DECEPTION IN WIRELESS SENSOR NETWORKS - PREDICTING INTRUDER BEHAVIOR

By

ATUL RAVINDRAN

Bachelor of Technology in Computer Science

Cochin University of Science and Technology

Cochin, Kerala, India

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2009

DECEPTION IN WIRELESS SENSOR

NETWORKS - PREDICTING

INTRUDER BEHAVIOR

Thesis Approved:

Dr. Johnson Thomas

Thesis Adviser

Dr. Subhash Kak

Dr. Venkatesh Sarangan

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I thank my advisor Dr. Johnson Thomas for his advice, concern and encouragement regarding my research, and for his tremendous support in writing this thesis. I also thank my friends who have helped me achieve all the milestones that I’ve crossed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Wireless Sensor Network Security

Wireless sensor networks have revolutionized many fields in the business and physical world, from monitoring and conservation, manufacturing and asset tracking, automation of fields such as transportation and healthcare to military usages such as data collection in hostile environments.

Mastering a sensor network setup requires knowledge of signal processing, network protocols, embedded systems, distributed algorithms and information management [1]. These sensors have very limited computational, storage and communications resources. Hence implementing complex cryptographic protocols for security is difficult if not impossible due to the resource constraints. This makes such networks vulnerable to malicious attacks. Attacks on sensor networks deployed on unmanned networks which are tied to real world systems, could easily cripple the physical system associated with it [2]. It becomes important to handle such instances and usages to provide a basic security framework to resist an attack or malicious usage. Security in wireless sensor networks thus becomes very critical.

1.2 Deception in Wireless Sensor Networks

Once an intruder has been detected, there are many ways to deal with the intruder. The intruder may be removed from the network. However sometimes it may not be possible to eject the intruder from the network. For example, the intruder may be too powerful. One may not even want to remove the intruder under some circumstances. For example, before removing an intruder, the location of the intruder has to be traced. Another possible reason for containing the intruder is to study its moves and patterns to analyze the ultimate motive of the intruder, for future references and enhanced protection schemes. Deception is one possible response in such circumstances. If the attacker is too powerful or time is needed to trace back, deception may be employed.

To deceive an attacker, the good node must provide the attacker with the data he expects at the right time, with the right contents and at the correct rate. This can be achieved by pre-processing his moves and matching it up with the actual response from the node under suspicion. In other words, the success of deception can be determined only by monitoring the attacker. This requires predicting attacker behavior and if the actual behavior is as predicted, the deception is deemed to be successful. In such a case a time series prediction algorithm may be used to predict the behavior of the attacker. Here we propose a fast, memory efficient method which is reasonably accurate to predict the moves of an intruder. This method emphasizes on speed and simplicity of algorithm over accuracy, as the final destination of deployment is in a sensor network setting, which has limited memory and computational capability. Speed is also an important factor as the window frame which we get to predict and analyze is volatile and thus the results have to be obtained and analyzed quickly.

1.3 Role of Prediction Component in Deception Framework

The prediction component is not used as a primary component in the deception system, but instead, is used to check the validity of the deception system. The deception system is used in cases as described above, for example, when the attacker is too powerful to be flushed out of the system. In such cases, the deception system is employed and the attacker's behavior is monitored over time. The prediction unit is used to predict the behavior of the attacker based on previous data. The values obtained by the prediction component can be compared against the real values obtained from the attacker node to decide whether the deception is working or not.

Another, possible use of the prediction unit is to predict the point in time at which the attacker would transmit data. This information can be used to activate the sacrificial node (the node which carries out the deception) to respond to the attacker only at that point, when the prediction unit has predicted a signal from the attacker. It is useful, because the sacrificial node can be used to perform normal functions when it is not performing the role of the sacrificial node. This increases the number of nodes available for regular operations.

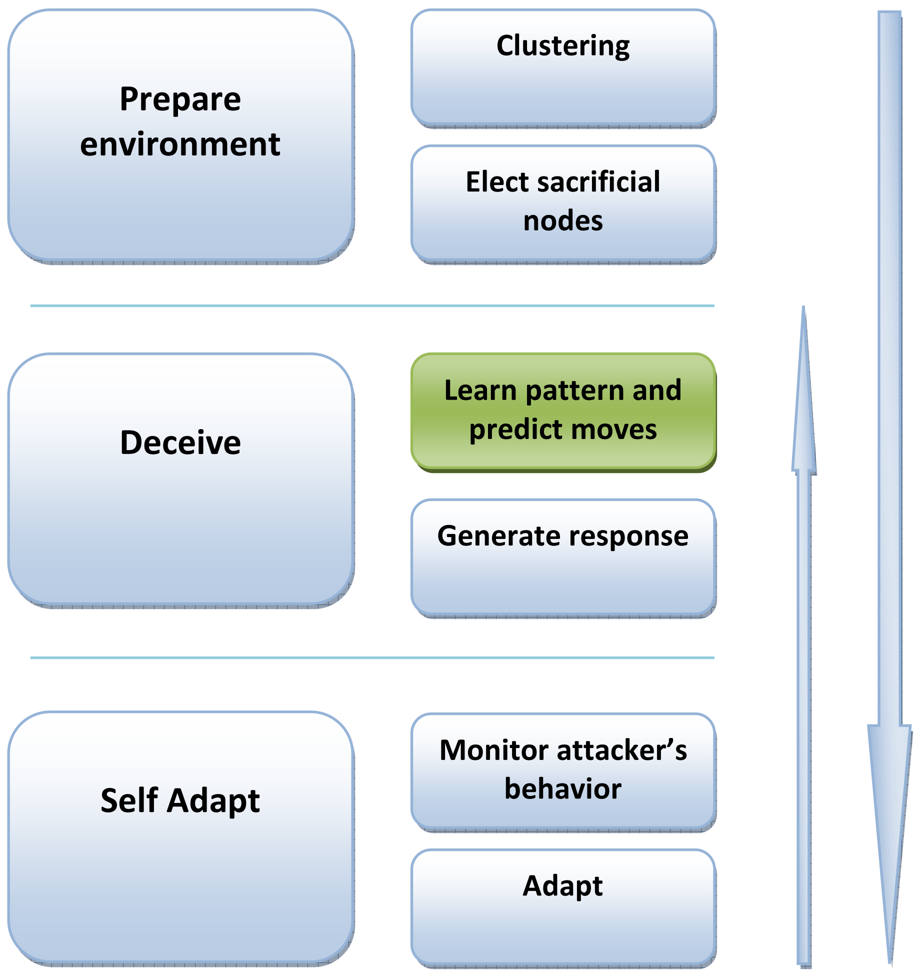


Fig. 1. Placement of prediction component in the deception

1.4 Issues in WSNs

Wireless sensor networks transmit data between nodes at a high rate and this makes the problem of prediction a hard one. Thus the deception of a malignant node, which is controlled by an intruder, has to be performed at a quick rate. The window frame obtained is small and the calculation of the next possible value sent by the intruder has to be performed within this frame. This problem thus seeks a method to quickly find values which only needs to be reasonably accurate to identify the intruder pattern. Moreover, the other problems in a wireless sensor network setting are the limited computational capability and the small memory system. Working on such a system narrows our range of prediction algorithms that can be applied, as most existing algorithms are memory intensive and computationally expensive and are therefore not suitable in a sensor network setting.

The prediction algorithm that we have implemented works by trying to predict data based on the elements towards the end of a time series prediction. It scans a limited number of data points into memory and searches for patterns in the scanned data set to make a future prediction. If it is not able to find an exact pattern, it initiates other components of the algorithm, which scans for others trends in data. The others trends here means any common pattern such as increasing or decreasing trend or a repeatedly occurring element. The algorithm aligns itself with the original time-series by rescanning and repeating the steps mentioned above.

This thesis also includes tests conducted using data obtained from a sink-hole attack and also tests performed to determine parameters used within the algorithm. It was validated by comparing the results obtained by the prediction logic to the original data. Predictions were made at points already obtained from the sink-hole data, so in this case the bench mark was the sink-hole data obtained from the WSNs.

1.5 Class of Attacks

Our algorithm is designed to be used at a point after the attacker has been identified. Prediction can be used to validate the functionality of the deception system deployed and to change the strategy if necessary. Our strategy would work on Selective-forwarding, Sink-hole and Worm-hole attacks. This method would not be successful in aiding resistance to Sybil attacks. In all the classes of attacks for which this strategy is applicable, the attacker's movements can be predicted. This can be compared with the moves he has performed, after the deception system has been engaged. Comparison would allow us to verify whether the attacker has been contained to sacrificial nodes or not. The method would not work on Sybil attack, as the attacker changes the address and appears as different nodes. This would not help us validate the scheme as the address of the real attacker is not available to validate against the historic data.

Selective forwarding is a technique of forwarding only certain routing messages and removing the others from the system. Selective forwarding maybe part of a sink-hole attack, which essentially is a technique of advertising a high quality link so as to attract data traffic into the compromised node. Worm-hole attacks refers to the activity of replaying messages read at one location at another location which is generally a point located at a distance from the original node. In worm-hole attacks, the attacker uses a different channel o transport the message intercepted. Hello-flooding is the attack strategy of advertising a node as a high-link quality neighbor. This may be used to drive all the traffic to a single node and thus making the network collapse, as a result of the high load on a single channel [13].

CHAPTER II

REVIEW OF LITERATURE

Forecasting using time series dataset has been a topic that has seen a lot of research interest, with highly successful techniques for prediction. Since it has seen a lot of research there exist a lot of techniques which predict values with a high rate of accuracy, both for linear and chaotic datasets. At the beginning interest was mainly focused on linear prediction and since then it has been extended to chaotic datasets. There exist techniques which predict the data on chaotic datasets without any manual intervention [3]. These techniques can be used in most cases, since the accuracy rate is very high and are reasonably fast. However, the memory requirements are very high and fast computational resources are needed.

2.1 Existing Forecasting Techniques

Sensor networks are a highly researched topic of interest ([4], [5], [6]). There are a lot of techniques – ARMA, ARIMA, Fractal FOREcasting, neural network based systems etc., which has been proposed already. Some are either too simplistic - producing only linear results, which are hardly of any use from a practical standpoint, while others require a lot of training time for data analysis and result generation [3]. Neural network based prediction algorithms which can be instantaneously trained are available for general purpose prediction schemes [14]. These neural networks based prediction algorithms require space complexity is larger than what a simple WSN node can handle. In our deception scheme, time is very limited and these approaches are therefore not useful. There are techniques however which efficiently scans the dataset, do not require large training times and which operate in a black-box setting, like Fractal FOREcasting [3]. All of them are made for a general purpose and not for any specialized domain. The Fractal FOREcasting [3] requires a good computational processor in order to quickly calculate the values required as parameters for the Delay coordinate embedding technique used in the black-box. For calculating the parameters a powerful processor is needed, unlike the wireless sensor network nodes deployed in WSNs.

The method that we are following is different from other’s research, because we are designing the algorithm with the sole purpose being its usage in a wireless sensor network, for deceiving an attacker. Deception is subjective to some extent and therefore we can afford some inaccuracy while gaining on speed, while working in a highly resource constrained environment. This technique could also be used in a similar environment where the conditions are comparable. The main difference between our work and previous works in the field is with regard to the above mentioned condition which is the targeted work environment, i.e. wireless sensor networks. This gives us the flexibility of working on an algorithm which quickly scans the data available and computes a result which might not be very accurate, but is acceptable for our purposes. This allows us to work on fast computation, low memory usage and simple prediction formats. Since the kind of data produced by the wireless sensor devices fall between a certain data format frame, we can operate within this frame, rather than using complex algorithms used for predicting financial data and such.

CHAPTER III

METHODOLOGY

3.1 General Outline of Algorithm

Wireless sensor networks are extensively used in environments where human intervention and data transportation bandwidth is limited. Also, these systems do not have enough memory and computational resources to perform highly complex calculations.
Our system aims to predict an intruder’s pattern, using the limited computational power and memory, within a reasonable level of accuracy.

The proposed approach is a trade-off between speed and accuracy. Our algorithm is designed to compute the value of the next time series elements, using simple calculations and analysis. The proposed solution involves working on a fairly inaccurate set and deciding on the moves a particular node would employ. This is basically done by analyzing the data for a particular pattern and then, forecasting a particular node’s values before it actually starts transmitting those.

The algorithm proposed here can mainly be divided into four components, that works together to achieve the time series prediction. The main components of the prediction system can be elaborated as follows.

- Input trimmer
- Time series prediction part
- Spike handler / band pass filter
- Adaptable scheme – pattern detector

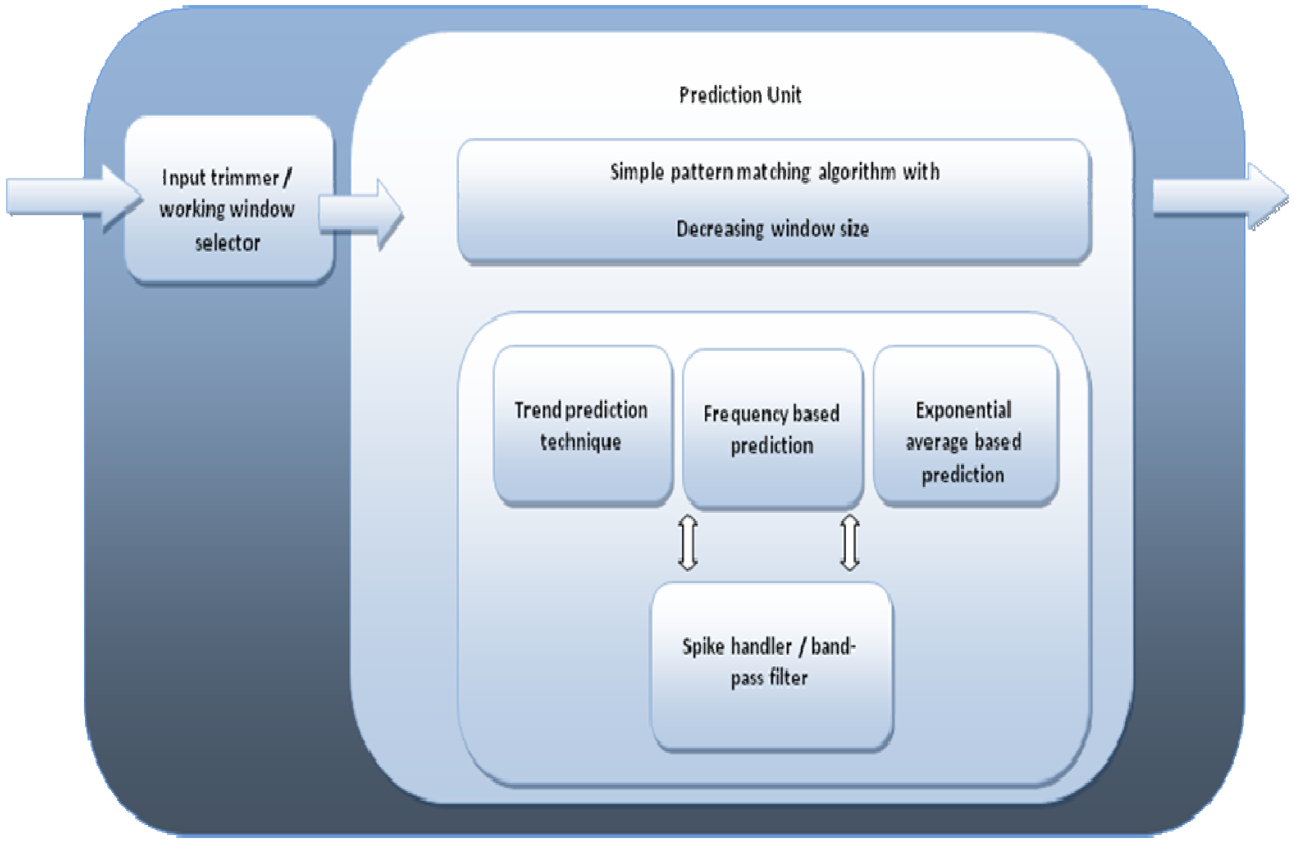

Input trimmer / working window selector
Prediction Unit
Simple pattern matching algorithm with Decreasing window size
Trend prediction technique
Frequency based prediction
Exponential average based prediction
Spike handler / band-pass filter

Fig 2. Structure of various components of the algorithm.

3.2 Input Selection / Working Window

Input selection strategy is an essential pre-processing stage in order to achieve high accuracy in the predictions. It is applicable to many domains like pattern matching, time series prediction, econometrics etc. Problems that can occur due to poor input selection are

- "Curse of dimensionality" – if the input dimensionality is too large. This also increases the computational cost and memory requirements, which is very limited in the wireless sensor network implementation. Curse of dimensionality refers to the exponential growth of volume associated with adding extra dimensions to a mathematical space. In this context, it refers to the addition of extra historic data which will not be useful for the purpose of future predictions.

- Understanding complex models takes more time than simple ones.

Input selection strategy can be grouped, broadly under Directed selection strategy or Recursive selection strategy. Direct selection is more accurate, but would require much higher computation, where-as recursive selection which is not as accurate can be computed much faster than the direct selection method [10].

There are different input selection strategies available for time series prediction, such as Mutual Information and Non parametric noise estimator [7]. There are also strategies which use neural networks to select the inputs that contribute mostly towards the accuracy in prediction [11]. Input selection has been done by performing different tests on the algorithm in order to obtain the optimum value for the prediction accuracy without increasing the computation load. The input selection, essentially selects the *n* number of elements towards the final value of the point to be predicted. This is based upon the fact proximity plays an important role in the characteristics of values.

3.3 Time Series Prediction

This is the most important part of the algorithm. It takes input from the “Input selection strategy” part and uses it for predicting, with a fair amount of accuracy. It then checks for any matching pattern and predicts the values using the short-term prediction component. It tries to obtain a directional trend and then searches for the directional pattern in the time series dataset. If it finds a match, it applies the scaling factor and predicts data points. If it is not able to find any matching pattern, it applies either a trend prediction or frequency based prediction- by utilizing the window dataset data and direction, it decides whether to execute the trend based predictor or frequency checker component. In this step, if the algorithm detects any spikes in the data set, it uses the band-pass component to filter the spikes and then repeats the prediction again. The prediction based on a simple pattern matching tool [8] to minimize the computation and thereby, save energy. Our aim is to use as little resources as possible, so as to conserve power.

3.4 Spike Handler / Band Pass Filter

This function is called by the Time series prediction component to clear the data of any spikes before processing a trend / auto correlation based prediction. This section deals with the problem of filtering time series, so that some values are removed and some are retained. There are different kinds of filters, low-pass, high-pass or combined band-pass filters [9].

There are different approaches for achieving filtration, such as

- Fourier Methods
- Centered, Non-recursive weighting methods
- Response function

Fourier method uses Fourier transformations to carry out the filtration. This has the implicit problem of losing the correct data structure after transformation, especially towards the end of the time series [10]. After evaluating the different approaches, considering the minimum computation desired in case of wireless sensor networks, the centered non-recursive weighting method has been incorporated into our algorithm for providing band pass filtering.

3.5 Filtering Algorithm

Steps in the algorithm are as follows:

- It starts by trimming the input based on the previously determined range and uses only this data as the historic data, till the next look-back is performed. Trimming the data is extracting the final *n* number of points as specified in the algorithm. The input length is set at 100 elements, by performing tests and is referred as previously determined range. The algorithm scanning of real data from the time series into the input array is referred as look-back.

- Trimmed input and the working window are sent to the pattern detector, to create a directional pattern for the input and the working window. Here working-window is the final 30 elements of the current input data (100) being used for the purpose of prediction.
- If a directional pattern match is found between the working window and the historic data, the exact pattern matching component is initialized.
- If from the exact pattern matcher, an exact pattern is found then the value is predicted based on the value of the next element in the historical data and the result is scaled to match the final output. Exact pattern match works by looking for scaled differences between adjacent values. The scaling factor is required to increase or decrease the value obtained from the historical data, to the current level of data-points in the working window.
- If no directional pattern is found, it reduces the window size till 70% of current window size and repeats the directional pattern again. The working window is reduced by removing the elements inside the working window in a FIFO fashion. If it is still not able to find a pattern, then the next step is initialized.
- In-case an exact pattern is not found, the working window's direction pattern is checked for any trend pattern to predict the next value. Trend pattern refers to values which are either increasing or decreasing continuously. Since we have the directional pattern of the working window, it would either be a continuous 1 or 0

string. The absence of 1 or 0 can be checked and is used for determining whether the values of the working window follow an increasing or decreasing trend.

- If a trend pattern is found, it is again checked with the frequency checker
- If a single item is found to occupy more than 60% of the working window's array, a frequency based prediction is made. Frequency based prediction refers to the technique of determining the no. of occurrences of a single element in the working window, and predicting that the element is most likely to occur again.
- If the frequency checker gives a negative output, then the mean average of the values inside the working window is determined and added to the final element of the working window. This occurs if it has been previously determined that, the working window follows a trend based pattern and does not have a exact match in the historic data under review.
- If an exact and trend pattern is not discovered, it is again sent to the frequency checker
- If a the frequency checker gives a positive output, that element is used as the output for prediction
- Else, if the frequency checker also gives a negative output (here there is no exact/trend pattern), exponential averaging is used to arrive at the next value. Exponential averaging is an averaging technique which gives more weight to the final value. $\mu_{\text{Total}} = \alpha\mu_n + (1 - \alpha)\alpha\mu_{n-1} + (1 - \alpha)^2 \alpha\mu_{n-2} + (1 - \alpha)^3 \alpha\mu_{n-3} +$. High value of α - gives more weight to recent data and low value of α - more weight to old data. Here μ_n is the most recent data point and μ_{n-1} the previous one and so on. μ_{Total} is the point obtained by performing the exponential averaging.

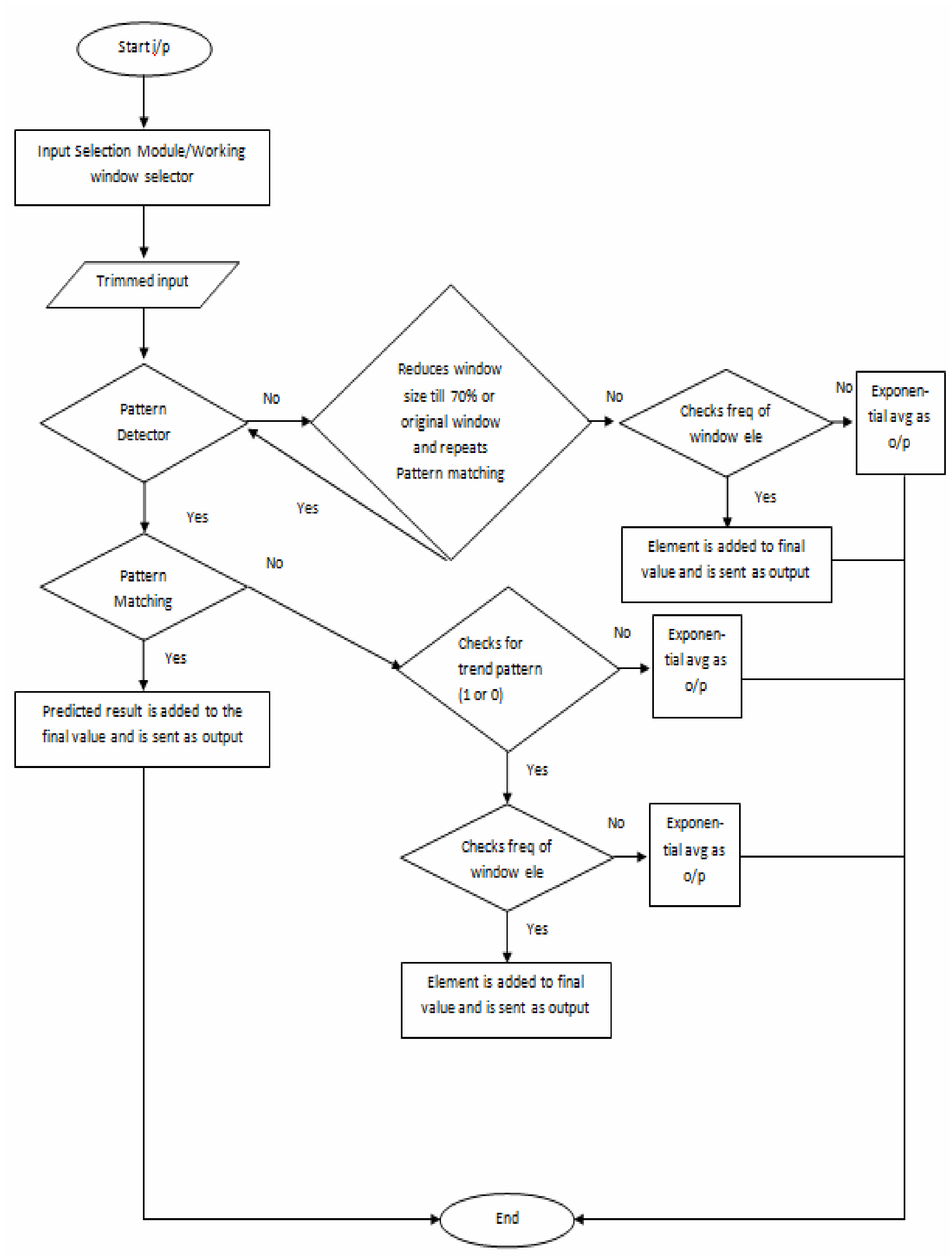

Start i/p
Input Selection Module/Working window selector
Trimmed input
Pattern Detector
No
Reduces window size till 70% or original window and repeats Pattern matching
No
Checks freq of window ele
No
Exponen-tial avg as o/p
Yes
Yes
Yes
Element is added to final value and is sent as output
Pattern Matching
No
Checks for trend pattern (1 or 0)
No
Exponen-tial avg as o/p
Yes
Predicted result is added to the final value and is sent as output
Yes
Checks freq of window ele
No
Exponen-tial avg as o/p
Yes
Element is added to final value and is sent as output
End

Fig 3. Flowchart of algorithm

CHAPTER IV

EXPERIMENTS AND FINDINGS

Experiments were conducted on data obtained from a sink-hole attack performed on a TinyOS platform. The attack experiment setting included a total of 20 sensor nodes, out of which 1 node was set as an attacker node, and the remaining 19 as normal sensor nodes. The TinyOS emulation was performed as 6 stages which were distinguished based on the attacker's sending pattern.

4.1 Sink-Hole Attack

In a sinkhole attack a compromised node tries to draw as much traffic as possible to itself. This is done by making itself look attractive compared to the surrounding nodes in terms of routing metrics. It typically does this by advertising that it is very close to the base station, when in fact it may not be [12]. Furthermore, the more the attacker advertises its routing information, the more the traffic it will attract.

The 6 different patterns included

- Linearly increasing frequency of signal transmission by the attacker

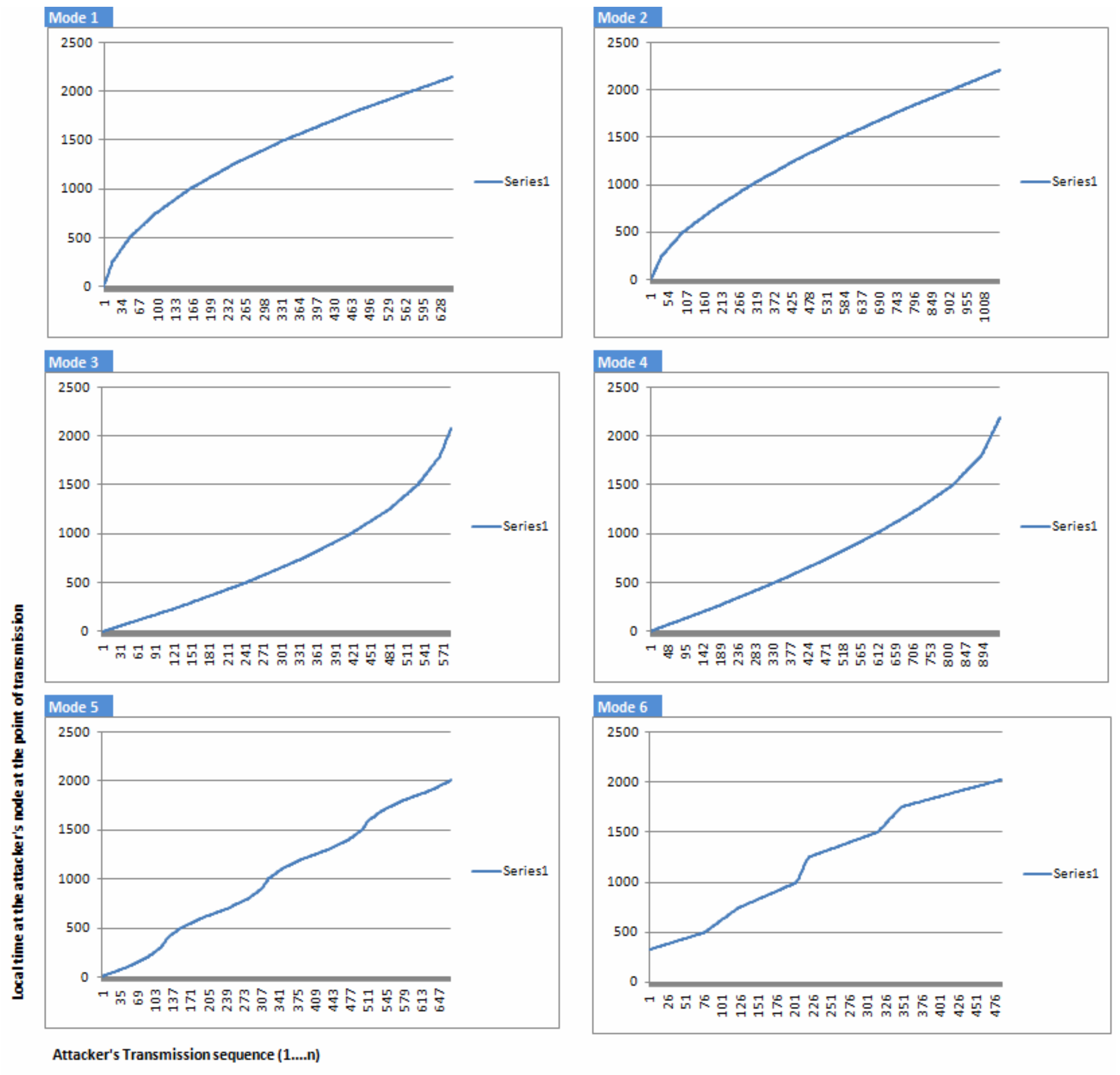
- Non-linearly increasing frequency of signal transmission
- Linearly decreasing frequency of signal transmission
- Non-linearly decreasing frequency of signal transmission
- Combined/Sine wave pattern based frequency of signal transmission
- Random pattern based frequency of signal transmission

Packet Format

- Data format: decimal format;
- Packet: [Message type : Source Node, Original Node, Receiving Node, Local Time of Source node]
- Examples, Sensing: [S: 15, 15, 1, 860] and Routing: [R: 1, 1, 65535, 861]

Explanation of message/nodes used

- S: Sensing Message
- R: Routing Message
- Source Node: The node that is sending the message.
- Origin Node: The node from whom the message originates.
- Receiving Node: The node to which the source node transmits the message.
- Local time of the source node: The system is not perfectly time-synchronized, so the local time is recorded instead.
- E.g. Node 2 sends sensing message to node 3. This is then forwarded by node 3 to Base station (address 0). The message that node 3 forwards to base station will be [S: 2, 3, 0, Local time of Node 3].



x-axis: attacker’s transmission sequence(1^{st}, 2^{nd}..n) , y-axis: message transmission time of attacker in seconds

Fig 4. Graphs plotted to show the overall structure of frequency variations.

Mode 1: Linearly increasing frequency of signal transmission by the attacker

Mode 2: Non-linearly increasing frequency of signal transmission

Mode 3: Linearly decreasing frequency of signal transmission

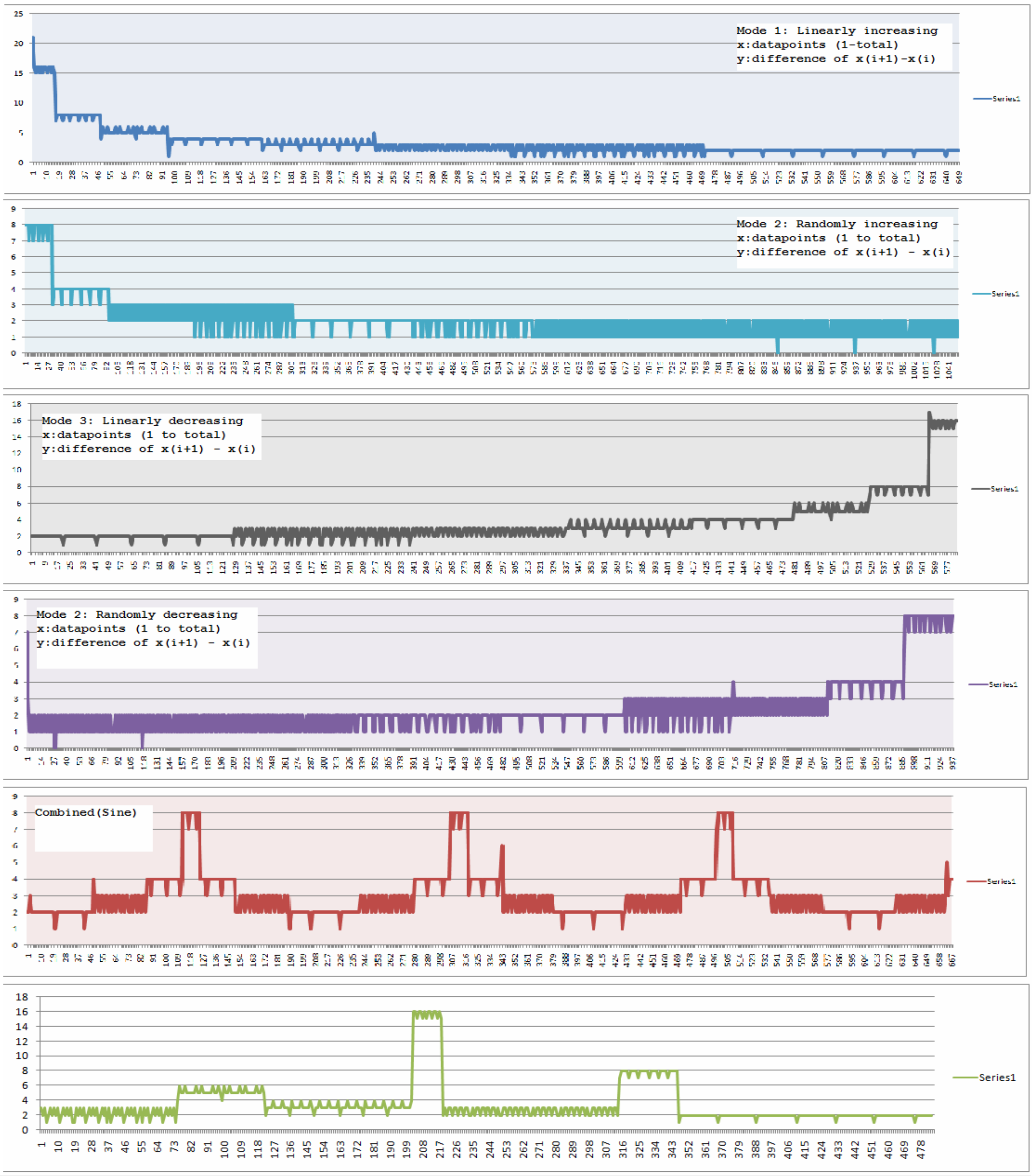
Mode 4: Non-linearly decreasing frequency of signal transmission

Mode 5: Combined/Sine wave pattern based frequency of signal transmission

Mode 6: Random signal transmission pattern

The data obtained from the experiments were further used to conduct tests to determine the optimum value of the input and the window size for use in the prediction system.

The prediction system analyses the internal structure of differences between the adjacent values to determine the overall pattern for use in prediction. The figure below shows the pattern of data obtained from the different sink-hole attacks performed.

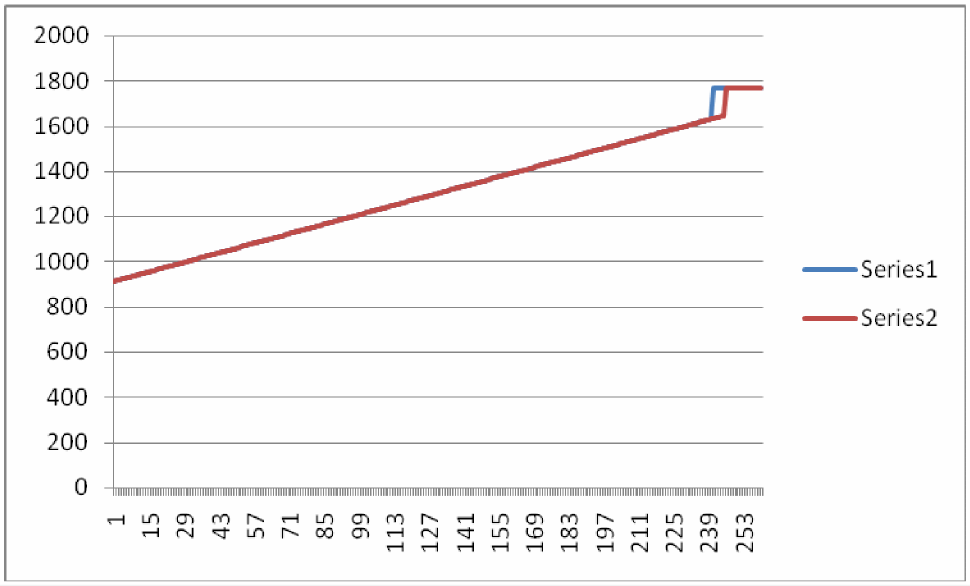


x-axis : Attacker's transmission sequence(1st, 2nd…n), y-axis : Difference between adjacent values(x_{i+1}-x_i)

Fig 5. Graphs plotted to show the internal structure of frequency variations using the differences between adjacent values.

4.2 Individual Tests

The individual tests show the working of individual components of the algorithm. These were performed to validate the different modules used in the algorithm before integration as a single system. Fig 6. shows how the algorithm detects change in a continuously increasing pattern and modifies the predicted value accordingly.

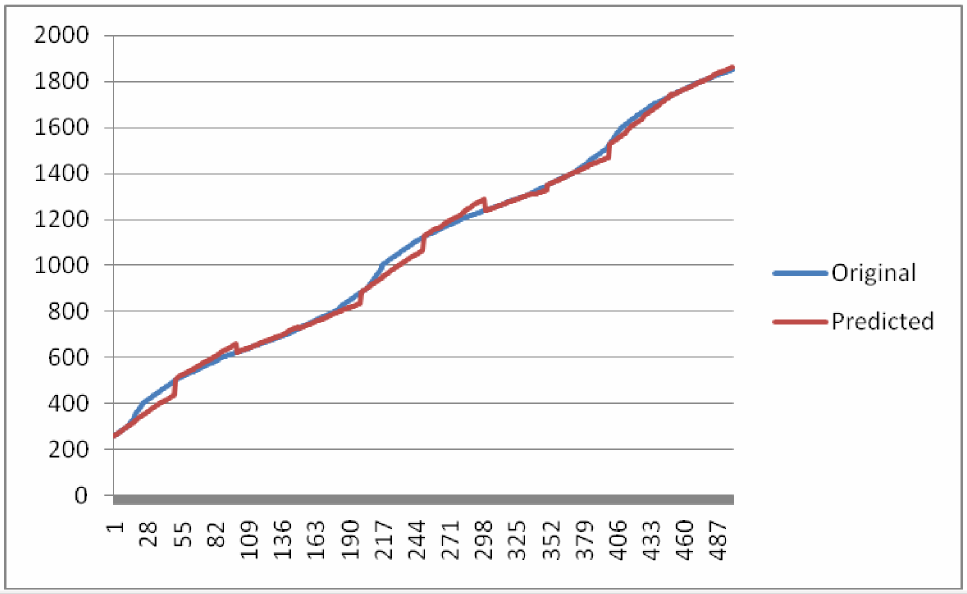


x-axis: attacker's transmission sequence(1st, 2nd..n) , y-axis: message transmission time of attacker in seconds

Fig. 6 Displays the algorithm tracking change and predicting values (trend based)

Fig 6. has been plotted using the data that has been generated for the purpose of displaying how the various components of the algorithm work. It shows that the alogithm has detected a change in the pattern of data (blue) and has modified the prediction accordingly(red).

Figure 7. shows the trend based prediction compoent at work with a single step-ahead prediction. It can be seen from the image, that the variations of the sine wave are picked after a small time interval and is adjusted in the predicted values. The predicted values increases first and then decreases according to the sine wave pattern in the original data. This fluctuation can be observed in 3 parts on the graph data.



x-axis: attacker's transmission sequence(1st, 2nd..n) , y-axis: message transmission time of attacker in seconds

Fig. 7 Displays the algorithm predicting based on trend pattern

4.3 Input and Window-size Length Determination

The runtime tests of the algorithm included a total of 45 with varying input lengths and window size (*n* final elements of the input points) tests which can be summarized as follows

- 15 tests using the window size as 10

- 15 tests using the window size 30
- 15 test using the window size as 50

Each of these 15 tests can further be classified as

- 5 tests each using the input size as 100
- 5 tests each using the input size as 200
- 5 tests each using the input size as 300

Here input size refers to the no. of elements scanned into memory from the historic data obtained from the sink-hole attack (attackers transmission time). Effectively, this becomes the historical data as far as the algorithm is concerned. The window size (most recent one) is scanned to derive the pattern in the window. The window is the later part of the input. The remainder of the input (that is, without the window) is scanned to determine if the pattern in the window is also present in the earlier remainder of the input.

5 points in time from the historic data (obtained from the sink-hole attack) were selected to compare the accuracy with the already available historic data for all the different tests. The 5 points that were chosen based on the criteria that they were at least after the 300th data point, so as to allow enough learning set for the tests which required 300 points to start with.

The points that were selected are as follows:

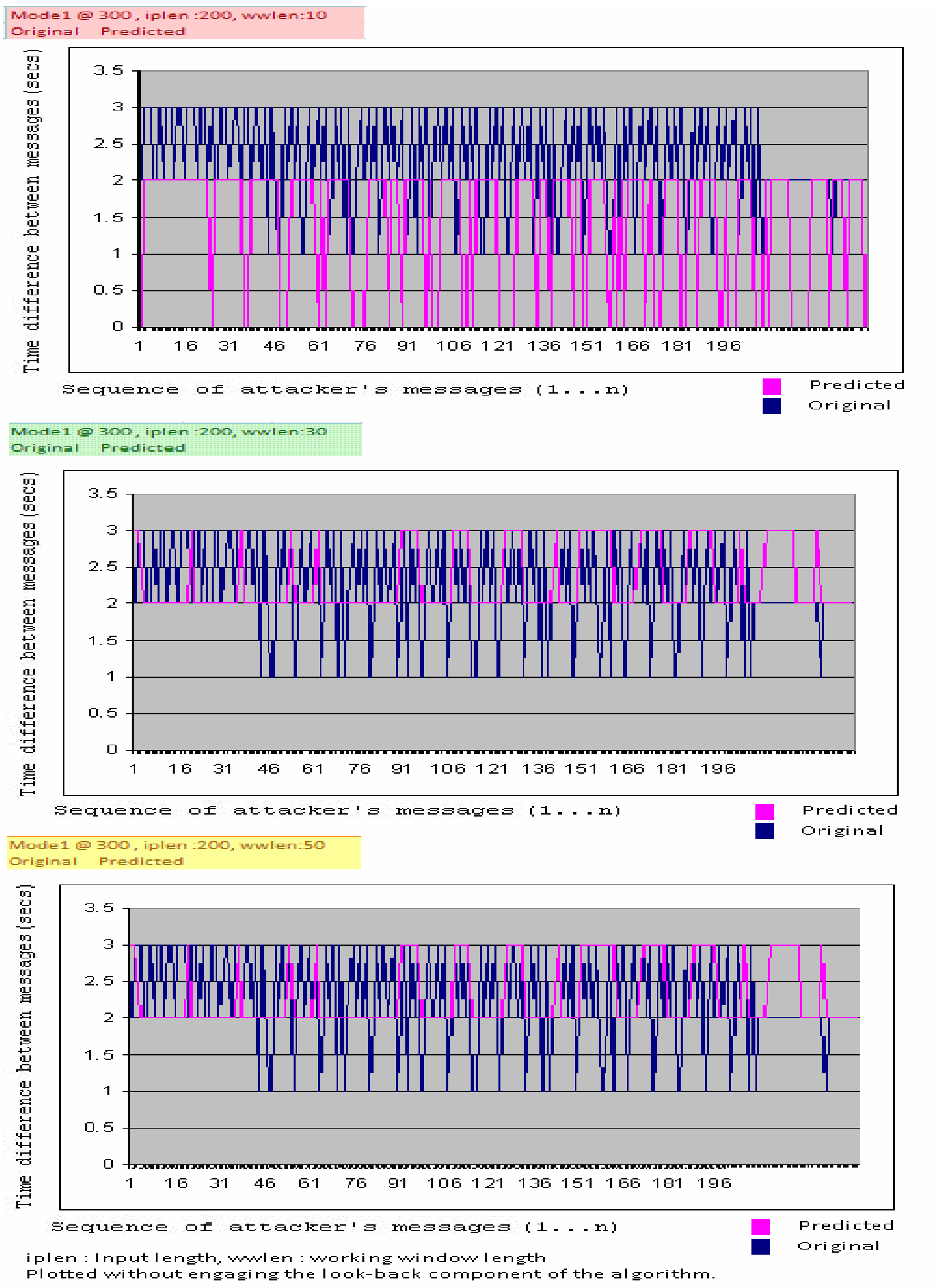
- Data from Mode 1, prediction starting @ 300th point in the time series data
- Data from Mode 5, prediction starting @ Final-100th point in the time series data

- Data from Mode 3, prediction starting @ 300th point in the time series data
- Data from Mode 4, prediction starting @ 500th point in the time series data
- Data from Mode 2, prediction starting @ Final-100th point in the time series data

Mode 6 was omitted as mode 6 is a random combination of mode 2 and mode 4. If the prediction started at the 300th point, and the input size is 100, then the data from the 200th to 300th point was used for prediction. 5 predicted points after the 300th point (if the prediction started at the 300th point) were compared to the actual or historical data. The 5 predicted points were from the 300th point till the end of the data sequence.

The objective of conducting the tests was to determine the best values for input length and the working window (which is the final n no. of points of the input length). These values determine how many historic data points from the time-series data-set is scanned into memory, for the purpose of predicting future values. The tests were conducted by running the algorithm for different combinations of input length and working window length. Graphs were plotted for each of the results obtained. The graphs compared the original and predicted values and displayed the variations between them.

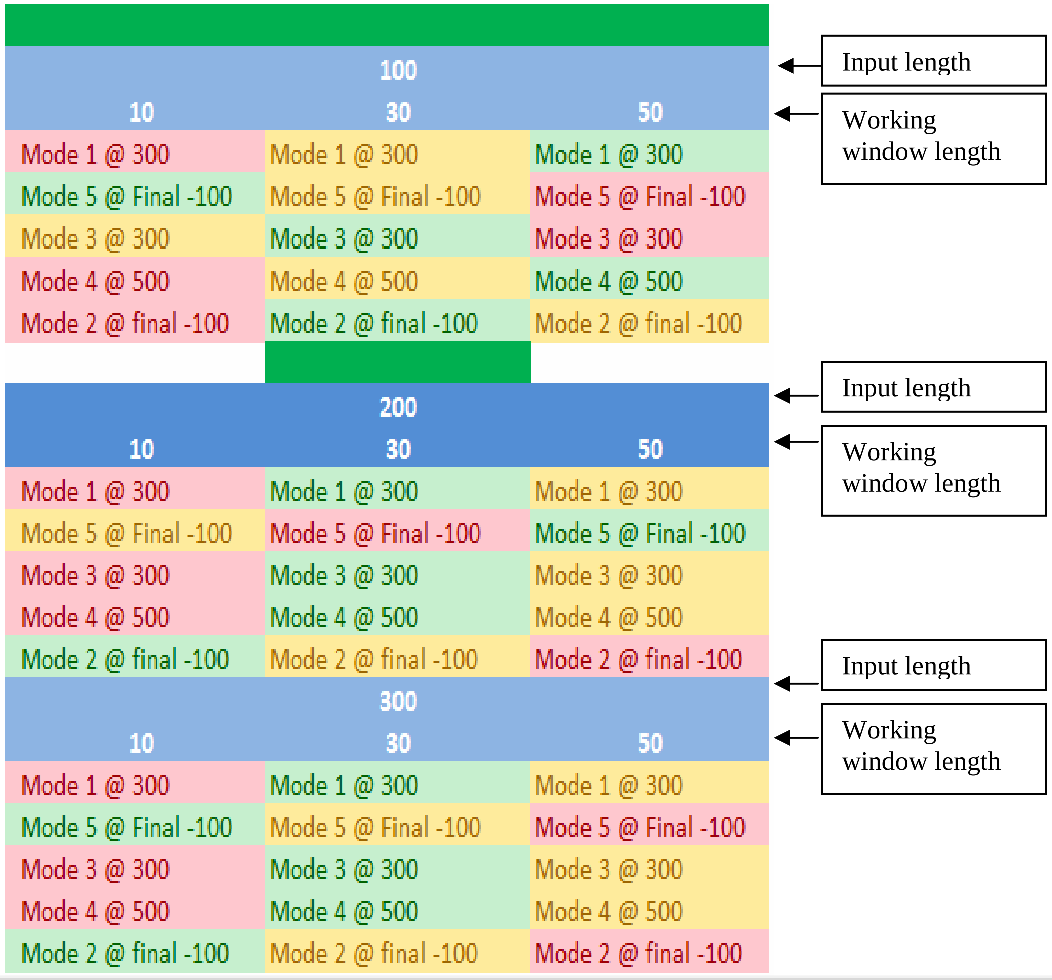
The algorithm consistently performed in the acceptable level when the input size was set at 100 and the window size at 30.



iplen : Input length, wwlen : working window length
Plotted without engaging the look-back component of the algorithm.

Fig 8. Graphs show the comparison of values obtained with different window sizes

The 45 tests performed were plotted on graphs to identify the best parameters to use, so as to obtain the optimum prediction results from the algorithm. The table below displays the results obtained. The table is categorized into 3 sub-tables based on the input length used as the input. It is color coded based on good (green), moderate (yellow) and bad (red) for each test. Graphs were plotted to check for the best match between the original and predicted values, for each of the tests conducted to label them as good, moderate or bad based on the mean % error calculated.



Input length		100	
Working window length	10	30	50
	🟥 Mode 1 @ 300	🟨 Mode 1 @ 300	🟩 Mode 1 @ 300
	🟩 Mode 5 @ Final -100	🟨 Mode 5 @ Final -100	🟥 Mode 5 @ Final -100
	🟨 Mode 3 @ 300	🟩 Mode 3 @ 300	🟥 Mode 3 @ 300
	🟥 Mode 4 @ 500	🟨 Mode 4 @ 500	🟩 Mode 4 @ 500
	🟥 Mode 2 @ final -100	🟩 Mode 2 @ final -100	🟨 Mode 2 @ final -100

Input length		200	
Working window length	10	30	50
	🟥 Mode 1 @ 300	🟩 Mode 1 @ 300	🟨 Mode 1 @ 300
	🟨 Mode 5 @ Final -100	🟥 Mode 5 @ Final -100	🟩 Mode 5 @ Final -100
	🟥 Mode 3 @ 300	🟩 Mode 3 @ 300	🟨 Mode 3 @ 300
	🟥 Mode 4 @ 500	🟩 Mode 4 @ 500	🟨 Mode 4 @ 500
	🟩 Mode 2 @ final -100	🟨 Mode 2 @ final -100	🟥 Mode 2 @ final -100

Input length		300	
Working window length	10	30	50
	🟥 Mode 1 @ 300	🟩 Mode 1 @ 300	🟨 Mode 1 @ 300
	🟩 Mode 5 @ Final -100	🟨 Mode 5 @ Final -100	🟥 Mode 5 @ Final -100
	🟥 Mode 3 @ 300	🟩 Mode 3 @ 300	🟨 Mode 3 @ 300
	🟥 Mode 4 @ 500	🟩 Mode 4 @ 500	🟨 Mode 4 @ 500
	🟩 Mode 2 @ final -100	🟨 Mode 2 @ final -100	🟥 Mode 2 @ final -100

Mode X @ Y : Data from Mode X, prediction starting @ Y point in the time series data

Table 1. Table comparing results obtained from different tests, based on varying window size and input length. [Green: Good, Yellow: Neutral, Red : Bad]

For input length 100, window size 10 gives 3 bad, 1 good and 1 moderate result.

For input length 100, window size 30 gives 3 moderate and 2 good results

For input length 100, window size 50 gives 2 good, 2 bad and 1 moderate results.

For input length 200, window size 10 gives 3 bad, 1 good and 1 moderate result.

For input length 200, window size 30 gives 3 good, 1 bad and 1 moderate result.

For input length 200, window size 50 gives 3 moderate, 1 good and 1 bad result.

For input length 300, window size 10 gives 3 bad and 2 good results.

For input length 300, window size 30 gives 3 good, and 2 moderate results.

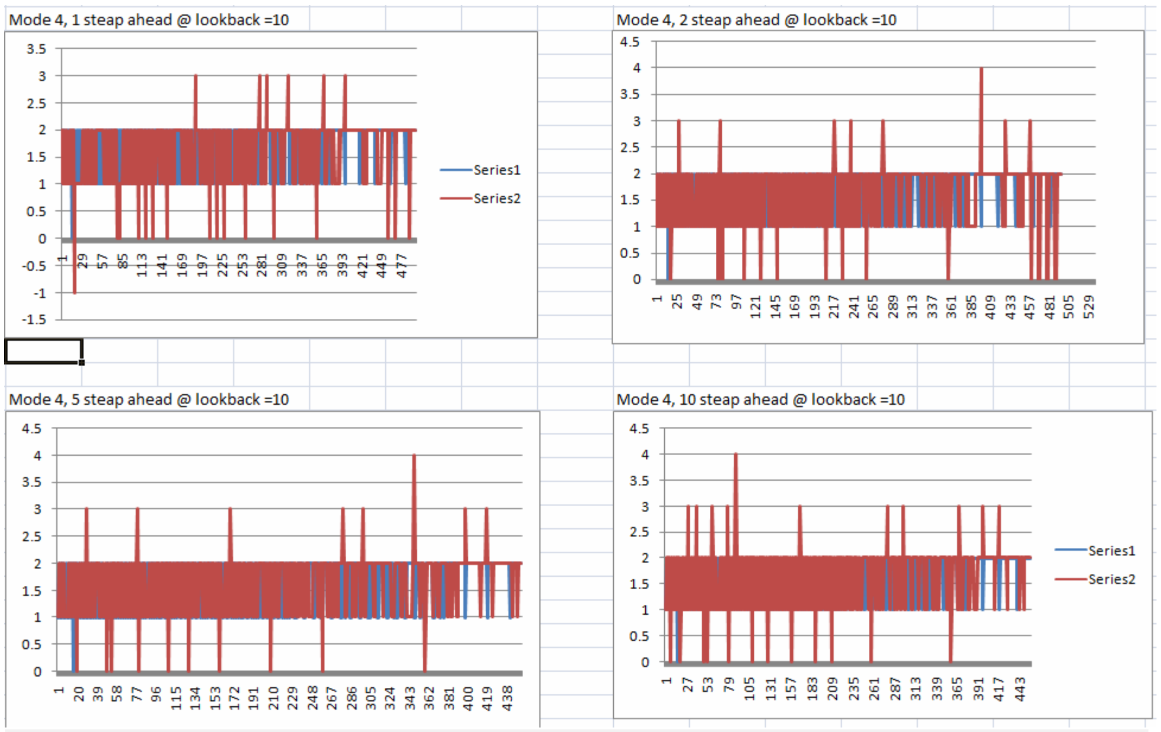
For input length 300, window size 50 gives 3 moderate and 2 bad results.

It is evident from the table that the window length 30 performs consistently for all the input lengths. Also, when the graphs are compared based on the input lengths, it displays that the input length 100 gives more accurate results on the sink-hole dataset. Thus, the input length is set at 100 and the working window is set at 30.

4.4 Multi-step Ahead Predictions

The algorithm was modified to output multi-step ahead prediction results for each pass. Step-ahead prediction here refers to the ability predict multiple point in a single pass of the algorithm without re-looking at the input elements or historic data. The algorithm performs 2 types of value predictions. The first type is by only looking at the input scanned and the second is by rescanning the historic data into the input array.

The results have been compared with the actual values obtained from the sink-hole attack and are shown in Fig 7. The single step-ahead algorithm gives the best results. The difference between, original and predicted values starts to increase, as the number of points predicted per step increases. This is can be attributed to the frequency/ exponential component of the algorithm as they flatten the graph for 10 points in-case of a failed trend prediction.

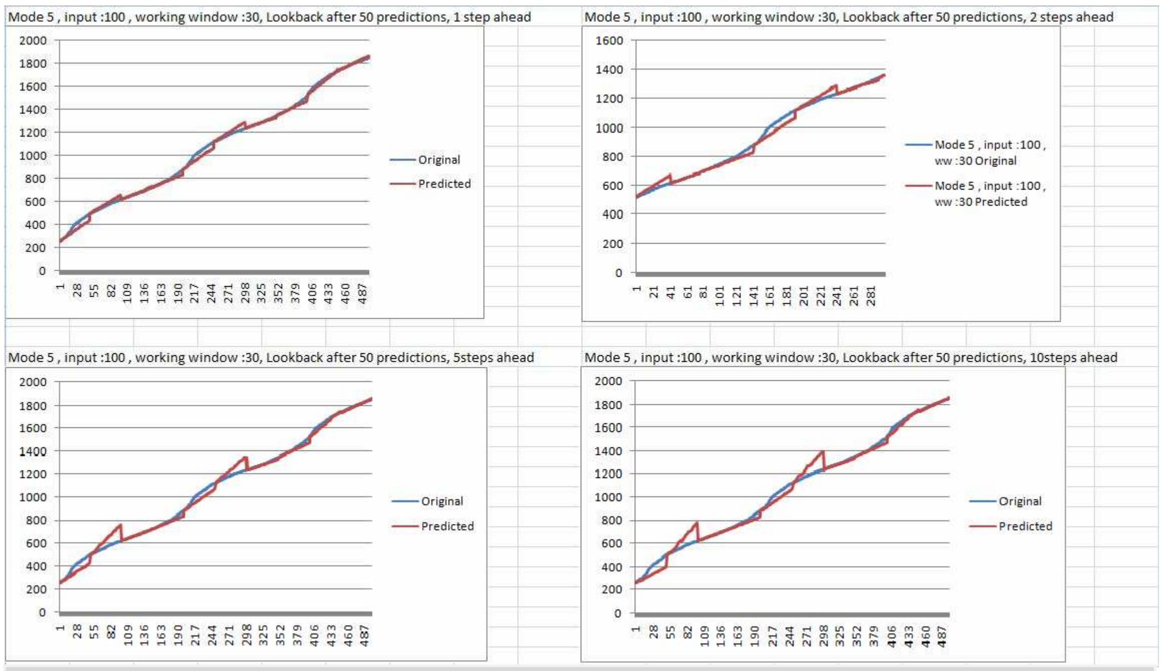


x-axis: local time at attacker node , y-axis: attacker's transmission time difference (secs)
series 1 : original values, series 2: predicted values

Fig 9. Comparison of values obtained with multi-step ahead prediction and sinkhole data

4.5 Usage as a System with Look-back

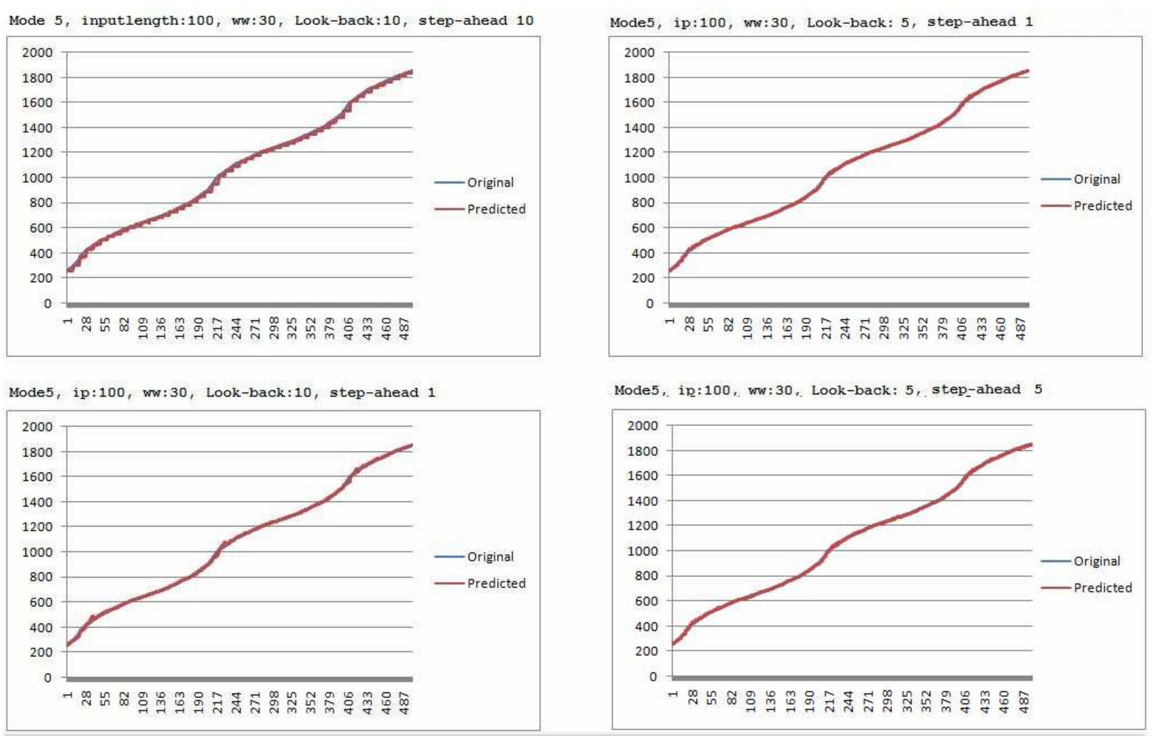
The algorithm is used as a system of comparison in our deception framework, and thus we get the opportunity to look-back at the data after certain time interval. Look-back here refers to the method of rescanning input elements from the original time series, and replacing the input array with the new values. The algorithm can be set to learn from the historic data after a certain number of predictions. Look-back allows the system to correct any errors which might have propagated into the predicted results. It has been found that the algorithm performs within the error margin of 650 (Mean square error) or 2.196% (Mean percentage error) for sine-wave pattern, if it is allowed to scan the data after 50 predictions using a single step ahead at a time. At each look-back the system re-scans the values from the historic data to predict the next *n* points. The figure below shows the comparison between single step and multi step ahead predictions for a sine-wave based data (mode 5) obtained from the sink-hole attack.



x-axis: attacker's transmission sequence(1st, 2nd..n) , y-axis: message transmission time of attacker in seconds

Fig 10. Comparison of values form high look-back and varying step-ahead points

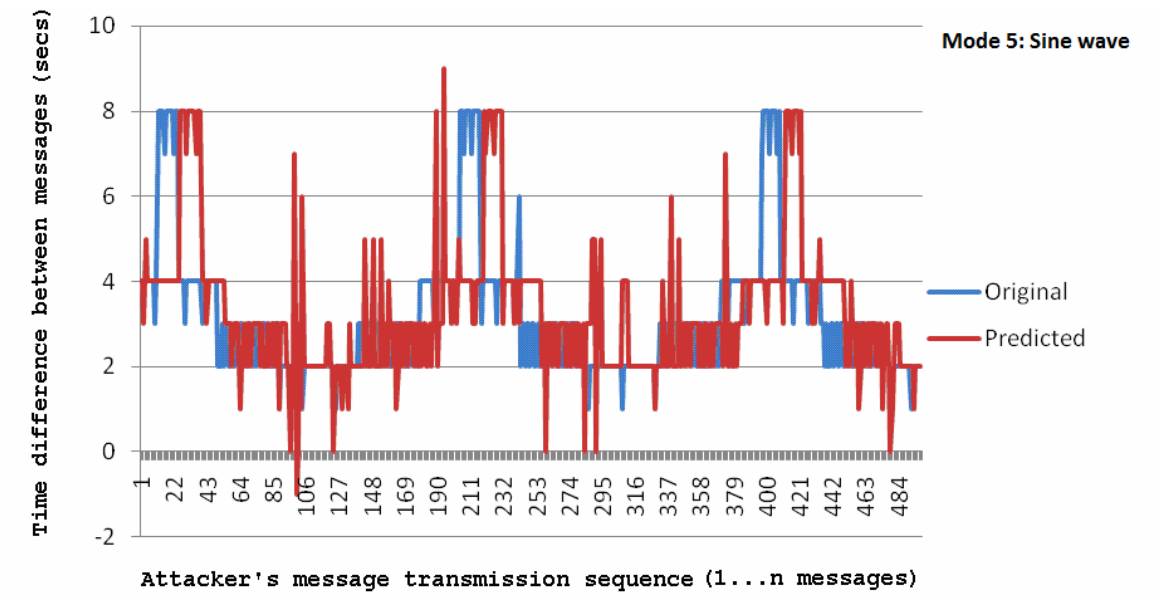
It is clear from the graphs that the algorithm produces its best results when it is used as a single step-ahead prediction algorithm (Fig 7.a). There is a slight lag before the values aligns with the historical data points and is determined by the look-back that set for the algorithm. If a lower value is set for the look-back, the system produces accurate results which are almost very close to the historic data, which it scans. The look-back has been set at 50, because it reduces the number of scans of the raw data and still produces acceptable results. The Figure below, displays the results obtained by setting the look-back at a lower value, as 10.



x-axis: attacker's transmission sequence(1st, 2nd..n) , y-axis: message transmission time of attacker in seconds

Fig 11. Comparison of values from low look-back and varying step-ahead points for mode 5 (sine wave)

As is evident from the graphs in Fig 11, the values are almost perfectly aligned with the historical data. This is because of the low value of re-scans set in the algorithm. The tests conducted used the sink-hole data has led us to conclude that, from a practical standpoint, it is appropriate to set the look-back at a value around 50, so that there are not too many rescans to remove the error margin from the predicted value.



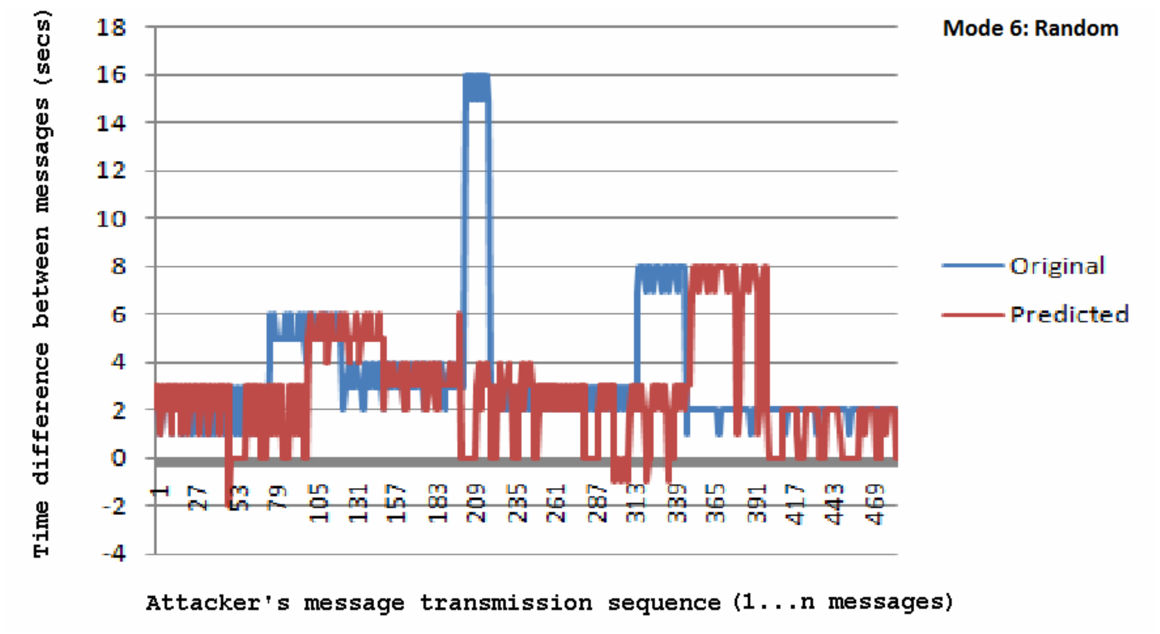


Fig 12. Comparison of original and predicted values for mode 5 & mode 6

The Figure above shows the comparison between values from the original dataset and the predicted results. As is evident from the graph, the values show a time lag, before the values are re-plotted on the predicted results and are different, because of the look-back interval set. The results are not perfectly similar to the original values and shows distortion. In these graphs, some values were omitted to show the internal data structure.

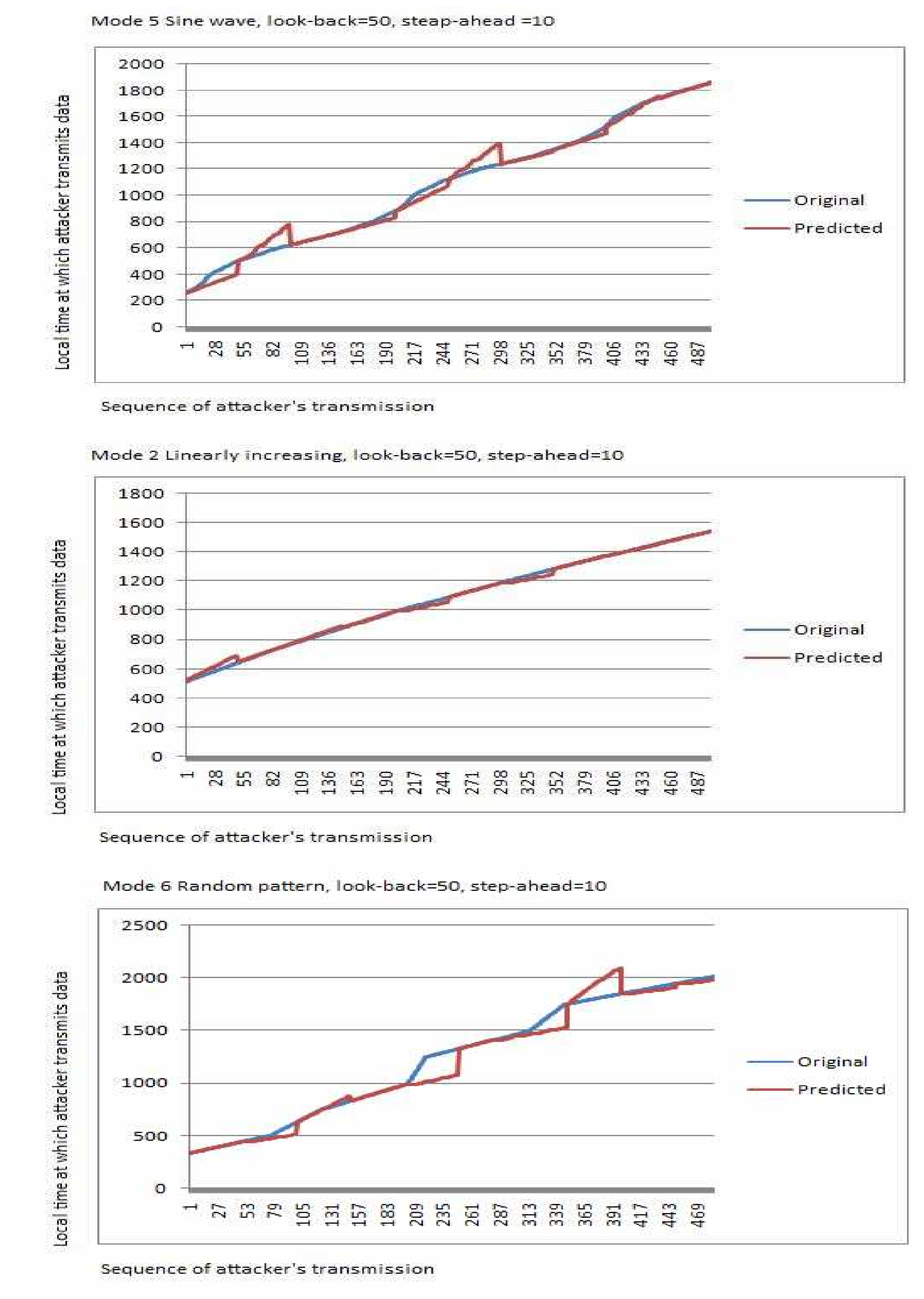
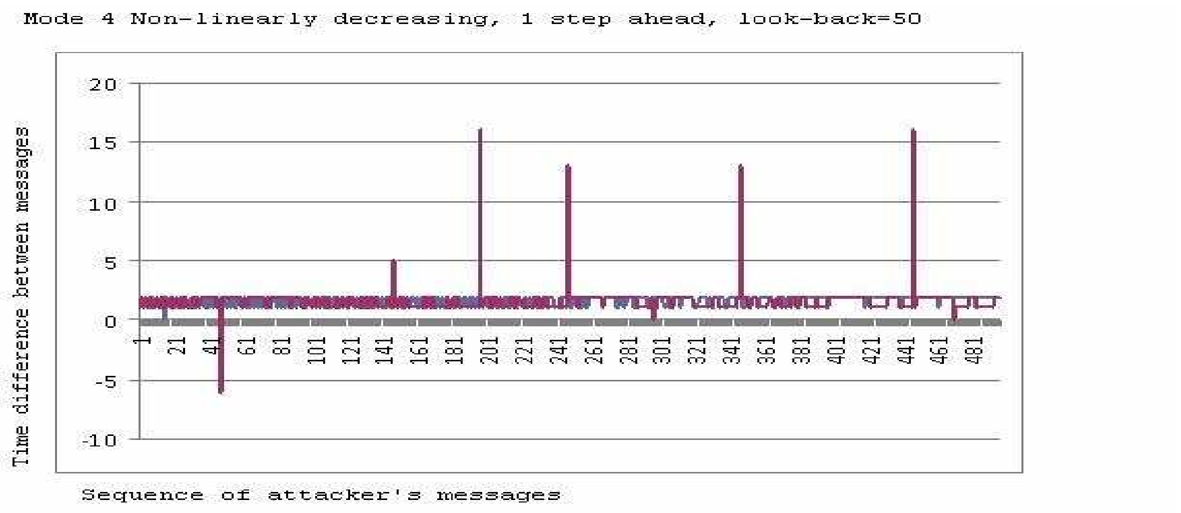
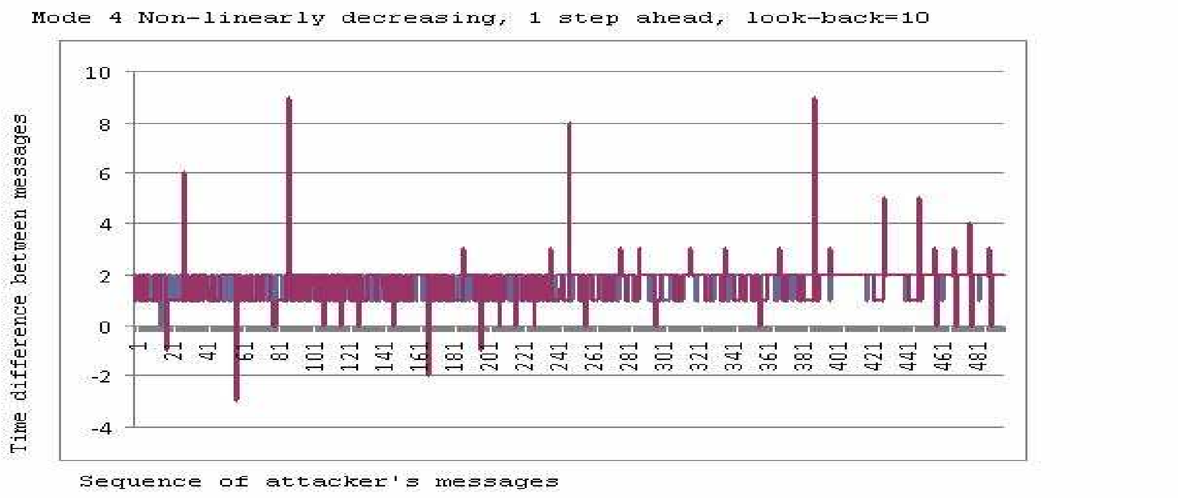


Fig 13. Different modes, using constant look-back and step-ahead values





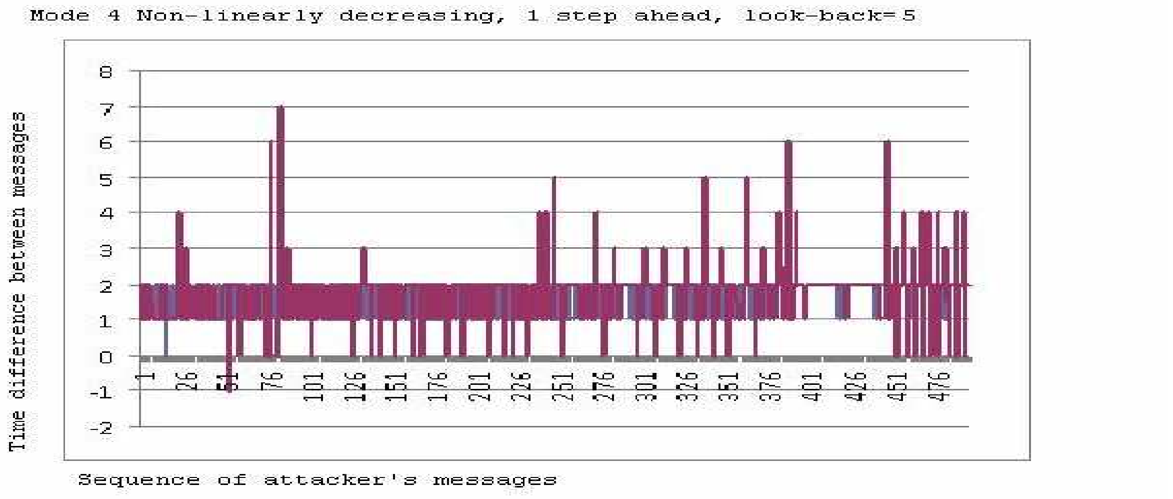


Fig 14. Constant mode and step-ahead with varying look-backs

From fig. 13, we can conclude that the algorithm takes time to align itself to the original values if there is a steep increase or decrease from the adjacent values. This is evident in the figure as it is almost aligned for constantly increasing values. In the fig. 14, the graphs show the differences between the original and predicted values. It is clear that, the difference increases with the increase in look-back. So for more accurate predictions, the look-back scan parameter has to be set to a low value.

4.6 Analysis of the Algorithm

No. of passes: single pass

Run-time of the algorithm: Constant *n*

Memory requirements: Constant *n*

The proposed approach is therefore very simple as desired.

Mean percentage error for all the modes

Mode 1 – Linearly increasing	
Look-back : 5	**Look-back : 10**
1 step-ahead : 0.064	1 step-ahead : 0.101
2 step-ahead : 0.919	2 step-ahead : 0.123
5 step-ahead : 0.127	5 step-ahead : 0.21
10 step-ahead : 0.2	10 step-ahead : 0.258
Look-back : 50	
1 step-ahead : 0.433	
2 step-ahead : 0.338	
5 step-ahead : 0.778	
10 step-ahead : 1.36	

Mode 2 – Non-linearly increasing	
Look-back : 5	**Look-back : 10**

1 step-ahead : 0.083	1 step-ahead : 0.117
2 step-ahead : 0.113	2 step-ahead : 0.148
5 step-ahead : 0.131	5 step-ahead : 0.225
10 step-ahead : 0.23	10 step-ahead : 0.272
Look-back : 50	
1 step-ahead : 0.36	
2 step-ahead : 0.623	
5 step-ahead : 0.8331	
10 step-ahead : 0.98	

Mode 3 – Linearly decreasing	
Look-back : 5	**Look-back : 10**
1 step-ahead : 0.214	1 step-ahead : 0.34
2 step-ahead : 0.216	2 step-ahead : 0.361
5 step-ahead : 0.308	5 step-ahead : 0.519
10 step-ahead : 0.404	10 step-ahead : 0.602
Look-back : 50	
1 step-ahead : 1.49	
2 step-ahead : 0.91	
5 step-ahead : 1.354	
10 step-ahead : 1.66	

Mode 4 – Non-linearly decreasing	
Look-back : 5	**Look-back : 10**
1 step-ahead : 0.186	1 step-ahead : 0.278
2 step-ahead : 0.167	2 step-ahead : 0.209
5 step-ahead : 0.171	5 step-ahead : 0.225
10 step-ahead : 0.128	10 step-ahead : 0.2449
Look-back : 50	
1 step-ahead : 0.996	
2 step-ahead : 0.417	
5 step-ahead : 0.347	
10 step-ahead : 0.425	

Mode 5 – Sine wave(Combined)	
Look-back : 5	**Look-back : 10**
1 step-ahead : 0.242	1 step-ahead : 0.495
2 step-ahead : 0.216	2 step-ahead : 0.536
5 step-ahead : 0.304	5 step-ahead : 0.7488
10 step-ahead : 0.433	10 step-ahead : 0.748
Look-back : 50	
1 step-ahead : 2.196	

2 step-ahead : 1.73	
5 step-ahead : 3.9	
10 step-ahead : 4.05	

Mode 6 – Random pattern	
Look-back : 5	**Look-back : 10**
1 step-ahead : 0.312	1 step-ahead : 0.623
2 step-ahead : 0.443	2 step-ahead : 0.647
5 step-ahead : 0.387	5 step-ahead : 0.77
10 step-ahead : 0.99	10 step-ahead : 0.84
Look-back : 50	
1 step-ahead : 3.314	
2 step-ahead : 3.04	
5 step-ahead : 3.88	
10 step-ahead : 4.21	

All values are percentages.

Table 2. Mean percentage error calculated for all the available sink-hole data patterns

4.7 Comparison of Complexity

This complexity of the algorithm is much less when compared to some of the general purpose prediction algorithms widely used. Since there are only a few comparison/assignments and also since the parameters are set as a constant, the complexity of this algorithm is lower than other general purpose prediction algorithms. For example, the time complexity of F4 Fractal Forecasting has been specified as $O(NL_{opt}^2)$ [3], where L_{opt} is the optimal lag length, whereas the time complexity of this algorithm is a constant.

CHAPTER V

CONCLUSION

In this thesis, we proposed an algorithm to speed-up the process of time series prediction, which has been tailored to suit wireless sensor networks. Based on the previous researches and the limitations of wireless sensor networks, we proposed an approach of trading accuracy for efficiency. Our plan was to predict values fast enough to be used in a wireless sensor network prediction, which generally works on a small window frame.

Simulation results support our claim, as we were able to predict the values fairly accurately. The prediction module fits into a larger framework of a deception system, which is not part of this thesis. The values obtained from the prediction model, is used to defend against an attack from a potential attacker who has already been identified.

Future work can include porting this algorithm into a real sensor network for achieving the larger and more holistic goal of defending against an attacker without ejecting him off the network. Furthermore, the determination on the fly of parameters used by the system such as the input length, working window length etc., would enhance the usability of the algorithm.

REFERENCES

[1] Feng Zhao and Leonidas J. Guibas., *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann, 2004

[2] Frank Stajano, Dan Cvrcek, and Matt Lewis., Steel, Cast Iron and Concrete: Security Engineering for Real World Wireless Sensor Networks. *Applied Cryptography and Network Security*, Pages 460-478, 2008.

[3] Deepayan Chakrabarti, and Christos Faloutsos., F4: Large-Scale Automated Forecasting Using Fractals. In *Conference on Information and Knowledge Management'02*, Pages 1-8, November 2002

[4] J. E. Gehrke, F. Korn, and D. Srivastava., On computing correlated aggregates over continual data streams. In *Proceedings of the 2001 ACM Sigmod International Conference on Management of Data,* Pages 1-12, May 2001.

[5] P. Bonnet, J. E. Gehrke, and P. Sheshadri., Towards sensor database systems. In *Proceedings of the Second International Conference on Mobile Data,* Pages 3-14, January 2001

[6] S. Babu and J. Widom, Continuous queries over data streams. *Sigmod Record*, Pages 109-120, September 2001

[7] Time series predictions, http://www.cis.hut.fi/research/reports/biennial04-05/cis-biennial-report-2004-2005-12.pdf [last accessed - 2005]

[8] Sameer Singh, Elizabeth Stuart., A Pattern Matching Tool for Time-Series Forecasting, In *Proceedings of the 14th International Conference on Pattern Recognition*-Volume 1, Pages 103-105, 1998

[9] Filtering of Time Series, http://www.atmos.washington.edu/~dennis/552_Notes_7.pdf [last accessed - Mar 17, 2008]

[10] Jarkko Tikka, Amaury Lendasse, and Jaakko Hollme´n., "Analysis of Fast Input Selection: Application in Time Series Prediction", Internet Corporation for Assigned Names and Numbers 2006, LNCS 4132, Pages 161-170, 2006

[11] Jarkko Tikka, Jaakko Hollme´n., “Sequential input selection algorithm for long-term prediction of time series”, *Neurocomputing* Volume 71, Pages 2604– 2615, 2008

[12] Ioannis Krontiris, Tassos Dimitriou, Thanassis Giannetsos and Marios Mpasoukos., “Intrusion Detection of Sinkhole Attacks in Wireless Sensor Networks”, Algorithmic Aspects of Wireless Sensor Networks 2007, LNCS 4837, Pages 150– 161, 2008

[13] Chris Karlof and David Wagner., “Secure routing in wireless sensor networks: attacks and countermeasures”, In *Proceedings of the IEEE International Workshop on Sensor Network Protocols and Applications*, Pages 113-127, 2002

[14] S. Kak, “A class of instantaneously trained neural networks”, *Information Sciences*, vol. 148, Pages 97-102, 2002.

Atul Ravindran

Candidate for the Degree of

Master of Science

Thesis: DECEPTION IN WIRELESS SENSOR NETWORKS – PREDICTING INTRUDER BEHAVIOR

Major Field: Computer Science

Biographical:

Personal Data:

Born on 7th March 1984 in India.

Education:
Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in August, 2009.

Experience:
Worked as a software professional at Tata Elxsi Ltd., from Oct 2005 – Jul 2007.

Name: Atul Ravindran Date of Degree: July, 2009

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: DECEPTION IN WIRELESS SENSOR NETWORKS – PREDICTING INTRUDER BEHAVIOR

Pages in Study: 43 Candidate for the Degree of Master of Science

Major Field: Computer Science

Wireless sensor networks (WSNs) have been used in different sectors such as transportation, agriculture, military etc., Since WSNs are generally used in unmanned territories, and providing security is an important requirement. Deception in WSNs is a method for providing a security framework. Time-series prediction is a component used to validate the effectiveness of the deception framework employed. The time-series prediction implemented in this thesis requires little memory and computation power to predict within a certain accuracy threshold. It works by scanning the data towards the end of the existing time series, checking for patterns and trends. Averaging is applied if it is not possible to arrive at a definite conclusion based on the data scanned. The algorithm was tested with data obtained from a sink-hole attack in WSNs. Simulation results show that the prediction was most accurate with a look- back value of 10 and input lengths set to 100. The proposed approach requires little memory and computational power and is therefore suitable for WSNs.

ADVISER’S APPROVAL: Dr. Johnson Thomas