# A DECEPTION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

By

RUIYI ZHANG

Bachelor of Science in Electronic Engineering
Sichuan University
Chengdu, Sichuan, China
2004

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2008

A DECEPTION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

Thesis Approved:

Dr. Johnson P. Thomas

Thesis Advisor

Dr. Xiaolin Li

Dr. Venkatesh Sarangan

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I thank my advisor, Dr. Johnson Thomas, for his precious advice and guidance on this research. Without his tremendous help and support, I would not be able to finish this thesis. I also would like to present my thanks to Qifan Zhang for her help in proof-reading and correcting the presentation of this thesis. Finally, I want to thank my parents for giving me life and raising me.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

# CHAPTER 1

# INTRODUCTION

## 1.1 Deceptions in Information Systems

Sun Tzu once said in *The Art of War*: "All warfare is based on deception.". To some extent, information protection is defensive information warfare. Although we don't assert that all of information protection is based on deception, the quote of Sun Tzu definitely prompts the importance of deception in modern information warfare.

Deception is a way to manipulate others' perceived reality by intentionally distorting the truth. We define the term 'deception' as processes in which the information system tries to deceive attackers by sending them falsified messages or by concealing truth from them. The reason why deceptions can be highly effective against certain kind of attacks is because information systems are generally considered to be honest. Deceptions introduce more complexity into an information system, which is a potential downside, but the enormous number of benefits makes deceptions appealing. Among others, the notable benefits are:

- deception gives the defending side chances to actively manipulate the attacking side's behavior;
- deception gives one more level of protection;
- deception increases the attacker's uncertainty;
- deception increases the sophistication required for attack;
- deception exhausts the attackers' resources;

- deception allows defenders to track the attackers' attempts at entry and respond before attackers do anything harmful.

Whaley [1] distinguishes two categories of deception, simulation (showing the false) and dissimulation (hiding the real). He also argues that everything substantive can somewhat be either simulated or dissimulated, or be done by both.

Sub-categories of simulation types of deceptions are below:

- *mimicking:* showing the false through imitation (e.g. phishing PayPal spams. They are imitating the familiar behaviors of the genuine PayPal mailinglists);
- *inventing:* showing the false by displaying a different reality (e.g. a running server sends a message "The system is shut down" to an identified hacker);
- *decoying:* showing the false by diverting attention (e.g. in a sensor network, a group of unimportant sensor nodes shows exaggerated activities to attract the attacker away from the critical ones).

Likewise, there are sub-categories of dissimulation types of deceptions:

- *masking:* hiding the real by making it invisible (e.g. making private folders HIDDEN on Windows machines);
- *repackaging:* hiding the real by disguising (e.g. camouflages used by snipers);
- *dazzling:* hiding the real by confusion (e.g. in communication, sending redundant junk packets which can be identified and dropped by authorized participants makes recovering information harder for evasdroppers).

While it can be reasonably asserted that all information systems are in many ways quite similar, there are differences between systems used in warfare and systems used in other applications, if only because the consequences of failure are extreme and the resources available to attackers are so high. For this reason, military situations

tend to be the most complex and risky for information protection and thus lead to a context requiring extremes in protective measures. While deceptions might seem overwhelming in civil and commercial projects, given the extreme natures of military applications, military applications appear to be a relevant scenario for applying deception technologies.

As far as we are aware, no effort has been devoted to develop a deception technology for sensor networks. Given the wide adoption of sensor networks, the importance of sensor networks in military and commercial applications makes us believe that it is worthwhile to explore the idea of deception in sensor networks. We believe even if information protection is not all about deception, but to some extent, deception based information protection technology could be a welcomed addition to the defenders' arsenal.

## 1.2 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are a family of wireless networks consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. Applications based on sensor networks are finding their way into numerous important domains including transportation systems, precision agriculture, battlefield monitoring, disaster zone management, homeland security and healthcare to name a few.

Types of devices included in WSNs are sensor nodes, base stations, and optionally cluster heads. Sensor nodes can be considered as miniature computers which have extremely basic functionalities in terms of their interfaces and components. They usually consist of a processing unit with limited computational power and limited memory, sensing components, communication components (usually radio transceivers), and a power source usually in the form of a battery. The base stations are distin-

Figure 1.1: Crossbow Mica 2 Sensor Node

guished components of a WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end users. The cluster heads are generally more powerful than sensor nodes but weaker than the base stations. They usually perform data aggregation operations, security related functions, and many other functionalities which are not suitable to put into sensor nodes.

## 1.3 Constraints and Security Issues in WSNs

Unlike conventional networks, WSNs have some unique constraints, including:

- limited computational and storage resources due to miniature form factor and cost restriction;
- limited and usually not replenishable power source;
- harsh deployment environment;
- dynamic network topology;
- high hardware and communication failure rate;
- unattended operation.

The fore-mentioned limitations imposed on WSNs render the majority of the current security approaches in conventional wired or wireless networks impractical for

WSNs. There are four major security challenges in WSNs' applications.

- The conflicting interests between minimizing resource consumption and maximizing security;
- the capacities and constraints of hardware of sensor nodes influence the type of security mechanisms;
- the ad-hoc networking topology renders WSNs susceptible to link attacks;
- the wireless communication scheme renders traditional security practices in wired networks impractical.

Although a lot of work has been done in protecting sensor networks, most of these works have focused on providing effective key management techniques for authentication or secure routing of messages. However, dealing with or responding to intrusions that have infiltrated the network has not been discussed. For example, should part of the network be quarantined? What is the best response to a particular attack? There are a number of possible responses to an attack. The compromised part of the network may be quarantined and isolated from the rest of the network, the sensor nodes may collaborate and counter attack by sending a continuous stream of data (this may be expensive in terms of energy), nodes may send the attacker false data with the intention of deceiving the attacker, messages may be encrypted, all nodes must authenticate themselves and so on.

## 1.4 Introduction to Deception Framework

In this thesis, we focus on developing a deception framework for protecting sensor networks. Deception gives one more level of protection. It is particularly useful if the objective is to buy time. For example, if the attack has been identified to an area in the network, extra time will enable trace-back to the exact source of the attack. Extra

time will also allow the system to bolster its defenses, for example, after an attack has been detected, it gives the base station time to inform the rest of the network that all messages will be encrypted from now onwards or all nodes must be authenticated. It may be that the objective is to prevent the attacker from moving to a different part of the network where the network is collecting critical data. The attacker is deceived long enough to complete the data collection after which the deceiving may stop. Deceiving also exhausts the attackers' resources.

Once the attacker has been identified, deceiving is an effective approach for eavesdropping attacks and the sensor network may broadcast or unicast faulty information to the eavesdropper purposely. However, deception is not a suitable response to all kinds of attacks. Attacks like node replication, masquerading or those which modify the semantics of packets may be appropriate for deception. Disruptive attacks such as Denial of Service attacks where the attacker may be expecting some responses is also appropriate for deceiving. Hijacking attacks or selective forwarding attacks where there is no explicit interaction between the attacker and the victim are not good candidates to be handled by deception, but it does not mean it is impossible. However, with attacks where deception may be applicable, it may not be the best strategy depending on the costs and payoffs.

In our approach, deception is carried out by nodes that neighbor the attacker. The nodes that sacrifice their power and time to deceive the attacker protect other nodes from being attacked. Nodes that are not neighboring the attacker can carry on with normal communications. A subset of the neighboring nodes, called sacrificial nodes, which focus exclusively on deceiving the attacker and perform no other function, are used to carry out the deception. Attacking activities are collected and analyzed at a central location, after the characteristic of the attack is extracted from collected data, a deception is conducted through sacrificial nodes.

Our contributions in this thesis are twofold. We propose a design and an archi-

tecture of sensor networks to effectively utilize deception technologies in securing the networks. A mathematical generalization of the framework is provided in forms of time series. On top of the framework, we build a deception algorithm to deal with a type of exhaustive DoS attacks by using a time series prediction algorithm called Fractal FOREcasting. An algorithm to identify changes of attacking paradigms is also proposed. We simulate the generalized mathematical model and the algorithm of identifying attacking paradigm changes with synthetic attacking data. The novelty of our work lies in the fact that, as far as we are aware, we are the first to systematically investigate the application of deceptions in sensor networks.

The rest of this thesis is started with a case study to show a possible application scenario of our framework. In Chapter 2, we briefly review related work in securing sensor networks, in deceptions in warfare related researches and information protections, and in predicting time series. In Chapter 3, algorithms and procedures of our deception framework are detailed. We present the simulation results and analysis in Chapter 4. In Chapter 5, we conclude our findings and offer insights for future work.

### 1.4.1 Case Study

Now let's consider a Battlefield Surveillance System which utilizes Sensor Networks technologies. The Surveillance System is placed at an important gateway of the battlefield. The owner of the Surveillance System wants to prevent opponents from crossing the important gateway. Upon discovering the Surveillance System, an opponent has several choices to deal with the Surveillance System. They could destroy the system by physical forces, but this action could alert the owner of the Surveillance System and lead to a failed mission. They could also choose to invade into the Surveillance System. By capturing several nodes in the network, the opponent has chances to be able to trick the Surveillance System by sending forged information. This is a typical scenario of attacks in sensor networks.

Given a functional and effective Intrusion Detection System, the defender may find out the suspicion at the gateway. Upon discovering the attack, the defender could either launch some sort of Attack Recovery techniques, i.e. Encryption Key Revocation, Nodes Exclusion. The downside of those techniques is that they will let the attacker become aware that he has been detected. With our deception technique, the defender can trick the attacker into thinking the attack is still effective, but in reality, the attacking nodes are already quarantined from the network. Since the attacker thinks the attack is in their hands, they would choose to move their troop into the gateway area. The defender will then surprise them with a sudden attack.

This is a typical scenario of what our Deception technology can offer. Our intention is to offer an active mean of defense to the arsenal of network defenders.

# CHAPTER 2

# REVIEW OF LITERATURE

## 2.1 Security Issues and Advancements in WSNs

In this section, we briefly review previous work in security of sensor networks. The problem with asymmetric cryptography, in a wireless sensor network, is that it is typically too computationally intensive for individual nodes in a sensor network. Although, this is true in the general case, [2][3] show that it is feasible with the right selection of algorithms. Symmetric cryptography systems have been proposed including 3DES (Triple DES), RC5, AES [4]. Eschenauer and Gligor propose a key predistribution scheme [5] that relies on probabilistic key sharing among nodes within the sensor network. Further enhancements have been proposed in [6][7]. The LEAP protocol [8] is based on the observation that no single security requirement accurately suites all types of communication in a wireless sensor network. Therefore, four different keys are used depending on whom the sensor node is communicating with, for example, one key is used for group communication etc. Chan and Perrig [9] describe a mechanism for establishing a key between two sensor nodes that is based on the common trust of a third node somewhere within the sensor network. Secure mechanisms for broadcasting and multicasting have also been proposed [10][11].

## 2.2 Deception Technologies in Information Protection

The deception technology we are proposing is essentially a way of dealing with attackers. Traditionally only defensive approaches are adopted. The reason why our

approach is offensive is because in our approach we try to manipulate the attacker's activities. We briefly review a number of approaches which have been identified to defend against attacks. For example, Wood and Stankovic [12] defend against attacks by identifying the compromised part of the sensor network and effectively routing around the unavailable portion. They describe a two-phase approach where the nodes along the perimeter of the attacked region report their status to their neighbors who then collaboratively define the jammed region and simply route around it.

The applications of deception to the defense of information systems are relatively new. The most notable application of deception in modern networked information system is the Honeynet [13] in which a fake but attractive network is built in order to mislead attackers as well as waste attackers' time and resources. Some impressive results have been shown for conventional information systems [14][15][16]. But, due to the differences between sensor networks and conventional information systems, theories developed for conventional information systems are not transplantable to sensor networks. As far as we are aware, no effort has been devoted to develop a deception mechanism in sensor networks.

### 2.3 Overview of Time Series Prediction Technologies

A large number of linear time series models exist, such as $AR$, $MA$, $ARIMA$, $ARFIMA$[17][18], and so on. Use $ARMA(M, N)$ as an example, the model is illustrated in the equation below.

$$x_t = \sum_{m=1}^{M} a_m x_{t-m} + \sum_{n=0}^{N} b_n e_{t-n} \tag{2.1}$$

where $x$ denotes the value at a given time $t$. $x_{t-m}$s are values prior to time $t$. $e$s are error terms for each time points. The sum of $a_m x_{t-m}$ represents the internal dynamics, and $b_n$s are weighing factor for input $e_t$. The disadvantage of linear model predictions is that it assumes linearities in underlying process, which is not true for

non-linear processes.

Neural Networks [19][20] emerge as another popular approach to predict time series. The base of Neural Network approaches is to find a function $f$ which is the best fit for $x_t = f(x_{t-1}, \ldots, x_{t-w})$. They are all similar except the methods of estimating $f$ are different. The idea of Neural Networks is sound and simple. However, it suffers from the outstanding problems of all Neural Networks: large number of parameters and long training time. Hidden Markov Models are also used for predictions of non-linear time series [21][22]. But the algorithm of finding parameters is $O(N^2)$ for number of nodes $N$, therefore the scalability of HMMs is questionable.

# CHAPTER 3

# METHODOLOGY

Expectations play a key role in the susceptibility of the target to deception. If the deception presents observables that are very far outside of the normal range of expectations, it is likely to be hard for the target to ignore it. If the deception matches a known pattern, the target is likely to follow the expectations of that pattern unless there is a reason not to. Essentially, the deception framework tries to meet the expectation of the target by mimicking a well known pattern of the network traffic using minimal resources.

In [23], we proposed a simplistic approach which yields promising simulation results. However, many issues are still remained unaddressed. Main critics about the simplistic approach include 1) it is tightly associated with a specific attack type, which is an exhaustive DoS attack; 2) the framework is not extensible; No other deception algorithms can be introduced. To address these outstanding issues, in this thesis, we introduce an extensible deception framework. A noticeable feature of this framework is that it is a generic deception framework, where additional deception methods can be plugged in effortlessly.

We assume that an intrusion detection system will detect the type of attack and then the vicinity of origin or the source of attack. Deception is particularly useful when the attacker is able to monitor the attack or receive some feedback arising from the attack. The attack we look at specifically is a denial of service attack. Here the attacker broadcasts requests to the sensor nodes. The attacker is more powerful than the sensor nodes and can broadcast to nodes that may be multiple hops away. Each

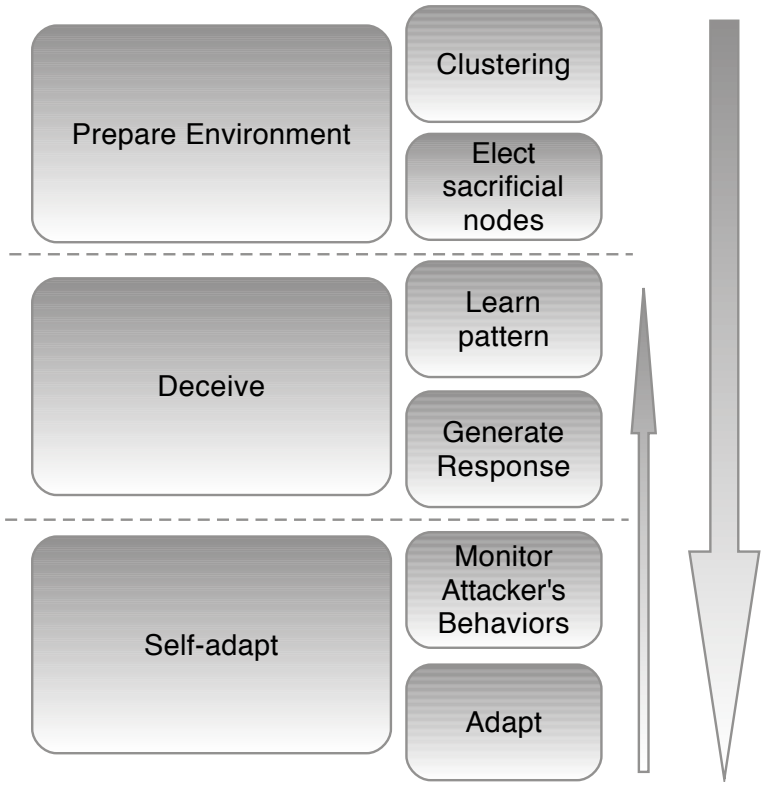


Figure 3.1: Logical Structure of The Deception Components

sensor can only communicate with its immediate neighbors. All the nodes including those that are multiple hops away from the attacker receive a request, respond to each request, thereby tie up the network to responding to the attacker's requests. The attacker is able to determine the success of his attacks by monitoring the rate of response to the request.

A deception will be conducted in three steps (Fig. 3.1):

- Initiate environment: form clusters, elect sacrificial nodes
- Respond to adversary: learn attacking pattern, generate responses
- Self adaptation: monitor behaviors of adversary, adapt

## 3.1 Design and Architecture

The primary goal of the architecture is to provide an infrastructure that is energy efficient and enables the deception framework to respond to attacks in a timely fashion. A distributed computation infrastructure which eliminates delays caused by long trips of packets and minimize energy consumptions is proposed.

Fig. 3.2 shows a heterogeneous physical infrastructure which consists of sensor nodes (dots in Fig. 3.2), one or more base station(s), and a group of Distributed Deception Agents (stars in Fig. 3.2). Distributed Deception Agents (DDAs) act as cluster heads which are specifically designed for the purpose of deceiving attackers. Comparing to ordinary sensor nodes, they have stronger computational capability, larger storage capacity, longer communication range, and hardened security. DDAs are used as the localized center place of local data processing and decision making. A layered communication hierarchy is formed. One layer is the communication of sensor nodes. The other layer is the DDAs and base station(s). Given stronger communication ability, DDAs communicate with each other and the base station in long range wireless links.

The sensor field is divided into sub-zones. Each sub-zone has at least one DDA in it. The deception actions are executed by sacrificial nodes who are selected among all nodes therefore their role is changed when required. Criteria of choosing optimal set of sacrificial nodes are presented below.

Distributed Deception Agents obtain an overview of sub-zones by surveying digests from individual nodes. A digest is a series of average values over a certain period of time called time windows. Each node keeps two digests of its interaction with an adversary. Digest $Digest_{in,i}$ records inbound requests coming from the adversary to node $i$, and digest $Digest_{out,i}$ records outbound responses sent from node $i$ to the adversary. Whether one or both digests are used by DDAs depends on requirements of deception algorithm on DDAs. Thus only the required digest will be transmitted

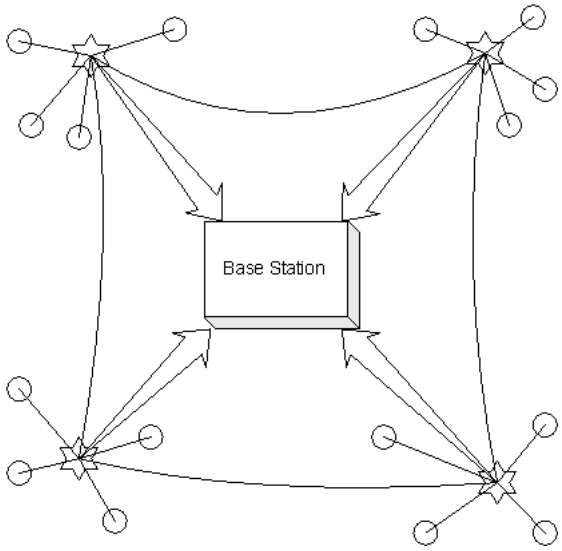


Figure 3.2: Network Architecture

to nearby DDAs.

Fig. 3.3 displays "Who does what" relations of hardware components and logical components in our deception framework. The tearoffs of the logical component Decision means that this functionality spreads in both DDAs and base stations. The base stations are responsible for making high level decisions, such as whether to deceive or not. Localized decisions, such as choosing sacrificial nodes, are made by DDAs.

### 3.1.1 Secure Deception Facilities

The proposed approach employs a simple key management scheme. A group key enables the base station to communicate with the entire network. A pairwise key also enables the base station to communicate individually with each node. If an attacker is detected, the base station uses the pairwise keys to reset the group key so that the attacker (assuming it has access to the previous group key - it may be a sensor node) is excluded from the group. Hence, the rest of the network can communicate normally without the attacker being able to read the messages. The base station informs the sacrificial nodes to use their individual keys to act as sacrificial nodes and sends them the different parameters.

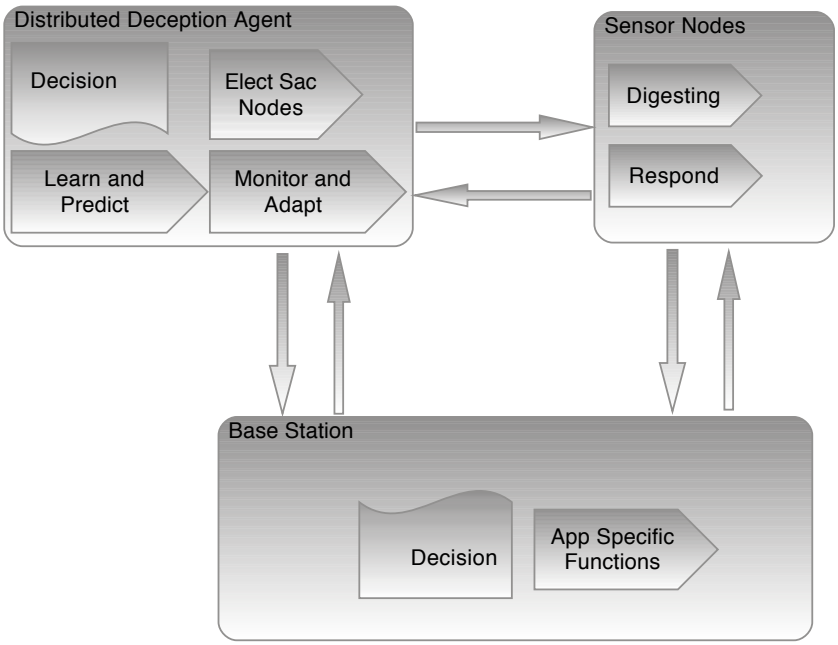


Figure 3.3: Physical Structure

Another critical concern of the sacrificial nodes is that it carries executable code for deception functionalities. Since we assume the adversary is able to tamper individual nodes with ease, it is reasonable to assume that there are high chances of executable code containing deception logics being reverse-engineered by the adversary. That in turn leads to an ultimate failure of our deception technology.

Remote code installation [24] could be of great use in this scenario. Before a breach discovered, all sensor nodes contain only code for regular operations. Code related to deception are only stored in DDAs. When an attack is detected by Intrusion Detection System, the selected sacrificial nodes will receive installation of code for deception functionalities from a nearby DDA. As DDAs are generally considered more secure than individual sensor nodes, the deception code are protected from tampering and the defender's intention of deception is well hidden from the attacker.

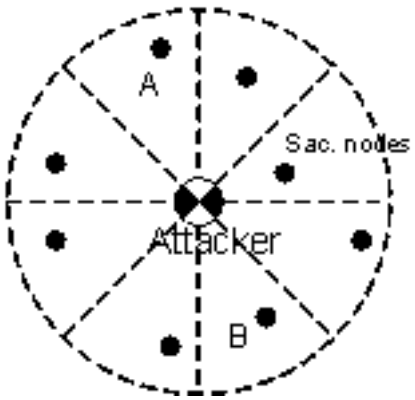


Figure 3.4: Sectors Surrounding An Attacker

### 3.2 Determine Sacrificial Nodes

The sacrificial nodes are nodes who sacrifice their power and time to deceive the attacker and prevent other nodes from being attacked. They are neighboring nodes to the attacker and within communication range of the attacker. The DDAs decide how many sacrificial nodes are needed to carry out the deception. They try to minimize the number of sacrificial nodes while make sure that the chosen nodes are capable of fulfilling the deception mission. The layout of sacrificial nodes are also decided by DDAs. It is important because 1) evenly scattered sacrificial nodes will give the attacker the impression that the response to the attacker's requests are coming from all the surrounding nodes; 2) over concentrated sacrificial nodes will jeopardize normal nodes' abilities of sensing information in that region. The DDAs use the following heuristic approach to choose the sacrificial nodes.

We assume the DDAs are able to determine the upper bound of the number of responses an attacker expects in a typical attack (given by specific deception algorithm). The number of sacrificial nodes is determined as $\lceil n/m \rceil$, where $n$ is the number of responses the attacker expects in a time unit and $m$ is the maximum number of packets a sacrificial node can transmit in the same time period.

*Power Level (P):* The power level is the power remaining in the battery of a node.

A node which has power level lower than the minimal power level will not be chosen as sacrificial node.

*Sector:* The communication region of the attacker is divided into $N$ equally sized sectors. The perimeter of the region is not defined by the communication range of the attacker, rather, it's defined by the communication range of the sensor nodes since only nodes within the region can send packets to the attacker. In Fig. 3.4 the attacker is surrounded by 8 sectors. The number of sectors is defined by the DDA.

*Opposite Sector:* A sector which is at the opposite side of the communication region regarding the attacker. For example, sector $B$ is the opposite sector of Sector $A$ in Fig. 3.4.

### 3.2.1 Initialization

We use a heuristic approach to get a set of sacrificial nodes from all nodes within the communication range of the attacker (all nodes within the circle area in Fig. 3.4). The procedure is as below:

1. Every node maintains a neighbors list which contains all neighbors to and from whom the node can send and receive packets. Nodes maintain the list by periodically sending beacon messages and receiving acknowledge messages to neighboring nodes. The acknowledge message contains basic information about the neighboring node, such as power level.

2. Assume the DDA decided that $k$ sacrificial nodes are needed for the deception. The DDA first randomly picks a node from the communication region. Any node with the attacker in its neighboring list is considered within the communication region of the attacker. We denote the node as $s_1$. A flag called VISITED is put on a visited sector;

3. Randomly pick a node from the opposite sector of the sector from which node

$s_1$ is picked. If there is no node is eligible to be a sacrificial node in the opposite sector, neighboring sectors of the opposite sector will be checked;

4. The next sacrificial node is randomly picked from unvisited sectors, in this case all sectors except sector $A$ and $B$;

5. The process of picking the 4th sacrificial node is similar to step 2;

6. Repeat step 1-4 until all sectors are visited at least once;

7. Clear all VISITED flags, and repeat step 1-5 until $k$ sacrificial nodes are chosen.

### 3.2.2 Nodes Rotation and Failover

Rotating roles of deceiving (sacrificial nodes) and sensing (regular nodes) among sensor nodes benefits the deception mission in two aspects. Firstly, rotating reduces the unevenness of power levels between deceiving nodes and sensing nodes, thus prolongs the overall life span of the network near the attacker. Secondly, since more nodes participate in the deceiving activity, it may give the attacker an impression that the attack is more influential than it really is. Failover is a mechanism of switching to redundant or standby nodes in case of failures or unanticipated termination of a node. Due to the high power and functional failure rate of sensor nodes, we consider the capability of failover is a must have.

We resolve the rotating and failover operations in following procedure.

1. Upon a node $Node_n$ receives deceiving instructions $I$ (the instructions encapsulate information for a node to conduct deception independently) from a DDA, it elects a neighboring node as a standby node $Node_s$, and subsequently transmits $I$ to $Node_s$. The power level of the standby node has to be higher than the electing node.

2. For rotating purpose, after a pre-defined amount of time, $Node_n$ notifies $Node_s$ to initialize deceiving with instructions $I$. It then shuts down the deceiving functionality and starts to act as a sensing node.

3. For failover purpose, the standby node $Node_s$ periodically sends a heartbeat request to $Node_n$, $Node_n$ would respond to the heartbeat request with a heartbeat response. The standby node $Node_s$ will start to deceive if certain amount of heartbeat responses are missing in a row (heartbeat requests contain sequence number).

4. In both cases, during the transition of a standby node to a sacrificial node, $Node_s$ elects a standby node and transmits $I$ to the standby node.

## 3.3 Pattern Learning and Prediction

We consider the information received by a DDA is a streaming time series input. By streaming, we mean the time series is continuous and never be ended in the course of deception. In this section, we model the deception system as a time series and integrate a time series prediction technology into our deception framework.

### 3.3.1 Problem Formulation

Data flows in the network are modeled as a collection of time series $T = \{S_{ij}\}$ $(i, j \in \{1 \dots n\})$ where $\{1 \dots n\}$ denotes sensor nodes (including DDAs). Data flows received by Distributed Deception Agents (DDAs) are also modeled as a collection of time series $D = \{X_k\}$ where $k$ denotes DDAs. $D$ is a subset of $T$. Hence the time series of individual DDA is $X_k = \sum_l S_{lk}$ where $l$ denotes neighbors of the DDA.

The application's specific payloads (e.g. temperature, vibrations, etc.) carried by the data flow can be modeled as time series. These information represent application domain knowledge. We call them logical information. Similarly, physical character-

istics (e.g. transmission rate) of the network can also be modeled as time series. They are independant to the application domain and closely related to the underlying network structures. We name them physical information. Whether the logical information or physical information are collected depends on needs of the specific deception.

In this thesis, as a proof of concept, we only consider deceptions in which only one property of the network, be it logical information or physical information, is involved. However, we believe a deception system with multiple properties can be done by using a collective time series or multiple independent time series. This is the topic we would further investigate in future work.

Since in our use case only one DDA will be operating in a sub-zone in which an attacker presents, analyzing the operation of a single DDA suffices to demonstrate the capability of our deception method. The time series on a single DDA is defined:

$$X = \{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots\} \tag{3.1}$$

where $X$ is an infinite time series. Subscripts $\{1 \ldots n \ldots\}$ denote the time point of each value.

We define a simplified model of requests and responses as a $1:1$ mapping. The assumption we made is that if the type of attacks and the physical system do not change, the mapping function remains the same. Function $ReqToResp()$ is an inverse version of function $RespToReq()$. Thus, considering the network as a black box, we have a relation between $T_{req}$ and $T_{resp}$, where $T_{req}$ and $T_{resp}$ are collective time series of requests and responses respectively.

$$ReqToResp(T_{req}) = T_{resp} \tag{3.2}$$

$$RespToReq(T_{resp}) = T_{req} \tag{3.3}$$

By the definition, a prediction of future input can be presented as a prediction of future responses.

$$Predict(T_{req}) = Predict(RespToReq(T_{resp})) = RespToReq(Predict(T_{resp})); \quad (3.4)$$

Hence, by predicting future responses, the future requests are well reflected.

A training set is a set of time series generated by a same physical system in different time periods.

$$TS = \{X_1, \ldots, X_N\} \quad (3.5)$$

where $X_i = \{x_{it}, x_{i(t+1)}, \ldots, x_{i(t+l)}\}$. $x_t$ is the value of the time series at time $t$, and $l$ is the length of the time series $X_i$.

Also, a query sequence is provided to be predicted upon with. The query sequence and the training set must come from the same physical system.

$$Q = \{q_1, q_2, \ldots, q_n\} \quad (3.6)$$

Therefore, the problem of prediction future responses becomes: given a training set $TS$ and a query sequence $Q$, predict $q_{n+1}, q_{n+2}, \ldots$ and so on. A simple form of this prediction problem called $Predict - 1Problem$ is that there is only one training sequence in the training set, which is also the query sequence.

### 3.3.2 Fractal-based Delay Coordinate Embedding Prediction

We expect the ideal prediction algorithm to be able to take an infinite streaming time series input and produce a streaming output. Due to the limited hardware and the large input data set, the parameter learning algorithm has to be a One Pass Algorithm, which means all input data will be scanned only once and then be

discarded. During the initialization, the algorithm finds parameters by itself without manual setup, which means no human intervention will be needed.

Chakrabarti, D. and Faloutsos, C. [25] proposed an time series prediction approach using a forecasting method called "Delay Coordinate Embedding". This has the advantage of being able to handle periodic as well as chaotic datasets. Its disadvantage is that its parameters have to be set manually. Authors of [25] developed F4 (Fractal FOREcasting) system on top of "Delay Coordinate Embedding", in which they provide automatic methods to do parameter induction, without any human intervention. This results in a black-box which, given any time series, can automatically find the optimal parameters and build a prediction system.

*DEFINITION 1: Delay Coordinate Vector: The vector* $b = [x_t, x_{t-\tau}, \ldots, x_{t-l\tau}]$ *is called a delay coordinate vector because its terms are the time-delayed data values from the time series.*

*DEFINITION 2. Lag Plot: A plot of this* $(l+1)$ *dimensional vector space is called the Lag Plot for lag* $l$*. This vector space is also called the Phase-Space.*

The theoretical basis of "Delay Coordinate Embedding" is sound and proved in [26][27]. Thus, the problem left can be divided into two sub-problems.

1. finding optimal lag length $L_{opt}$;

2. finding optimal number of nearest neighboring data points $k_{opt}$

In [25], the author uses a plot called FDL (Fractal Dimension vs. Lag) to decide optimal lag length $L_{opt}$. As the lag length $L$ approaches the optimal value, the FDL plot is flatting out. Therefore the optimal value is found at a point where the FDL plot is not flatting out further. To decide $k_{opt}$, a heuristic method combining with observations are used. Being verified by experiments, the optimal number of nearest neighboring data points $k_{opt} = 2f + 1$ where $f$ is the intrinsic dimensionality of the training set.

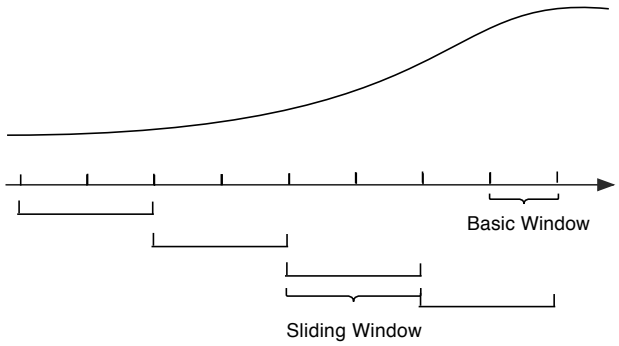


Figure 3.5: Basic Windows and Sliding Windows

## 3.4 Self Adaptation

As the attacker may change its attacking strategies at any time, to consistently deceive the attacker, the defender is required to keep tracking behaviors of the attacker. If the attacker changes its strategies while the defender still sends responses reflecting old strategy of the attacker, this abnormality of responses could lead the attacker to discovering the deception. Therefore, in order to successfully deceive the attacker in a long run, a mechanism of self adapting should be in place.

### 3.4.1 Problem Formulation

A defender largely depends on identifying changes of the attacker's attacking data flow to decide whether the attacker changes its attacking strategy. We propose a method of comparing data from different time periods to reveal possible changes in attacking strategies.

To begin formulating the problem, we define three time periods (Fig. 3.5) from smallest to largest:

- Time point - the smallest unit of time over which the system collects data, e.g., ten seconds.

- Basic window - a consecutive subsequence of time points over which the system maintains a digest (i.e., a compressed representation) e.g., two minutes.

- Sliding window - a user-defined consecutive subsequence of basic windows over which the user wants statistics, e.g., an hour.

Sizes of time points and basic windows are fixed in the same system, but sliding windows may have different sizes.

Objectives are the results a defender tries to achieve by deceiving the attacker. The objectives differ from application to application. They could be keeping the attacker's attacking strategy unchanged, keeping the attacker in place, or deterring the attacker away, and so on. From this perspective, objectives are also considered as the defender's expectation about inputs of the attacker. Previous experiences, be it online learning or offline experiment results, guide the selection of objectives. In DoS attack, our expectation is that the attacker will keep the attacking rate steady and not change its location. It further suggests that a steady rate of attacker's input will be the objective. We name the objective as $V_{obj}$. In case of deceptions on DoS attacks, the request/response exchange becomes the objective when the rate stablizes.

We model the attacking data flow as a non-ending time series $N_t = \{n_t, n_{t+1}, \ldots, n_{t+n}\}$ where $t$ denotes time points. Individual value of $n_t$ represents a digest of values in the time point. Uppercase $W_i$s and lowercase $w_k$s denote sliding windows and basic windows respectively. A precise definition of the problem is then formed as, *if* $Diff(W_i, V_{obj})$ *constantly exceeds a tolerance value* $t$*, the data flow* $N_t$ *is considered to be changed.* Problems remained are 1) how the difference is decided; 2) how frequent the tolerance value is exceeded considered as 'constantly'.

### 3.4.2 Difference Calculation

In general researches on time series, a lot of work [28][29] has been done on finding repeated patterns which are found in known datasets from unknown datasets. To index, classify, and find repeated patterns, a metric of similarity has to be defined first. The pattern can be in terms of moving average, time warping, or motif [30].

Such metric may seem overkill for our scenario since approaches proposed in above-mentioned literatures focus on types of highly variant time series, e.g. stock prices, or focus on processing very large set of time series. Since we try to find the deviations of the attacker's input from previous input, the problem is much easier than finding similar patterns. Hence a simpler yet more efficient approach is more relevant. We use a simple statistic approach to identify significant changes in the attacker's input.

Due to the fact that the input data we trace are rates of the attacker's input in time points, we compare harmonic means of two sliding windows to determine the difference. The objective value is considered as a standard, and the values in upcoming sliding window $W_i$ compares against it.

Harmonic mean is appropriate for situations when the average of rates are desired. Let $k = |W|$ be the number of time points in a sliding window $W$, and let $n_{i,j}$ denote the value of $j$th time point in sliding window $W_i$, harmonic mean $HMean$ of sliding window $W_i$ is given as:

$$HMean(W_i) = \frac{k}{\sum_{j=1}^{k} \frac{1}{n_{i,j}}} \tag{3.7}$$

When the deception begins, the steady rate of input becomes the objective $V_{obj}$. Let $b$ be the number of basic windows in a sliding window, a normalized difference is given as:

$$\delta = |\frac{HMean(\{w_{n+b+1}, \ldots, w_{n+2b}\}) - V_{obj}}{V_{obj}}| \tag{3.8}$$

The normalized differences give us an intuitive feeling about the movement of the attack. The larger the difference, the larger the movement of the attack. For a human, the trend is easy to be identified. However, an algorithm is needed for machines. An eligible algorithm should demonstrate following properties: 1) ignoring temporary surges in the input; 2) identify the change of paradigm in a timely fashion.

We define a term $\Delta$ as the measurement of deviation from the objective. $\Delta$ is a

non-negative real number initialized to be 0. $\Delta$ increases when a sliding window is evaluated and a difference is reported in. Increments of $\Delta$ are governed by a ceiling function $ceil()$ in which $inc = \delta$ when $\delta < c$ and $inc = c$ when $\delta \geq c$ and $c$ is the ceiling value. The ceiling function helps us to eliminate the impact of sudden surges in inputs. $\Delta$ also decreases over time. For each time point, the value of $\Delta$ is calculated as:

$$\Delta_{t+1} = \delta + dec(\Delta_t) \tag{3.9}$$

It is required the function $dec(x)$ to demonstrate a property that the closer the $x$ to 0, the lower the decreasing rate. After experimenting and failing, we finally found a function which serves our purpose well. Equation 3.10 shows the decrement function. $e$ in equation 3.10 denotes the decreasing rate. The value of $e$ is less than but very close to 1 (e.g. 199/200). It doesn't change in the entire process but, however, it does demonstrate the property that we expect. Fig. 3.6 demonstrates the required property of the decrement function.

$$dec(\Delta_t) = (1 + \Delta_t)^e - 1 \tag{3.10}$$

The difference $\delta$ comes in at a rate of once per sliding window. We define a time series $D_t$ as the difference time series. In $D_t$, values of all time points are zeros except those values reported at the end of each sliding window. As a result, the cumulative measurement of deviation $\Delta$ is calculated:

$$\Delta_{t+1} = D_{t+1} + dec(\Delta_t) = D_{t+1} + (1 + \Delta_t)^e - 1 \tag{3.11}$$

A simple method is used to decide if the attacking data flow exceeds a tolerance value CONSTANTLY or not. We define the tolerance value as $T$, a test window $W_{test}$. For example, we have sliding windows of size $|W_i| = 60(sec)$, a testing window of size

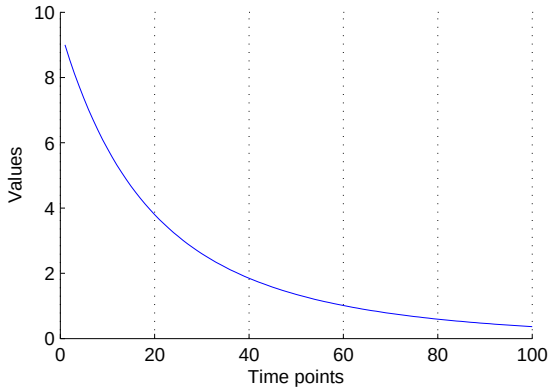


Figure 3.6: Property of Function dec()

$|W_{test}| = 300(sec)$. Five $\delta$ points will be reported in this test window. If, in this test window, the values of $\Delta$ are above $T$ for the majority of time, we can consider the paradigm of the attack changed.

Although selecting of the tolerance value $T$ and the length of test window $W_{test}$ is largely up to developers of specific applications, we show in Chapter 4 that our methodology could help application developers in finding optimal values of the parameters.

# CHAPTER 4

# RESULTS

## 4.1 Network Setup

Fig. 4.1(a) shows the normal operation of the network with data sent to the base station. If an attack is detected, nodes located close to the attacker or attack zone's edge act as sacrificial or deceiving nodes. The base station identifies the sacrificial nodes and informs them of their changed role in the network using the key management outlined above. Similarly, the base station informs the deceiving nodes of their new role. In Fig. 4.1(b), some of the neighboring nodes are sacrificial nodes. The other neighboring nodes re-route their packets as suggested in [12]. In Fig. 4.1(c), some of the neighboring nodes are deceiving nodes which transmit both normal and deceiving traffic. The neighboring nodes re-route their normal packets as suggested in [12] and send the deceiving traffic to the attacker. The sacrificial and deceiving nodes spoof their addresses so that the attacker would thinks it is getting packets from sensors that are more than one hop away and the sacrificial nodes also spoof the address of the attacker's neighboring nodes that are not sacrificial nodes.

## 4.2 Simulation

We model the collective behaviors of a sub-zone in our deception framework as a single entity. The entity represents a group of sensor nodes and one Distributed Deception Agent. Since we have done thorough analysis on behaviors of sacrificial nodes, we will not repeat the same subtle details of interactions among nodes and the DDA.

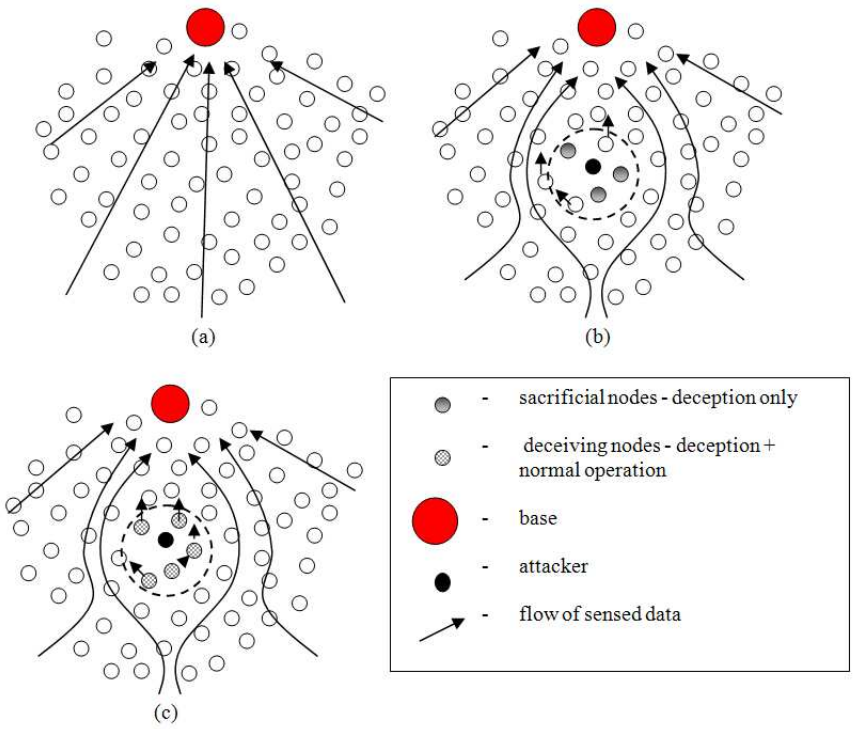


Figure 4.1: Deception Scenarios

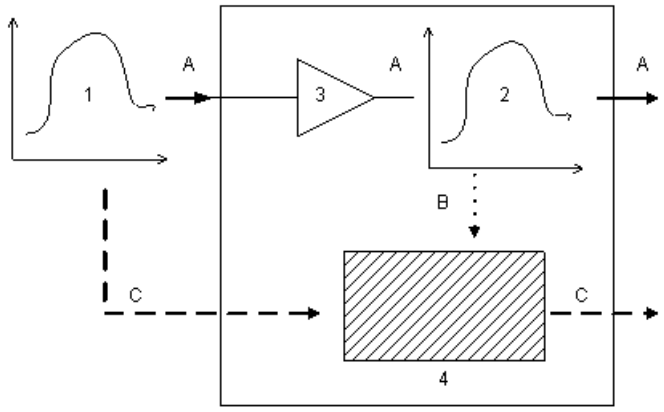


Figure 4.2: Deceiving Sequence

Rather, the sub-zone of interests is regarded as a black box, where only input and output are considered.

Fig. 4.2 shows the sequence of our deception activity. Components 1 and 2 are the request data and the response data respectively. Component 3 symbolizes the transformation of requests to responses in a physical network. Component 4 presents our algorithm in which the attack is learnt and the deceiving responses are generated. Three routes of data flows are labeled. Route A presents the data flow before deception occurs. Route C shows the deception data flow. Route B will only be used when the deception agent is learning patterns. Our simulation first simulates the activity represented by Route B. Using results from the first simulation, we then simulate the deception activity, or specifically the prediction aspect, which is what Route C represents.

The attacking traffic flow is simulated as infinite time series. Based on the above-mentioned assumption that given a stable status of responding, the translation function between requesting and responding is unchanged, we simplifies the translation function to $Y_t = X_t$ where $X_t$ represents the requesting time series and $Y_t$ is the responding time series. As long as the value of both time series don't reach their respective upper bounds which are decided by the underlying physical system, the translation function will be preserved in the entire course of simulation. The latency caused by asynchronized communications between nodes introduces a shift of value among time points. Thus the resulting time series exhibits a noise-like property (Fig. 4.3(b) and Fig. 4.4(b)).

Two types of attacking traffics are simulated. A constant rate attacking traffic flow represents a fixed strength of attack where the attacker finds an equilibrium. The other type of traffic is periodical, specifically a sine function against time points. Recall that the deception system mimics response time series. Due to the limitation of medium capacity, the rate of responses must reach to a point where the system can

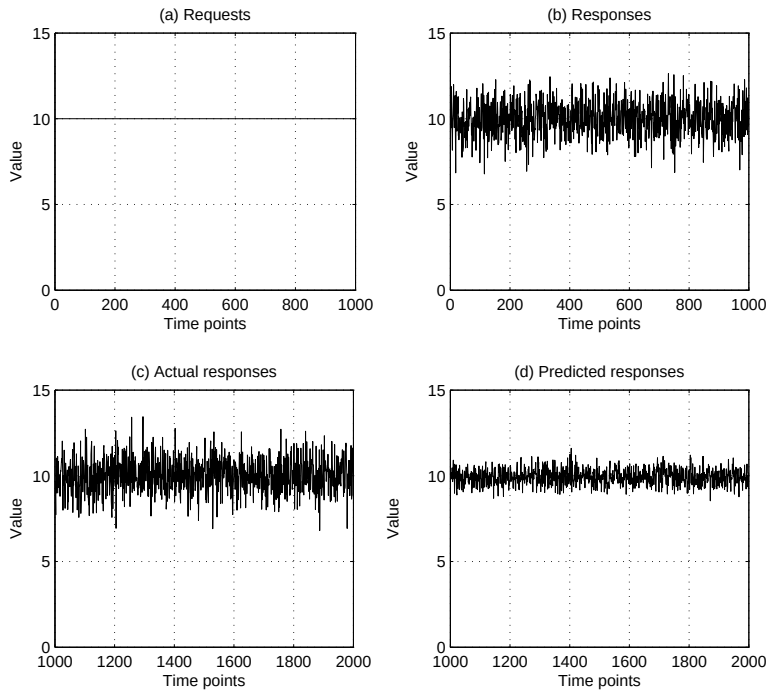


Figure 4.3: Constant Attacking Traffic Flow

not respond any more beyond that. Hence, we believe the constant and periodical attacking data suffice to show the effectiveness of our deception framework.

The simulation is written in C, Matlab, and glued up with Perl scripts.

## 4.3 Results of Deceptive Responses

Fig. 4.3 shows the result under constant rate attacking traffic flow. The attacking traffic flow is simply a time series of $X_t = 10$. The first 1000 time points of responses (Fig. 4.3(b)) are cached and used as a training set to train the deception algorithm. Fig. 4.3(c) shows the observed responses of size 1000 following the first 1000 time points. And Fig. 4.3(d) shows the predicted responses produced by the deception algorithm.

Fig. 4.4 shows the result of periodical attacking traffic flow. The equation used to generate the attacking traffic flow is $X_t = 10 + Sin(t/30)$. The deceiving process is

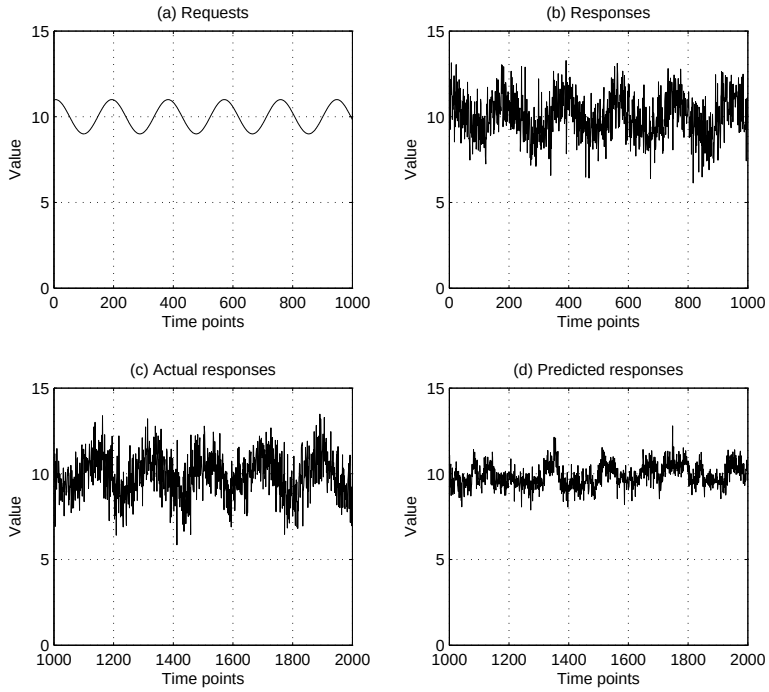


Figure 4.4: Sine Attacking Traffic Flow

similar to that of the constant rate attack.

Table 4.1: Quality of Deceiving Responses

| — | NRMSD | Mean (Observed) | Mean (Predicted) | Mean (Difference) |
|---|---|---|---|---|
| CONST | 0.1682 | 10.0272 | 9.9416 | 0.0856 |
| SINE | 0.1799 | 9.9342 | 9.8906 | 0.0436 |

As shown in Table 4.1, we calculate the Normalized Root Mean Square Deviation and Mean difference of observed sequence and predicted sequence. The mean difference gives a sense of how good the predicted sequence compares with the observed sequence quantitatively. The NRMSD then shows the quality of the prediction in terms of patterns of sequences. The difference of means in the range of sub 0.1 is impressive. Considering the large noise-like time shift behaviors of the time series, we believe both measurements yield impressive results for the two types of attacking flows.

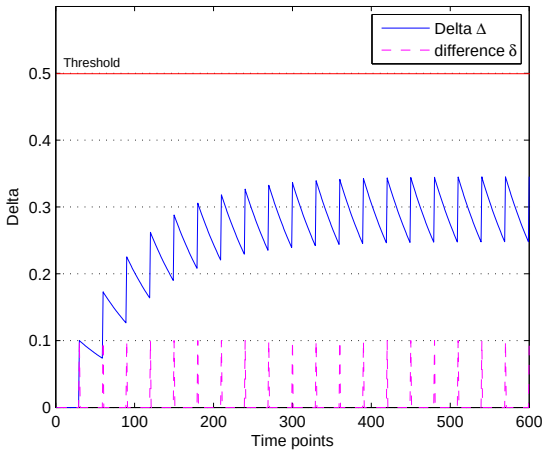


Figure 4.5: Attacking Flow with Normalized Difference of 0.1

## 4.4 Identifying Changes in Attacking Paradigms

In this section, we present and analyze results of simulations on identifying attacking paradigm changes. The primary goal is to show the effectiveness of our methodology. A synthetic attacking time series is used in the simulation. In each time series, 600 time points are presented. We assume the size of time points is one second, which means 10 minutes worth of data are included in each time series. The decreasing rate $e$ used in the simulation is 0.99. For the sake of simplicity, we assume the tolerance value $T = 0.5$. By our definition, if the values of $\Delta$ are higher than 0.5 for the majority of a test window, the attacking paradigm is deemed changed.

Fig. 4.5 shows the attacking flow which has constant normalized differences of 0.1 regarding the objective. An upper bound of 0.35 is shown in the figure. The result indicates that, for the tolerance of 0.5, an attacking flow with constant normalized differences of 0.1 will never exceed the tolerance. This behavior is desirable for our methodology because a low degree of errors (or noises) in the attacking flow is inevitable and expected. Fig. 4.5 also shows another interesting property of our algorithm, which is, for a given input flow, an upper bound of $\Delta$ for the flow can always be found. We call the property Constrained Upper Bound. This property helps

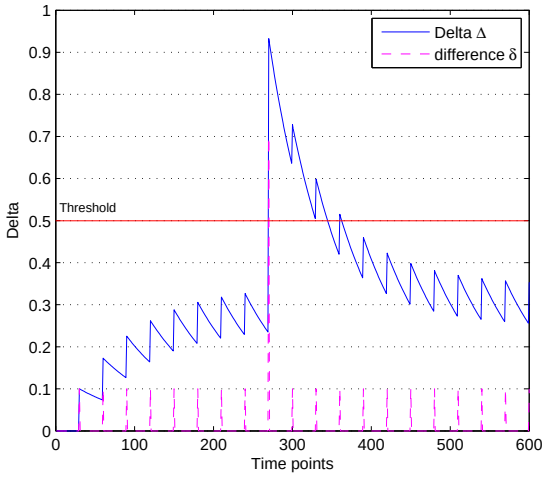


Figure 4.6: Attacking Flow with A Surge

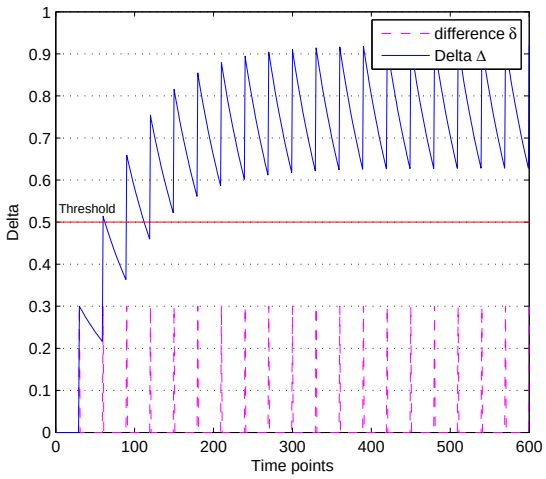


Figure 4.7: Attacking Flow With Normalized Difference of 0.3

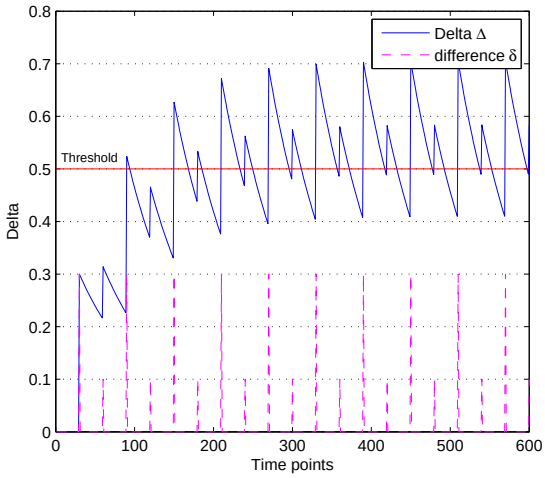


Figure 4.8: Fluctuating Attacking Flow

the developers of a specific application decide the optimal tolerance $T$. As far as an acceptable normalized difference $\delta$ is decided, the developer could use our algorithm to obtain the Constrained Upper Bound for the difference, which in turn becomes the lower bound of the tolerance $T$. Hence an optimal tolerance can be found.

Fig. 4.6 shows an attacking flow with a surge of normalized differences in one test window. It can be seen on the figure that the value of $\Delta$ declines rapidly after the surge happened, and the period of time in which the $\Delta$ value is higher than $T$ is brief. Based on this observation, the defender concludes the attacking paradigm is not changed.

For the attacking flow with constant normalized differences of 0.3 (Fig. 4.7), the $\Delta$ exceeds the tolerance 0.5 in second iteration, and stablizes at the upper bound of 0.9. If we use a test window of size 300 seconds, the $\Delta$ stays above the tolerance $T = 0.5$ for nearly 70% of the first test window. This result indicates that the change of paradigm is identified in 5 minutes. Similarly, an attacking flow with fluctuations is also simulated. For the test window starting from time points 151, in approximately 60% of the test window, the $\Delta$ is higher than the tolerance value $T$. By the definition of majority, we deem this result also indicates the attacking paradigm changed. It

takes 7.5 minutes to reach this conclusion.

According to the results of simulations, we believe our methodology identifies changes of attacking paradigms effectively. We also demonstrate the Constrained Upper Bound property of our algorithm helps developers obtain an optimal value of the tolerance $T$.

# CHAPTER 5

# CONCLUSIONS

In this thesis, a framework for deception in sensor networks is proposed. Deceptions can be used under certain scenarios such as when there is a need to attract an attacker to a less significant area of networks in order to protect high profile areas, or prevent the attacker from moving to a new location, and so on. A cluster-based semi-centralized architecture for efficient deception and communication is suggested. A method of choosing optimum layout of sacrificial nodes is also proposed. On top of the architecture, we propose a time series model of the network and a deception algorithm based on time series prediction. A method to identify changes in attackers' behaviors is also proposed. Simulation shows that our approach generates responses very closed to actual responses produced by the network.

Our work is built on top of [23] in which we developed a simple and efficient algorithm for deceiving DoS attacks. Our contributions in this thesis are two fold. First, we solved the outstanding issues in [23] by proposing an extensible framework of deception for sensor network. A time series based network model is also proposed. However, improvements will always be needed. We only have mimicking based deceptions implemented. Another deception, e.g. decoying, could be implemented on top of our framework to show the framework's universal applicability.

## BIBLIOGRAPHY


[1] G. Stein, "Encyclopedia of hoaxes," tech. rep., Gale Research, Inc, 1993.

[2] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON, 2004*, 2004.

[3] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: securing sensor networks with public key technology," in *Proceedings of the 2nd ACM workshop on Security of Ad hoc and Sensor Networks (SASN '04)*, (New York, NY, USA), pp. 59–64, ACM Press, 2004.

[4] B. Schneier, *Applied Cryptography*. John Wiley and Sons, 2nd ed., 1996.

[5] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 41–47, ACM Press, 2002.

[6] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 42–51, ACM, 2003.

[7] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 1, pp. 41–77, 2005.

[8] S. Zhu, S. Setia, and S. Jajodia, "Leap+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 4, pp. 500–528, 2006.

[9] H. Chan and A. Perrig, "Pike: peer intermediaries for key establishment in sensor networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1, pp. 524–535 vol. 1, 13-17 March 2005.

[10] R. D. Pietro, L. V. Mancini, Y. W. Law, S. Etalle, and P. Havinga, "Lkhw: a directed diffusion-based secure multicast scheme for wireless sensor networks," *Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on*, pp. 397–406, 6-9 Oct. 2003.

[11] L. Lazos and R. Poovendran, "Secure broadcast in energy-aware wireless sensor networks," in *IEEE International Symposium on Advances in Wireless Communications (ISWC'02)*, 2002.

[12] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[13] T. H. Project, *Know Your Enemy*. Boston: Addison-Wesley, 2002.

[14] N. C. Rowe, "Designing good deceptions in defense of information systems," *20 Annual Computer Security Applications Conference*, 2004.

[15] N. C. Rowe, "A model of deception during cyber-attacks on information systems," *Multi-Agent Security and Survivability, 2004 IEEE First Symposium on*, 2004.

[16] F. Cohen, "A mathematical structure of simple defensive network deceptions," *Computers and Security*, vol. 19, no. 6, pp. 520–528, 2000. 9.

[17] M. Priestley, *Spectral Analysis and Time Series*. London: Academic Press, 1981.

[18] C. Chatfield, *The Analysis of Time Series*. London: Chapman and Hall, 1989.

[19] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences*. Phd thesis, Harvard University, Cambridge, MA, 1974.

[20] P. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neur. Net*, vol. 1, pp. 339–356, 1988.

[21] A. M. Fraser and A. Dimitriadis, "Forecasting probability densities by using hidden markov models with mixed states," *Time Series Prediction: Forecasting the Future and Understanding the Past*, 1993.

[22] X. Ge and P. Smyth, "Deformable markov model templates for time-series pattern matching," *Knowledge Discovery and Data Mining*, pp. 81–90, 2000.

[23] R. Zhang, J. P. Thomas, and V. M. Mulpuru, "Deception framework for sensor networks," in *3rd International Conference on Security and Privacy in Communication Networks (SecureComm 2007)*, 2007.

[24] T. Stathopoulos, T. McHenry, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks," tech. rep., Center for Embedded Networked Sensing, 2003.

[25] D. Chakrabarti and C. Faloutsos, "F4: large-scale automated forecasting using fractals," in *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, (New York, NY, USA), pp. 2–9, ACM, 2002.

[26] F. Takens, *Dynamical Systems and Turbulence; Lecture Notes in Mathematics*, vol. 898, ch. Detecting strange attractors in turbulence, pp. 366–381. Berlin:Springer-Verlag, 1981.

[27] T. Sauer, J. A. Yorke, and M. Casdagli, "Embedology," *J. Stat. Phys.*, vol. 65, no. 3/4, pp. 579–616, 1991.

[28] R. Cole, D. Shasha, and X. Zhao, "Fast window correlations over uncooperative time series," in *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, (New York, NY, USA), pp. 743–749, ACM Press, 2005.

[29] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 419–429, ACM Press, 1994.

[30] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic discovery of time series motifs," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 493–498, ACM, 2003.

[31] R. A. Kemmerer and G. Vigna, "Intrusion detection: A brief history and overview," *Security & Privacy*, 2002.

[32] S. Gerwehr, J. Rothenberg, and R. H. Anderson, "An arsenal of deceptions for infosec (ouo)," Tech. Rep. PM-1167-NSA, RAND National Defense Research Institute Project Memorandum, October 1999.

[33] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2/e ed., 2003.

[34] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Proceedings of the First IEEE, 2003 IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

VITA

Ruiyi Zhang

Candidate for the Degree of

Master of Science

Thesis: A DECEPTION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

Major Field: Computer Science

Biographical:

Personal Data: Born in Longchang, Sichuan, China on April 3rd, 1982.

Education:

Completed the requirements for the degree of Master of Science with a major in Computer Science, Oklahoma State University in December, 2007.
Received the B.S. degree from Sichuan University, Chengdu, Sichuan, China, 2004, in Electric Engineering

Experience:

Worked for NCS Pte. Ltd. Suzhou as a Software Engineer from Febrary 2004 to April 2005. Developed Unit Trust Investment module for iBanking internet service of Development Bank of Singapore.
Co-founded 5Dtech Inc., worked as the leading Software Engineer of 5Dtech Inc from May 2001 to June 2003. Developed Enrollment Management System for Sichuan Univerisity, the campus e-Commercial entity 5dcollege.com, and various other projects for local businesses.

Name: Ruiyi Zhang Date of Degree: May, 2008

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: A DECEPTION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

Pages in Study: 42 Candidate for the Degree of Master of Science

Major Field: Computer Science

Although a lot of work has been done in proposing secure mechanisms for sensor networks, very little work has been reported on how to respond to attacks that have infiltrated the network. In this thesis we suggest a response framework to attacks that have been detected. We look at one particular response in detail, namely deception. Our contributions in this thesis are twofold. 1) A time-series based deception framework is proposed for learning attacker's behaviors and deceiving the attacker. A mathematical generalization of the framework is also proposed. We simulate a denial of service attack and simulation results show that, to the attacker, genuine response data and the fake response data generated by our algorithm are highly indistinguishable. 2) A harmonic means mechanism is used to determine if the attacker has changed his behavior. Simulations are provided to show that our approach not only detects slight changes over time, but also does filter out abnormality in data samples to avoid false alarms.

ADVISOR'S APPROVAL: _____