Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 11, Number 1 (2015), pp. 89-104

http://www.ripublication.com

Code Based Rescheduling: A Highly Reliable Inter Layer Scheme to Prevent Physical layer Threat in Wireless Network

S. Raja Ratna*1 and Dr. R. Ravi 2

1Full Time Research Scholar, Department of Computer Science and Engineering, Francis Xavier Engineering College, Tirunelveli.
2Professer, Department of Computer Science and Engineering, Francis Xavier Engineering College, Tirunelveli, India.
1 gracelinrr@yahoo.com, 2csehod@francisxavier.ac.in
*(*Corresponding author)*

Abstract

A major threat to the information economy is the launching of physical layer threat in which the attacker effectively and stealthily corrupts the packet by injecting high level of noise resulting in packet loss at the receiver side. Traditional cryptographic way of protection is no longer efficient to prevent this threat, therefore this paper proposes a highly reliable Code Based Rescheduling scheme to conceal and shield the packet from the attacker. To prevent this attack, the proposed scheme operates in two phases in which the packet to be transmitted is concealed between the layers and this helps the physical layer to circumvent the inside attacker. In the first part, the log files are analyzed to extract the packet transfer time for different routes from the source to the sink. Using the extracted timing information, the test data is monitored for different routes and then a reliable route is identified using JIM flag. The second part rearranges the contents of the frame's blocks to be transmitted and provides secure communication between layers through many rounds of process. A separate code is generated for each round of process for secure transmission. From simulation studies, it is observed that this scheme achieves high throughput and low delay even in harsh environment. The throughput increases to 0.36mbps and delay decreases to 2.1sec for single jammer with 0.1 attack probability. It can also shield data even when the attack probability and number of attackers increases.

Keywords – Adversary, Corrupt, Communication, Rescheduling, Security.

1. INTRODUCTION

Owing to the open nature of communication medium, information systems are highly exposed to jamming attack. The attacker continuously emits signal on the channel so that the sender will always sense the channel as busy or it will overpower the transmitted signal by injecting high level of noise thereby disrupting the communication. As a result of jamming, legitimate traffic is completely blocked thus preventing successful data reception [3]. High level of noise (injecting false messages) corrupts the packet resulting in packet loss at the receiver side, thereby lowering throughput and increasing delay.

The aim of the attacker is to obstruct legitimate communication and it achieves by preventing the reception of legitimate packets. By doing so severe security threats are achieved at the physical layer and the attacker also prevents the users from using legitimate physical layer operations [1], [16]. Understanding the impact and complexity of such attacks and their countermeasures is of great interest to the network community.

Jammers are of four types, namely, constant, random, deceptive and reactive [20], [21]. Normally, the attacker attack the network from outside, but this paper considers jammer to be an inside reactive jammer. Reactive jammer is more efficient and its network performance does not degrade a lot like other type of jammers. The jammer is designed as a part of the network knowing all network secrets and it resides inside the network and is very dangerous because it leak vital information to the third party. The reactive jammer listens to the network activities and takes out significant information like packet type, source address or destination address of the transmitted packets, thus launching jamming attack which is difficult to identify. The jammer induces large amount of error bits into the packet and then retransmits it, so that the packet is wrongly received at the receiver side. It is better to prevent the data at the initial stage of attacker rather than to get cured after been attacked.

Due to this unique characteristic of jammer, an effective prevention scheme named Code Based Rescheduling (CBR) has been proposed to increase the security level of the packets to be transmitted. This scheme conceals the packet between the layers so that the attacker cannot find it. It consists of two phases: Timer Route Identification and Inter Layer Rescheduling. The timer route identification phase analyzes the log files to extract the packet transfer time for different routes between the source and the sink. Using the extracted timing information, the test data is monitored for different routes to eliminate the suspicious routes and to select a suitable route for data transmission. Inter Layer Rescheduling phase conceals the packet between layers and it consists of three steps: block rescheduling, flow connectivity and code evaluation. Specifically, the block rescheduling step is introduced to rearrange the content of the block and then the flow connectivity step is conducted based on the rescheduling results. Next the code evaluation step is used to provide secure communication between layers. CBR demonstrates significant advantage in terms of providing security under attack.

The paper proceeds as follows. Section 2 describes related works. Section 3 explains problem statement and assumptions of the proposed work. Section 4 explains Timer Route Identification phase, while section 5 describes Code Based Rescheduling

phase. Section 6 presents the simulations conducted in order to evaluate Code Rescheduling Scheme and summarizes the result. Finally, Section 7 concludes the paper.

2. RELATED WORKS

The focus of this work is to shield the data from attacker thereby preventing inside reactive jammer. There are number of prior works available that aid in preventing jamming attack like spread spectrum [5], frequency hopping [7] [12], multi path routing [6] [9], packet hiding [13], protocol based technique [15] [22]. But these works could prevent jamming attack only after the attack takes place, but this paper prevents jamming before attacking.

Proano *et. al.* in [13] have investigated the impact of an internal selective jammer who targets packets of high importance. The adversary is active only for a short period. They have also explained selective jamming in terms of network performance degradation. They have developed three schemes that prevent real time packet classification by combining cryptographic primitives with physical layer attributes. The packets to be transmitted are hidden between physical and MAC layer and then transmitted. Throughput and delay are studied on different types of jammers.

Chiang *et. al.* in [5] have proposed an optimized power efficient code tree system that provides input to physical layer and also helps the physical layer circumvent the jammer. Each receiver cooperates with the transmitter to detect any jamming that affects the receiver. Each transmission is sent on at most 2j+1 code simultaneously and results are based on evaluating packet delivery ratio.

Richa *et. al.* in [14] have proposed a simple, fair, self-stabilizing distributed MAC protocol called ANTIJAM to mitigate internal interference, requiring no knowledge about the number of participants in the network and it is also robust to intentional and unintentional external interference. The protocol is efficient and fair against powerful reactive adversaries who have complete knowledge of the past history. ANTIJAM features low convergence time and has excellent fairness property and also achieves constant throughput. Throughput is dealt under different jamming strategy as a function of network size.

Li *et. al.* in [7] have evaluated the communication efficiency of Uncoordinated FH (UFH) and Collaborative UFH (CUFH) in large-scale networks with the aim of preventing jamming in multi-channel networks using network delay as a metric. In this network the numbers of nodes are large and may exceed the number of channels. Without the use of secret keys, UFH achieves robustness to inside jammers but achieves poor communication efficiency due to the lack of coordination between the source and sink. Sub-optimal protocol CUTH-p has been proposed which simplifies the implementation of CUFH. In order to obtain better packet reception rate the number of relays are controlled in CUFH.

Pelechrinis *et. al.* in [8] have investigated an Anti-jamming Reinforcement System for random jammers and it uses both power control and rate control module to prevent jamming. Based on channel condition, the rate adaptation module assigns the transmission rate. In order to increase successful packet reception, power control

module tunes the clear channel assessment threshold. Appropriate tuning allows the transmitter to send packets even when jammed and it is examined using throughput as a metric.

Pelechrinis *et. al.* in [11] have proposed proactive frequency hopping technique to prevent jamming attack. A game theoretic approach is used to capture the interaction between link and jammer employing frequency hopping. If the number of orthogonal channel was larger, then proactive FH would be very effective in terms of throughput.

Cagalj *et. al.* as in [19] have focused on the prevention of radio channel jamming in sensor networks using worm hole based anti jamming techniques. A jammer hides some events that a sensor node would detect by jamming an appropriate subset of nodes, thus preventing the node from reporting to the network operator. Therefore the network operator could not receive the reports in time. To solve this problem three wormholes based anti jamming techniques have been proposed to mitigate jamming based on wires, frequency hopping and uncoordinated channel hopping. Wormholes are used as a reactive defensive mechanism. The sensor nodes can develop channel diversity in order to generate wormholes out of the jammed region, through which an alarm can be send to the network operator. Using channel diversity, the nodes under jamming themselves creates a route and escape from jamming. Mathematical models are also developed to study the proposed solutions.

Chen *et. al.* in [2] have proposed a Game Theoretical Anti jamming Scheme (GTAS) for preventing jamming attack in cognitive radio networks. Jamming and anti-jamming process are modeled as a Markov decision process between two players in a game attacker and the secondary users. To avoid jamming launched by external attacker's the secondary users proactively hop among accessible channels but this approach increases cost. To solve this problem, policy iteration scheme is proposed but this scheme is computationally complex. To reduce computation complexity, Q-function based approach has been proposed. GTAS achieves low computation complexity and very good throughput.

Popper *et. al.* in [17] focused on spread-spectrum anti-jamming broadcast without the requirement of shared secrets. The uncoordinated direct sequence spread spectrum modulation scheme is used by the communication nodes.

The proposed scheme has three contributions over prior schemes. 1) It captures the benefit of prevention before attacking, 2) To enhance network performance; the packets are concealed between layers thereby shielding the data. 3) The code is rescheduled for each round, so jammer could not easily identify it.

3. PROBLEM STATEMENT AND ASSUMPTIONS

3.1 Problem Statement

Consider two nodes u and v which communicate through the wireless medium. A jammer J is present within the communication range of u and v, and is a part of the network knowing all the network secrets. When the node u is transmitting a packet N to node v, the jammer corrupts the packet by injecting unwanted extra bits into it. The objective of the proposed work is to prevent the jammer from injecting unwanted bits

into the packet, thereby allowing the packet to reach the receiver safely. To do so, the packet has to be concealed from the jammer in order to avoid this attack. The main scenario is to make the network free from jammer and to improve the network performance.

3.2 Network Model

A wireless network consisting of *n* simple reliable nodes connected through wireless links is considered. If nodes are within the communication range of each other they communicate directly or communicate via multi hop through the wireless medium. A scenario is created where the nodes continuously send packet on the wireless channel.

3.3 Attacker Model

Attacker is placed in the multi hop path between the source and the sink in a network. When a node is transmitting a packet to another node, the attacker is modeled to interrupt and corrupt the packet by injecting large amount of error bits (noise) into the packet so that the packet is wrongly received at the receiver side. The attacker can jam any part of the packet sent from the source to the sink. The jammer node is superior to normal nodes. The inside jammer is designed as a part of the network knowing all network secrets and resides inside the network. The attacker is designed to know the protocol and the network channel status, this information helps the attacker to attack the network at any time. Whenever the jammer jam the channel, all nodes will notice the channel as busy, but the jammer can easily identify whether the channel is idle or busy. This paper considers jammer to be a reactive jammer. This jammer intensively listens to the network activities and then jams when the important data is transmitted.

3.4 Overview of Code Based Rescheduling (CBR) Scheme

The main steps of Code Based Rescheduling scheme are Timer Route Identification phase and Inter Layer Rescheduling phase. The proposed scheme selects a suitable route for data transmission and then conceals the packets to transmit through that route. The overview of Code Based Rescheduling Scheme is as follows

- In Timer Route Identification phase, the timer calculates the time needed for transmitting the data in all possible routes between the source and the sink and save it in the log file. Suspicious routes are identified and isolated by transmitting the test data and by using log file values a suitable route is selected for data transmission.
- The Inter Layer Rescheduling phase is the integration of three steps: a) block rescheduling, b) flow connectivity and c) code evaluation. In this phase, the data is concealed at the sender side and then transmitted. The content of the data blocks are rearranged using the block rescheduling step and then the flow connectivity step is conducted based on the rescheduling results and finally secure communication takes place between the layers using code evaluation step.

4. TIMER ROUTE IDENTIFICATION PHASE

The Timer Route Identification phase monitors the network for the problems caused by the attacker that reside in the network and corrupts the legitimate packets. This approach can be served as a preliminary process for identifying a suitable route from the source to the sink. The source node maintains a timer to collect the timing information of all nodes in a particular route. The source node (node 0) is denoted by n_0 or u, the first forwarder by n_1, the last forwarder by n_{fk} and the sink node is denoted by n_{fk+1} or v. For any route its source, the forwarding nodes and the sink are represented as $\{u, n_1, n_2 \ldots n_{fk}, v\}$.

Before the source u transmits a packet to the sink v, the source finds out m possible routes to reach the sink. Each node in the network is assigned a node id. The sink maintains a log file $L_{(u,v)}$ for m successful routes between u and v; it also maintains a counter C_p which keeps track of the number of forwarding nodes between u and v for all m routes. The node u maintains a timer T_p which calculates the time needed to transmit the data from one node to one hop next forwarder. Let n_2 and n_3 be two adjacent forwarding nodes and T_p calculates T_{n2n3}, the time required for transmitting the data between n_2 and n_3.

The timer gets updated at two places, when a packet enters a node n_2 until leaving the same node and when the packet leaves the node n_2 and enters adjacent forwarder n_3. Let T_{node} be the time needed for processing at a particular node and T_{link} be the time needed to move across the link from one node to adjacent node. The packet entering time and exiting time at node n_2 are represented as $T_{n2,ent}$ and $T_{n2,exit}$. Likewise the packet entering and exiting time at node n_3 are $T_{n3,ent}$ and $T_{n3,exit}$. The time $T_{node,n2}$ is the time difference between packet exiting n_2 and entering n_2, similarly $T_{link,n2}$ is the time difference between packet entering n_3 and exiting n_2. The time required for transmission between n_2 and n_3 is given in equation (1).

$$T_{n_2 n_3} = \left(T_{n_2,exit} - T_{n_2,ent} \right) + \left(T_{n_3,ent} - T_{n_2,exit} \right)$$

$$T_{n_2 n_3} = T_{node,n_2} + T_{link,n_2} \qquad (1)$$

Finally, the node n_3 composes and sent the following information to the sink $\{n_{2id}, n_{3id}, T_{n2n3}\}$, where n_{2id} is node n_2's id, n_{3id} is node n_3's id and T_{n2n3} is the time required for transmission between nodes n_2 and n_3. The time required for transmission between all adjacent nodes is similarly calculated and sent to the sink as shown below. Likewise the time is calculated for all m routes and sent to the sink. Let f_k be the number of forwarding nodes in a route, for f_k forwarding nodes there is f_k+1 links represented as l_k. The time needed for packet transfer between source u and the first forwarder n_1 is $T_{u,n1}$, similarly $T_{nfk,v}$ is the time between last forwarder f_k and the sink v. When the sink is f_k hops away from the source, equation (2) represents the total time $T_{u,v}$ required to transfer the packet between u and v.

$$T_{u,v} = \sum_{i=0}^{fk} T_{n_i,n_i+1} \qquad (2)$$

The information maintained in the sink for a route looks like the following: source id u_{id}, number of forwarders f_k, list of forwarders $\{n_1, n_2 \ldots n_{fk}\}$, time required for transmission between all adjacent forwarding nodes, total time $T_{u,v}$ and sink node

id v_{id}.

$$r_1 = \left\langle u_{id}, f_k, \ n_1, n_2 \ldots n_{f_k} \ , \ T_{u,n_1}, T_{n_1,n_2} \ \ldots T_{n_{f_k},v} \ , T_{u,v}, v_{id} \right\rangle$$

The sink also maintains *m* routes $\{r_j \mid _{j\in[1,m]}\}$ information in the log file. In order to detect the suspicious route, before sending the original data the source transmits the test data encrypted using its secret key. This test data allows the source and the sink to cooperatively detect jamming. If there is no jammer, the sink would receive the same test data sent by the source. If the sink is not able to decode correctly and receives a wrong message or if it exceeds the time $T_{u,v}$ calculated by the timer, then the jamming can be suspected. After suspecting the jamming in a particular route, *v* reports the jamming activities to *u* by sending Jamming Identified Message (JIM) using the flag *f* shared between *u* and *v*. The jammers are unaware of the flag and if *f* is set as zero, no jamming occurs and no message is sent to *u*, but if jamming occurs *f* is set as one and JIM message is sent to *u*. If jamming is suspected in a particular route then that route is marked as suspicious and packets are not transmitted in that route. By this process jamming can be prevented to some extent, in order to completely prevent jamming data must be concealed from the jammer using code based rescheduling phase.

5. INTER LAYER RESCHEDULING PHASE

The code based rescheduling phase is the integration of three steps: a) block rescheduling, b) flow connectivity and c) code evaluation. In this phase, the data is concealed from the attacker at the sender side and then transmitted. The outline system design of the code based rescheduling part is presented in Figure 1.

5.1 Block Re-scheduling step

In block rescheduling step, the content of the data blocks to be transmitted is rescheduled and it acts as the input to the Flow based Connectivity step. The steps of block rescheduling step are as follows.

Step 1: The frame at data link layer is divided into *n* sets of 64-bit each Set_1, Set_2 … Set_n. They are then passed to a Re-scheduler which works from step 2 to 7.

Step 2: For each set, the re-scheduler pad extra bits to the block to adapt the length of a square matrix. Let A_{ij} be an ω x ω square matrix with x_λ and y_λ be the number of rows and columns.

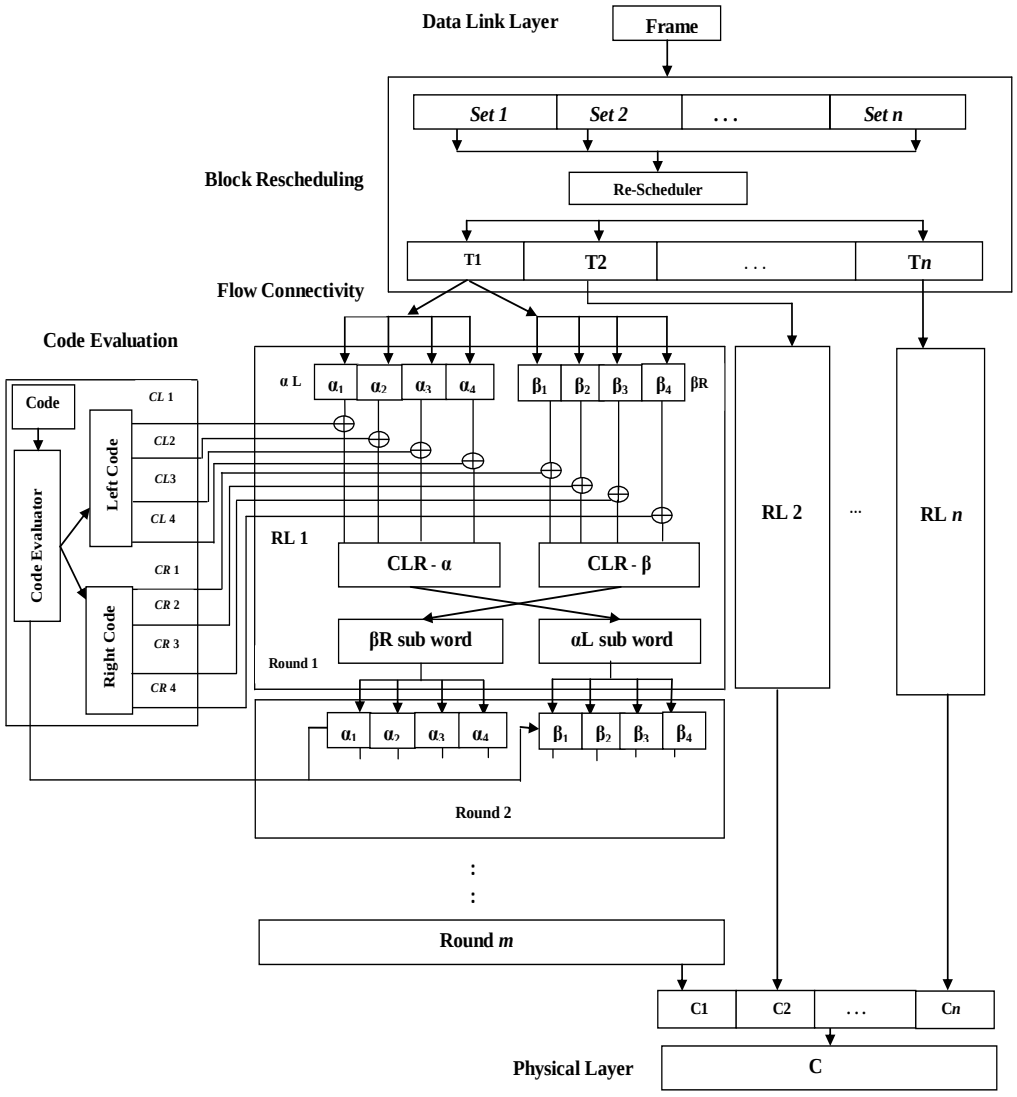

Figure 1. Functional architecture of Inter Layer Rescheduling Phase

$$A_{ij} = A\,[x_i\ y_j] \quad / \quad 1 \le i, j \le \omega$$
$$A[x_\lambda] = A\,[y_\lambda] = \omega \quad \text{if} \quad 1 \le i, j \le \omega$$

Step 3: For *n* sets, totally *n* square matrices are obtained A_1, $A_2 \ldots A_n$ are obtained. The operations from step 3 to step 5 are performed individually for each matrix.

Step 4: If A contains ω entries, circularly rotate the column elements left side and then circularly shift the row elements downwards. After column shifting operation, x_λ of matrix A_{ij} is changed to x_α as in equation (3)

$$\sum_{\lambda=1}^{\omega} A\ x_\lambda \Rightarrow \sum_{\alpha=1}^{\omega} A\ x_\alpha \ , \quad \alpha = \begin{cases} \lambda - \omega + 1 \ , & \lambda = \omega \\ \lambda + 1 \ , & \text{otherwise} \end{cases} \qquad (3)$$

Likewise after row shifting operation, y_λ of matrix $A_{i\,j}$ is changed to y_α based on the values of λ and ω and is represented in equation (4). The summation denotes all the elements.

$$\sum_{\lambda=\omega}^{1} A \quad y_\lambda \quad \Rightarrow \sum_{\alpha=\omega}^{1} A \quad y_\alpha \quad , \quad \alpha = \begin{cases} \omega - \lambda + 1, & \lambda = \omega \\ \lambda - 1 & , \text{ otherwise} \end{cases} \qquad (4)$$

Step 5: Compute A' by transposing equation (3) and equation (4) on A

$$A'[x_i y_j] = \sum_{i=1}^{\omega} \sum_{j=1}^{\omega} (A^T[x_i y_j]) \qquad (5)$$

Step 6: Let the notation $A^i{}_k$ represent an array A with *k* rows and *i* elements. Diagonally Exchange the elements of the matrix A' and to increase the security level, diagonally exchanged matrix is again right shifted and store it in matrix B' with π be a constant as shown in equation (6).

$$B'[x_i \; y_j] = \begin{cases} \sum_{\lambda=1}^{\omega} \begin{cases} \pi = A'^{\lambda}_{\lambda} \\ A'^{\lambda}_{\lambda} = A'^{\omega-1}_{\lambda} \\ A'^{\omega-1}_{\lambda} = \pi \end{cases} , \; \lambda = \omega - 1 \\ \sum_{\lambda=1}^{\omega} \begin{cases} \pi = A'^{i}_{\lambda} \\ A'^{i}_{\lambda} = A'^{\omega}_{\lambda} \\ A'^{\omega}_{\lambda} = \pi \end{cases} , \text{ otherwise} \end{cases} \qquad (6)$$

Step 7: After shifting, convert the square matrix B'$[x_i,y_i]$ into a array by performing exchange operation.

Step 8: Drop the padded extra bits and thus obtained is a 64-bit array T_1 which acts as the input to the flow based connectivity part. Totally *n* 64-bit arrays T_1, T_2 … T_n are obtained and then they are passed to the flow connectivity step.

5.2 Flow Based Connectivity step

The rescheduled data acts as the input to this part and it proceeds using the values of code evaluation step. The flow based connectivity step is explained from step 9 to 13 as below:

Step 9: Divide 64-bit input array T_1 into two 32-bit halves α_L and β_R in Round Level 1 (RL_1)

Step 10: Totally *m* rounds takes place between α_L and β_R

for *i* = 1 to *m*, where *m* = 10

$$\alpha_L = F_i[\alpha_L, CL_i]$$
$$\beta_R = P_i[\beta_R, CR_i]$$

Step 11: F and P are the heart of encryption at the left and right side of each round and is explained as follows

Step 11.1: α_L and β_R are divided into four 8-bit parts each α_1, α_2, α_3, α_4 and β_1, β_2, β_3,

β_4 where

$$\alpha_L = \{\alpha_j \mid j \in [1,4]\} \text{ and } \beta_R = \{\beta_j \mid j \in [1,4]\}.$$

Step 11.2: *CL* and *CR* are the intermediate codes generated by the code evaluation function and is explained in code evaluation step.

for j = 1 to 4, i=1(for round 1)

$\alpha_L = (\alpha_j \quad CL_i) \quad \oplus$

$\beta_R = (\beta_j \quad \oplus \quad CR_i)$

Step11.3: Circularly Left Rotate (CLR) the elements of α_L and β_R.

Step11.4: Exchange the elements of α_L to the right side and β_R to the left side.

Step11.5: For next round (i=2), previous round's β_R becomes the value of α_L and previous round's α_L becomes the value of β_R and then this round proceeds as in step 11.

Step 12: Totally *m* rounds takes place for all *n* sets, thus obtaining texts C_1, C_2 … C_n.

Step 13: The text C is the concatenation of C_1, C_2 … C_n. C is added with header at the physical layer and then transmitted through the transmission channel.

5.3 Code Evaluation Step

This section uses a code evaluator function which generates a separate 64-bit code for each round of flow based connectivity step. The working of code evaluator for each round is as follows.

Step 1: Divide a 64-bit input random code into two 32-bit halves left code *CL* and right code *CR*.

Step 2*: CL* and *CR* are again divided equally into four sub-codes each CL_1, CL_2, CL_3, CL_4 and CR_1, CR_2, CR_3, CR_4, where $\{CL_j \mid j \in [1,4]\}$ and $\{CR_j \mid j \in [1,4]\}$.

Step 3: The elements of *CL* and *CR* are circularly left rotated individually

for j = 1 to 4

CL = CLR (CL_j, CL_{j+1})

CR = CLR (CR_j, CR_{j+1}), where CLR stands for circularly left rotate.

Step 4: Exchange *CL* to the right side and *CR* to the left side as a whole and thus obtained is a newly generated 64-bit code. Then the generated code is XOR[ed] with the data in the flow connectivity step.

Step 5: For next round, the generated code is passed to the code evaluator to obtain a new code. In this round, the previous code's *CL* becomes the value of this round's *CR* and similarly, *CR* becomes the value of *CL*.

6. EXPERIMENTAL EVALUATION

A network of 500 x 500 m^2 is used for simulation with a random topology of 100 nodes. Clients and server exchange by communicating 2 KB size of data using TCP protocol. In order to avoid distortion due to network congestion, the size of the message is kept small. Attack probability (*AP*) is a measure or estimation of how the attack will happen. Probabilities are given a value between 0 (0% chance for

jamming to occur) and 1 (100% chance for jamming to occur). Higher the degree of probability, more likely will the jamming be.

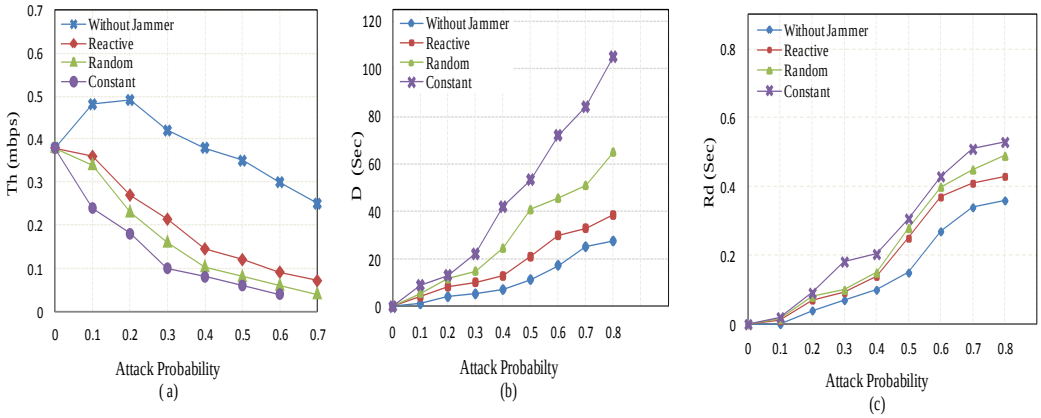

Figure 2(a) Throughput (b) Delay in file transfer (c) Route Discovery Delay under η = 1 with W_j, R_j, C_j and Re_j

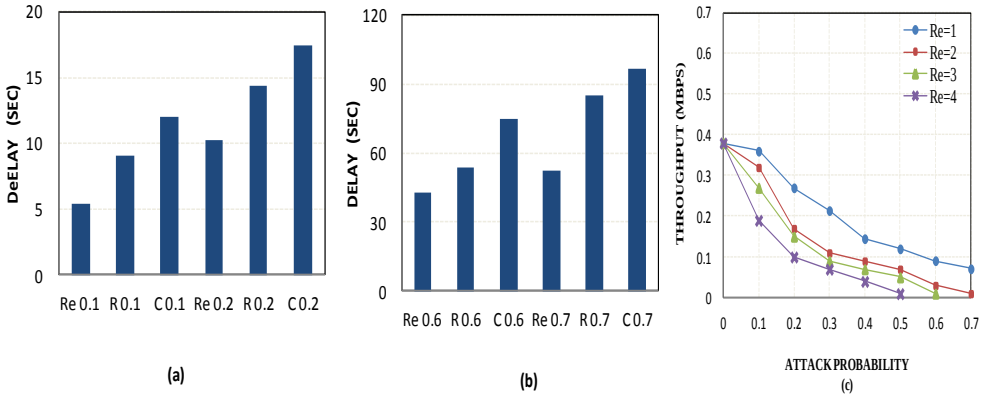

Figure 3 For η = 4 (a) *D* with 0.1 and 0.2 *AP* (b) *D* with 0.6 and 0.7*AP*, R0.1-Random jammer with 0.1 *AP*, C: Constant, Re: Reactive (c) Throughput for different Re_j count

6.1 *Simulation on Different Jammers*

The first set of experiments simulates throughput *Th*, delay *D* in file transfer and route discovery delay R_d as a function of attack probability. It is compared with Without jammer W_j, Random jammer R_j, Constant jammer C_j and Reactive jammer Re_j. Figure 2a, b and c shows the simulation results with lower jammer count $\eta = 1$, where η represents the jammer count. Since Re_j does not degrade the network performance like other jammers, throughput increases while delay and route discovery delay decrease than R_j and C_j and therefore it is hard to detect reactive jammer.

6.2 *Simulation on Varying Attack Probability*

The second set of experiments evaluates D for $\eta = 4$ with W_j, R_j, C_j and Re_j, where η represents the jammer count. Figure 3a shows the delay variation for 0.1 and 0.2 *AP*, whereas figure 3b shows the delay for 0.6 and 0.7 *AP*. Simulation results shows that as *AP* increases, D also gets increased. In Re_j as the probability increases from 0.1 to 0.2, delay also increases from 5.2sec to 10.4sec. Similarly as the probability increases from 0.6 to 0.7, delay increases a lot from 43.2sec to 52.2sec. As *AP* increases, Re_j provides better delay when compared to R_j and C_j. Figure 3c shows throughput for different jammer count as a function of attack probability for reactive jammer. As the jammer count and attack probability increases, throughput starts decreasing and it is very poor for Re_j =4.

6.3 Simulation on Route Discovery Delay

In the third set of experiments, route discovery delay R_d is simulated as a function of higher jammer count $\eta = 4$ for varying attack probabilities. As the attack probability increases, R_d increases and is higher in R_j and C_j when compared to Re_j. Figure 4a shows that as *AP* increases from 0.1 to 0.2 in Re_j, route discovery delay R_d increases from 0.03 to 0.06sec. Similarly as the probability increases from 0.6 to 0.7, R_d again increases from 0.43 to 0.51sec as in Figure 4b. As jammer count and *AP* increases, R_d also increases.

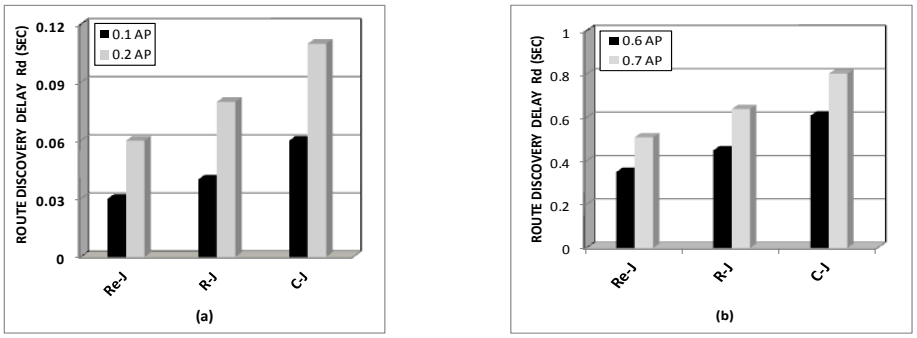

Figure 4 For η = 4 (a) R_d with 0.1 and 0.2 *AP* (b) R_d with 0.6 and 0.7 *AP*.

6.4 Simulation on Varying Jammer Count

In the fourth set of experiment, Table 1 shows the throughput *Th* in mbps, data rate D_r in mbps, delay D in sec and route discovery delay R_d in sec for varying jammer count of 0.1 *AP* for both Re_j and R_j. As Re_j increases from one to four, *Th* decreases from 0.36 to 0.19mbps, while D_r decreases from 0.7 to 0.3mbps and D increases from 4.2 to 5.4sec. As the number of jammers increases, D and R_d start increasing while D_r and *Th* decrease. Re_j is more powerful than R_j in terms of network performance.

Table 1 Parameters for different η with 0.1 *AP*

Parameters	Re_j				R_j			
	1	2	4	6	1	2	4	6
Th	0.36	0.32	0.19	0.09	0.34	0.30	0.27	0.19
D_r	0.7	0.6	0.3	0.1	0.65	0.61	0.5	0.4
D	4.2	4.7	5.4	6.8	5.7	6.4	9.2	11.1
R_d	0.01	0.018	0.03	0.045	0,015	0.023	0.04	0.06

6.5 Simulation on Proposed Scheme

In the fifth set of experiments, *D*, *Th* and R_d are simulated using the proposed CBR scheme, existing PUZ scheme [4], [10] and NO Prevention (NOP) scheme. Figure 5a, b and c shows the simulation results for η = 1 with respect to increasing attack probability by taking an average of 15 TCP connections. PUZ refers to securing the data using puzzles. The result shows that CBR scheme achieves better delay and throughput performance than PUZ. NOP achieves higher delay and lower throughput because there is no prevention for attacks. Figure 5d, e and f shows the variation of *D*, *Th* and R_d and for higher jammer count $\eta = 2$. As the count increases, R_d and *D* increase while *Th* decreases, but the proposed scheme could improve the values of throughput, delay and route discovery delay when compared to other schemes.

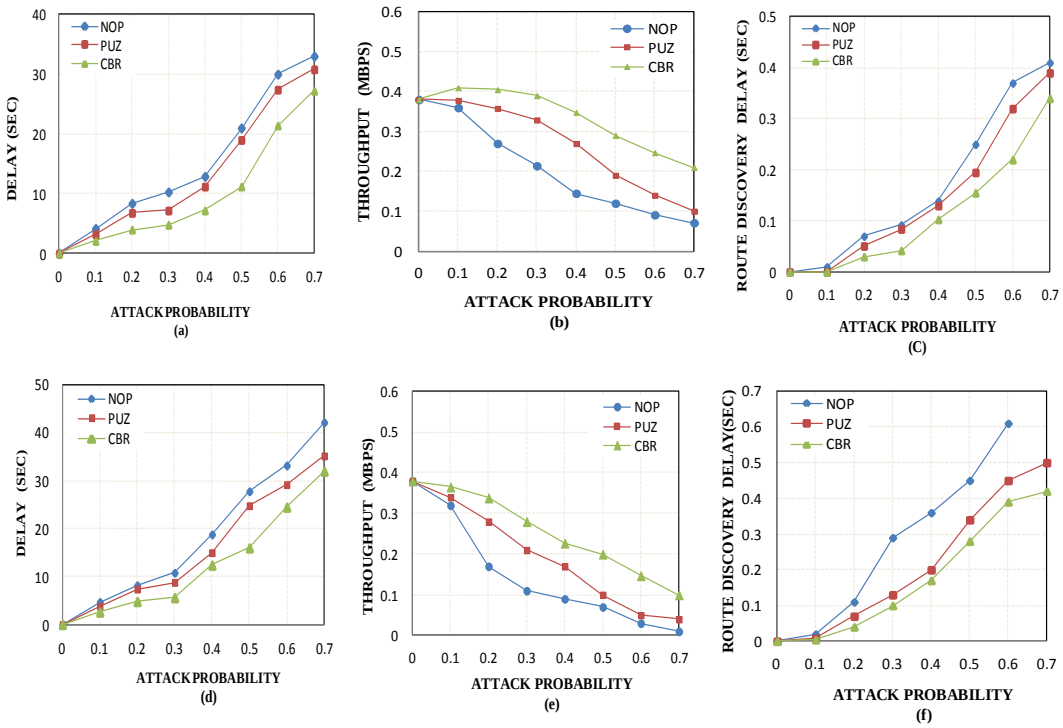

Figure 5 Comparing Delay, Throughput and Route Discovery Delay using CBR, Figure a, b, c for η = 1, Figure d, e, f for η = 2.

Table 2 shows the throughput in mbps for three schemes No Prevention, Puzzling and the Proposed scheme using CBR for lower jammer count η = 1 for varying attack probability. It is noticed that when compared with puzzling, the proposed scheme increases *Th* from 0.378 to 0.41mbps for 0.1 *AP* and from 0.19 to 0.29mbps for 0.5 *AP*.

Table 2 *Th* for three different schemes

Schemes	Attack Probability			
	0.1	0.3	0.5	0.7
No Prevention	0.38	0.214	0.12	0.071
Puzzling	0.378	0.329	0.19	0.1
Proposed	0.41	0.39	0.29	0.21

7. CONCLUSION

This paper addresses the problem of physical layer jamming attack of inside reactive jammer in wireless networks. The Inside jammer is a part of the network and it corrupts the packets by injecting high level of noise. It is experimentally verified that the reactive jammer is more powerful and hard to detect than other jammers by means of throughput, delay and route discovery delay. A new scheme has been proposed for preventing jamming attack based on data rescheduling and data concealing. The proposed CBR scheme first reschedules the data and then modifies it using the code generated by the code evaluator. Then the data is concealed between layers which results in the increase of security level of data. Simulation result shows that CBR yields better performance even when the attack probability and number of jammers increases. It also limits the distorting ability of the jammer. It is experimentally verified that when compared to PUZ scheme, CBR provides higher throughput of 0.41mbps and lower delay of 2.6sec for two jammers with 0.1 attack probability. It is also observed that the proposed scheme also attains higher data rate and lower route discovery delay.

REFERENCES

1. G. T. Amariucai, Physical layer security in wireless networks: Intelligent jamming and Eavesdropping, (May 2009).
2. I. R. Chen, F. Bao, M. J. Chang and J. H. Cho, Dynamic Trust Management for Delay Tolerant Networks and Its Application to Secure Routing, *IEEE Transaction on Parallel and Distributed Systems* **25**(5) (May 2014),1200-1210.
3. E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman and B. Thapa, On the performance of IEEE 802.11 under jamming, *Proc. IEEE INFOCOM* (2008), 1265–1273.
4. A.Juels and J. Brainard, "Client Puzzles: A Cryptographic Countermeasure

against Connection Depletion attacks," Proc. *Network and Distributed System Security Symp*. (NDSS), pp. 151-165, 1999.

5. J. T. Chiang and Y. C. Hu, Cross-Layer Jamming Detection and Mitigation in Wireless Broadcast Networks, *IEEE/ACM Transactions on Networking*, **19**(1) (Feb 2011), 1063-6692.
6. M. Furdek and N. S. Kapov, Attack-Survivable Routing and Wavelength Assignment for high-power jamming, *17th International Conference Optical Network Design and Modeling (ONDM)* (2013), 70 - 75.
7. C. Li, H. Dai, L. Xiao and P. Ning, Communication Efficiency of Anti-Jamming Broadcast in Large-Scale Multi-Channel Wireless Networks, *IEEE Transactions on Signal Processing*, **60**(10) (Oct. 2012), 5281-5292.
8. Pelechrinis K., IIiofotou M., and Krishnamurthy S., "A measurement – Driven Anti-jamming System for 802.11 Networks", *IEEE/ ACM Transactions on Networking*, vol .19, no.4, pp. 1208-1222, 2011.
9. H. Mustafa, X. Zhang, Z. Liu, W. Xu and A. Perrig, Jamming-Resilient Multipath Routing, *IEEE Transactions on Dependable and Secure Computing*, **9**(6) (Dec.2012), 852-864.
10. R. Rivert, A. Shamir and D. Wagner," Time-Lock Puzzles and Timed-Release Crypto," technical report, Massachusetts Inst. of Technology,1996.
11. K. Pelechrinis, M. IIiofotou, and S. V. Krishnamurthy, Denial of service attacks in wireless networks: The case of jammers, *IEEE Communications Surveys & Tutorials* **13**(2) (2011), 245-257.
12. K. Pelechrinis, C. Koufogiannakis and S. V. Krishnamurthy, On the Efficacy of Frequency Hopping in Coping with Jamming Attacks in 802.11 Networks, *IEEE Transactions on wireless communications*, **9**(10) (Oct. 2010), 3258-3271.
13. A. Proano and L. Lazos, Packet-hiding methods for preventing selective jamming attacks, *IEEE Transaction Dependable Secure Computing*, **9**(1) (Jan. 2012), 101–114.
14. A. Richa, C. Scheideler, S. Schmid and J. Zhang, An efficient and fair MAC protocol robust to reactive interference, *IEEE/ACM Transaction on. Networking* **21**(3) (June 2013), 760–771.
15. J. H. Sarker and H.T. Mouftah, Mitigating the effect of jamming signals in wireless ad hoc and sensor networks, *IET Communications* **6**(3) (Feb. 2012), 311–317.
16. Y. S. Shiu, S. Y. Chang, H. C. Wu, S. C. H. Huang and H. H. Chen, Physical layer security in wireless networks: a Tutorial, *IEEE Wireless Communications* **18**(2) (April 2011), 66-74.
17. Popper C., Strasser M., and Capkun S., "Jamming-Resistant Broadcast Communication without Shared Keys," *Proc. USENIX Security Symp*osium, 2009.
18. M. Spuhler, D. Giustiniano, V. Lenders, M. Wilhelm and Jens B. Schmitt, Detection of Reactive Jamming in DSSS-based wireless communications, *IEEE Transactions on wireless communications* **13**(3) (March 2014), 1593-1603.

19. Cagalj M., Capkun S., and Hubaux J.,"Wormhole- Based Anti jamming Techniques in sensor Networks", *IEEE Transactions on Mobile Computing*, vol. 6, no.1, pp.100–114, 2007.
20. G. Thamilarasu, S. Mishra and R. Sridhar, Improving Reliability of Jamming Attack Detection in Ad hoc Networks, *International Journal of Communication Networks & Information* **3**(2011), 57-66.
21. L. Wang and A. Wyglinski, A Combined Approach for Distinguishing Different Types of Jamming Attacks against Wireless Networks, *IEEE Pacific Rim (PacRim) Conference on Communications, Computers and Signal Processing* (2011), 809-814.
22. M. Wilhelm, I. Martinovic, J. Schmitt and V. Lenders, Reactive jamming in wireless networks: How realistic is the threat?, *In Proceedings of ACM conference on Wireless Network Security* (2011), 47-52.