which would lead to better image regeneration than DIRAS.

8.3.3 Denial of Service (DoS) Attack

This attack is targeted at any particular monitor node. An adversary takes control of a sensor node and generates coordinates which are always close to (x_j, y_j). The $j-th$ monitor node will be selected as the leader each time. Therefore, the $j-th$ monitor node will be sent numerous packets in a very short interval of time and will be blocked from all other communications and functioning. Note that the adversary does not need to know the exact coordinates of a monitor. It can simply keep on generating coordinates which are close to (x_j, y_j). To reduce the effect of this attack, the monitor nodes generate random coordinates for every Δ. This reduces the chances of this attack greatly. However, the attacker may generate images at an extremely quick rate in a given epoch time. To solve this issue, we have proposed load balancing for selecting the leader each image. Figure 8.1 depicts how load balancing helps in countering DoS attack. The attacker sensor generates coordinates very close to monitor with ID $M-3$. Without load balancing, $M-3$ has to regenerate all the images. By integrating load balancing, the image reconstruction task is distributed evenly in the network and none of the nodes get blocked.

DIRAS and DSITR are distributed, scalable, and improve privacy and security. DIRAS is more scalable in the sense that it consumes lesser overhead. On the other hand, DSITR provides better security and privacy because of encryption. The usage of digital signature makes the chances of Sybil attacks[15] almost negligible. However, there is a dependency on third party in case of DSITR which is the CA. To remove this dependency, we may need to build identity schema within the framework. Thus, with minor changes, DIRAS and DSITR can be made deployable.

Chapter 9

Conclusion

This report presents two models for regenerating image in a distributed manner. DIRAS does not use PKI which makes it independent of any certificate authorities. This does bring in some security issues which make the image regeneration erroneous. However, the overheads and latency in case of DIRAS is not very different from the centralized model because of the less amount of information that needs to be sent in each packet.

On the other hand, DSITR tries to overcome the shortcomings of DIRAS by incorporating network sharding and PKI. The main benefit of using DSITR over using DIRAS is the improvement in the quality of the image regenerated. The design of DSITR is relatively more complex than that of DIRAS. In addition, DSITR consumes more overhead than DIRAS. However, it provides benefits over DIRAS in case of security and which make it a reliable.

Both the models have their benefits and limitations. The results presented in Section 7 show the scalability of DIRAS and DSITR, and their ability to regenerate images. The discussion provided in Section 8 gives a comprehensive view on the implications obtained from the results and the privacy and security analysis of DIRAS and DSITR. In spite of the minor limitations that are there in the systems, both of the systems provide a formidable framework for regenerating image in a distributed manner. DIRAS and DSITR take into account the features of image regeneration and security of network and data in their own way, which makes both of them versatile frameworks for CPS.

Bibliography

[1] R. I. Abdelfatah. Secure image transmission using chaotic-enhanced elliptic curve cryptography. *IEEE Access*, 8:3875–3890, 2019.

[2] S. Agarwal. Secure image transmission using fractal and 2d-chaotic map. *Journal of Imaging*, 4(1):17, 2018.

[3] M. Ahmed and A.-S. K. Pathan. False data injection attack (fdia): an overview and new metrics for fair evaluation of its countermeasure. *Complex Adaptive Systems Modeling*, 8(1):1–14, 2020.

[4] A. Al-Kaff, D. Martin, F. Garcia, A. de la Escalera, and J. M. Armingol. Survey of computer vision algorithms and applications for unmanned aerial vehicles. *Expert Systems with Applications*, 92:447–463, 2018.

[5] B. A. Alqaralleh, T. Vaiyapuri, V. S. Parvathy, D. Gupta, A. Khanna, and K. Shankar. Blockchain-assisted secure image transmission and diagnosis model on internet of medical things environment. *Personal and ubiquitous computing*, pages 1–11, 2021.

[6] B. Anderson, S. Mou, A. S. Morse, and U. Helmke. Decentralized gradient algorithm for solution of a linear equation. *arXiv preprint arXiv:1509.04538*, 2015.

[7] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[8] R. Baheti and H. Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011.

[9] C. Belthangady and L. A. Royer. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nature methods*, 16(12):1215–1225, 2019.

[10] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale modeling & simulation*, 4(2):490–530, 2005.

[11] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[12] E. J. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[13] C.-h. Chen. *Signal and image processing for remote sensing.* CRC press, 2012.

[14] J. Chen, J. Benesty, Y. Huang, and S. Doclo. New insights into the noise reduction wiener filter. *IEEE Transactions on audio, speech, and language processing*, 14(4):1218–1234, 2006.

[15] J. R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[16] G. Dougherty. *Digital image processing for medical applications.* Cambridge University Press, 2009.

[17] G. Fiore, E. De Santis, and M. D. Di Benedetto. Secure mode distinguishability for switching systems subject to sparse attacks. *IFAC-PapersOnLine*, 50(1):9361–9366, 2017.

[18] M. A. Fischler and R. C. Bolles. A paradigm for model fitting with applications to image analysis and automated cartography (reprinted in readings in computer vision, ed. ma fischler,". *Comm. ACM*, 24(6):381–395, 1981.

[19] R. Gnanadesikan and J. R. Kettenring. Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, pages 81–124, 1972.

[20] Z. Guo, D. Shi, K. H. Johansson, and L. Shi. Optimal linear cyber-attack on remote state estimation. *IEEE Transactions on Control of Network Systems*, 4(1):4–13, 2016.

[21] H. Jaidka, N. Sharma, and R. Singh. Evolution of iot to iiot: Applications & challenges. In *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*, 2020.

[22] C. Kanellakis and G. Nikolakopoulos. Survey on computer vision for uavs: Current developments and trends. *Journal of Intelligent & Robotic Systems*, 87(1):141–168, 2017.

[23] Q. Ke and T. Kanade. Robust l/sub 1/norm factorization in the presence of outliers and missing data by alternative convex programming. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 739–746. IEEE, 2005.

[24] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE transactions on information theory*, 56(6):2980–2998, 2010.

[25] P. R. G. Kurka and A. A. D. Salazar. Applications of image processing in robotics and instrumentation. *Mechanical Systems and Signal Processing*, 124:142–169, 2019.

[26] L. Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[27] L. Liang, K. Zheng, Q. Sheng, and X. Huang. A denial of service attack method for an iot system. In *2016 8th international conference on Information Technology in Medicine and Education (ITME)*, pages 360–364. IEEE, 2016.

[28] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[29] A. Nedić and J. Liu. Distributed optimization for control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:77–103, 2018.

[30] H. Nguyen-An, T. Silverston, T. Yamazaki, and T. Miyoshi. Generating iot traffic in smart home environment. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–2. IEEE, 2020.

[31] F. Pasqualetti, F. Dorfler, and F. Bullo. Control-theoretic methods for cyber-physical security: Geometric principles for optimal cross-layer resilient control systems. *IEEE Control Systems Magazine*, 35(1):110–127, 2015.

[32] B. Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12), 2011.

[33] G. Rong-xiao, T. Ji-wei, W. Bu-hong, and S. Fu-te. Cyber-physical attack threats analysis for uavs from cps perspective. In *2020 International Conference on Computer Engineering and Application (ICCEA)*, pages 259–263. IEEE, 2020.

[34] Y. Shoukry and P. Tabuada. Event-triggered state observers for sparse sensor noise/attacks. *IEEE Transactions on Automatic Control*, 61(8):2079–2091, 2015.

[35] J. Tanner and K. Wei. Low rank matrix completion by alternating steepest descent methods. *Applied and Computational Harmonic Analysis*, 40(2):417–429, 2016.

[36] J. Thevenot, M. B. López, and A. Hadid. A survey on computer vision for assistive medical diagnosis from faces. *IEEE journal of biomedical and health informatics*, 22(5):1497–1511, 2017.

[37] A. Vibhute and S. K. Bodhe. Applications of image processing in agriculture: a survey. *International Journal of Computer Applications*, 52(2), 2012.

[38] G. Wang, J. C. Ye, and B. De Man. Deep learning for tomographic image reconstruction. *Nature Machine Intelligence*, 2(12):737–748, 2020.

[39] G. Wang, J. C. Ye, K. Mueller, and J. A. Fessler. Image reconstruction is a new frontier of machine learning. *IEEE transactions on medical imaging*, 37(6):1289–1296, 2018.

[40] W. Wei, B. Zhou, D. Połap, and M. Woźniak. A regional adaptive variational pde model for computed tomography image reconstruction. *Pattern Recognition*, 92:64–81, 2019.

[41] J. Wright, A. Ganesh, S. R. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *NIPS*, volume 58, pages 289–298, 2009.

[42] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278–305, 2019.