Security Aware Congestion Control Mechanism on SPLIT-TCP over MANETs

Sakshi Bhatia [*], Sanjeev Rana [**], Rajneesh Kumar Gujral [***]

[*] Asst. Professor, Deptt. of Computer Engg., M. .M University Mullana (Ambala), Haryana
[**] Professor, Deptt. of Computer Engg, M. .M University Mullana (Ambala), Haryana
[**] Professor, Deptt. of Computer Engg, M. .M University Mullana (Ambala), Haryana

Abstract- Wireless Ad hoc Networks TCP wrongly attributes packet losses due to the high Bit Error Rate (BER) location-dependent contention, unidirectional links, dynamic topology and the inherent fading properties of the wireless channel to as congestion. It causes an overall degradation of throughput; it especially affects connections with a large number of hops, where link failures are more likely. A number of cross layer solutions such as TCP-F, TCP-ELFN, ATCP, TCP-Bus and SPLIT-TCP has been proposed. Among them Split-TCP is well suited because this scheme converts longer TCP connections to multiple shorter TCP connection, in order to achieve greater Throughput. Another major issue at transport layer is security and few solutions has been proposed so far to provide secure communication and congestion control at the Transport Layer. In this paper we have proposed a security aware congestion control mechanism for MANETs that not only improves performance using SPLIT-TCP but also provide security at Transport Layer.

Index Terms- Split-TCP, ARAN, Congestion Control, Security

I. INTRODUCTION

The objectives of TCP-like Transport layer protocols in MANET include setting up of end-to-end connection, end-to-end reliable delivery of packets, flow control, congestion control, clearing of end-to-end connection. Similar to TCP protocols in the Internet, the mobile node is vulnerable to the classic SYN flooding attack or session hijacking attacks. However, a MANET has a higher channel error rate when compared with wired networks. Because TCP does not have any mechanism to distinguish between whether a loss was caused by congestion, random error, or malicious attacks, TCP multiplicatively decreases its congestion window upon experiencing losses, which degrades network performance significantly [1].

SYN flooding attack: The SYN flooding attack is a denial-of-service attack. The attacker creates a large number of half-opened TCP connections with a victim node, but never completes the handshake to fully open the connection. For two nodes to communicate using TCP, they must first establish a TCP connection using a three-way handshake. The three messages exchanged during the handshake, illustrated in Figure 1, allow both nodes to learn that the other is ready to communicate and to agree on initial sequence numbers for the conversation. During the attack, a malicious node sends a large amount of SYN packets to a victim node, spoofing the return addresses of the SYN packets. The SYNACK packets are sent out from the victim right after it receives the SYN packets from the attacker and then the victim waits for the response of ACK packet. Without any response of ACK packets, the half-open data structure remains in the victim node. If the victim node stores these half-opened connections in a fixed-size table while it awaits the acknowledgement of the three-way handshake, all of these pending connections could overflow the buffer, and the victim node would not be able to accept any other legitimate attempts to open a connection.

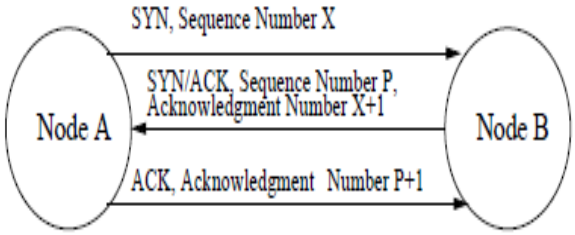

Figure 1: TCP Three-way Handshake

Normally there is a time-out associated with a pending connection, so the half-open connections will eventually expire and the victim node will recover. However, malicious nodes can simply continue sending packets that request new connections faster than the expiration of pending connections [2].

Session hijacking: Session hijacking takes advantage of the fact that most communications are protected (by providing credentials) at session setup, but not thereafter. In the TCP session hijacking attack, the attacker spoofs the victim's IP address, determines the correct sequence number that is expected by the target, and then performs a DoS attack on the victim. Thus the attacker impersonates the victim node and continues the session with the target. The TCP ACK storm problem, illustrated in Figure 2, could be created when an attacker launches a TCP session hijacking attack. The attacker sends injected session data, and node A will acknowledge the receipt of the data by sending an ACK packet to node B. This packet will not contain a sequence number that node B is expecting, so when node B receives this packet, it will try to resynchronize the TCP session with node A by sending it an ACK packet with the sequence number that it is expecting. The cycle goes on and on, and the ACK packets passing back and forth create an ACK storm. Hijacking a session over UDP is the same as over TCP, except that UDP attackers do not have to worry about the overhead of managing sequence numbers and other TCP mechanisms. Since

UDP is connectionless, edging into a session without being detected is much easier than the TCP session attacks. The rest of this paper is organized as follows. Section 2 covers an overview of SPLIT-TCP and ARAN protocols, Section 3 summarizes related work, Section 4 discuss proposed mechanism that degrade congestion and provide security, Section 5, simulation analysis and result discussion is presented and Section 6 concludes this paper with discussions.

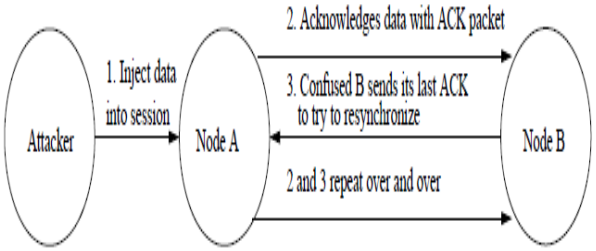

Figure 2: TCP ACK Storm

II. OVERVIEW OF SPLIT-TCP AND ARAN SECURITY PROTOCOL

In this paper, we have proposed a Security Aware Congestion Control Mechanism that consists two modules Split-TCP and ARAN. The Split-TCP is used to reduce delay and to provide security ARAN security protocol has been embed with it.

2.1 SPLIT-TCP

In ad hoc networks, traditional TCP protocol cannot handle node mobility well. Due to mobility of nodes frequent links break, lot of packet losses (until the routing layer discovers a new route). Furthermore, as the number of hops on a path increases, the probability of a link failure on the path increases. This implies that shorter TCP connections enjoy an unfair advantage in throughput as compared with longer connections. So this give birth to new enhanced TCP protocol i.e. Split-TCP. In Split-TCP [3] provides a unique solution to this problem by splitting the transport layer objectives into congestion control and end-to-end reliability. In the ad hoc wireless networks environment, congestion control demands local solutions. At the same time, reliability is an end-to-end requirement and needs end-to-end acknowledgments. Split-TCP splits a long TCP connection into a set of short concatenated TCP connections with a number of selected intermediate nodes (known as proxy nodes) as terminating points of these short connections. Figure 3 illustrates the operation of split-TCP where a three segment split-TCP connection exists between source node 1 and destination node 15. For any TCP connection, [4] certain nodes along the route take up the role of being proxies for that connection.

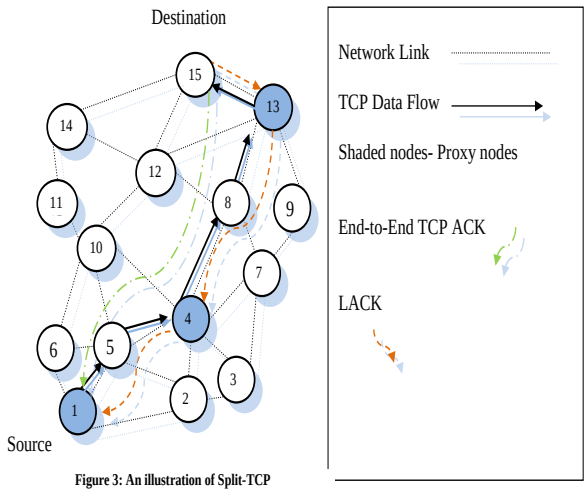

Figure 3: An illustration of Split-TCP

A proxy node receives the TCP packets, reads its contents, stores it in local buffer, and sends an acknowledgement to the source (or the previous proxy). This acknowledgement called local acknowledgement (LACK) does not guarantee end-to-end delivery. The responsibility of further delivery of packets is assigned to the proxy nodes. A proxy node clears a buffered packet once it receives LACK from the immediate successor proxy nodes for that packet. Split-TCP maintains the end-to-end acknowledgement mechanism intact, irrespective of the addition of zone-wise LACKs. The source node clears the buffered packets only after receiving the end-to-end acknowledgement for those packets [5] [6]. In the figure 3 node 1 initiates a TCP session to node 15. Node 4 and node 13 are chosen as proxy nodes. The number of proxy nodes in a TCP session is determined by the length of the path between source and destination nodes. Based on a distributed algorithm, the intermediate nodes that receive TCP packets determine whether to act as a proxy node or just as a simple forwarding node. The simplest algorithm makes the decision for acting as proxy node if the packet has already traversed more than a predetermined number of hops from the last proxy node or the sender of the TCP session. In fig the path between node 1 and node 4 is the first zone, the path between node 4 and 13 is the second zone, and the last zone is between node 13 and 15.The proxy node 4, upon receipt of each TCP packet from source node 1, acknowledges it with a LACK packet, and buffers the received packets. This buffered packet is forwarded to the next proxy node (node 13) at the transmission rate proportional to the arrival of LACKs from the next proxy node or destination. The transmission control window at TCP sender is also split into two windows, i.e. the congestion window and the end-to-end window. The congestion window changes according to the rate of arrival of LACKs from the next proxy node and end-to-end window is updated based on the arrival of end-to-end ACKs. Both these windows are updated as per traditional TCP except that the congestion window should stay within the end-to-end window. In addition to these transmission windows at the TCP sender, every proxy node maintains a congestion window that governs the segment level transmission rate [5]. In TCP-BUS explicit messages such as ICMP source quench are used for

congestion control. ECN is used to notify TCP sender in ATCP, congestion control is same as TCP. In split-TCP [5] since connection is split, the congestion control is handled within a zone by proxy nodes and proxy nodes maintain congestion window and handle congestion.

2.2 Authenticated Routing for Ad hoc Networks (ARAN)

ARAN uses public key cryptography to defeat all identified attacks. It takes care of authentication, message integrity, and non-repudiation, but expects a small amount of prior security coordination among nodes. During the route discovery process of ARAN, the source node broadcasts Route Request packets. The destination node, on receiving the Route Request packets, responds by unicasting back a reply packet on the selected path. The ARAN protocol uses a preliminary cryptographic certification process, followed by an end-to-end route authentication process, which ensures secure route establishment [5].

- **Route Formation Phase:**

Step 1: Each node, before attempting to connect to ad hoc network, must contact the certification authority and request a certificate, which contains the IP address of the node (IP_A), the public key of A (K_{A+}), a timestamp k of when the certificate was created, and a time e at which the certificate expires. These variables are concatenated and signed by K_T. The protocol assumes that each node knows a priori the public key of certification authority.

$$T \to A: cert_A = [IP_A, K_{A+}, t, e]K_T$$

Step 2: The route discovery of the ARAN protocol begins with a node broadcasting a route discovery packet (RDP) to its neighbors. The RDP includes a packet type identifier ("RDP"), the IP address of the destination X (IP_X), A 's certificate (cert $_A$) and a nonce N_A , all signed with A 's private key. Note that the RDP is only signed by the source and not encrypted, so the contents can be viewed publicly. The purpose of the nonce is to uniquely identify an RDP coming from a source. Each time, A, performs route discovery it monotonically increases the nonce. Each node validates the signature with the certificate, updates its routing table with the neighbor from which it received the RDP, signs it, and forwards it to its neighbors after removing the certificate and the signature of the previous node (but not the initiator's signature and certificate). Let B be a neighbor that has received from A the RDP broadcast, which it subsequently rebroadcasts.

$$A \to brdcst: [RDP, IP_X, N_A]K_A -, Cert_A$$

$$B \to brdcst: [[RDP, IP_X, N_A]K_A -]K_B -, Cert_A, Cert_B$$

Upon receiving the RDP B's neighbor C validates the signatures for both the RDP initiator, and B, the neighbor it received the RDP from, using the certificates in the RDP. C then removes B's certificate and signature, records as its predecessor, signs the contents of the message originally broadcast by Y and appends its own certificate C then rebroadcasts the RDP.

$$C \to brdcst: [[RDP, IP_X, N_A]K_A -]K_C -, Cert_A, Cert_C$$

Eventually, the message is received by the destination X, who replies to the first RDP that it receives for a source and a given nonce. This RDP need not have traveled along the path with the least number of hops; the least-hop path may have a higher delay, either legitimately or maliciously manifested. In this case, however, a non-congested, non least-hop path is likely to be preferred to a congested least hop path because of the reduction in delay. Because RDP's do not contain a hop count or specific recorded source route, and because messages are signed at each hop, malicious nodes have no opportunity to redirect traffic. After receiving the RDP, the destination unicasts a Reply (REP) packet back along the reverse path to the source. Let the first node that receives the REP sent by X be node D.

$$X \to D: [REP, IP_A, N_A]K_X -, Cert_X$$

The REP contains the address of the source node, the destination's certificate, a nonce, and the associated timestamp. The destination node signs the REP before transmitting it. The REP is forwarded back to the initiating node by a process similar to the process described for the route discovery, except that the REP is unicasted along the reverse path. Let D's next hop to the source node C.

$$D \to C: [[REP, IP_A, N_A]K_X -]K_D -, Cert_X, Cert_D$$

C validates D's signature on the received message, removes the signature and certificate, then signs the contents of the message and appends its own certificate before unicasting the REP to B. Each node checks the nonce and signature of the previous hop as the REP is returned to the source. When the source receives the REP, it verifies the destination's signature and the nonce returned by the destination.

$$C \to B: [[REP, IP_A, N_A]K_X -]K_C -, Cert_X, Cert_C$$

- **Route maintenance**

When no traffic has occurred on an existing route for that route's lifetime, the route is simply de-activated in the route table. Data received on an inactive route causes nodes to generate an Error (ERR) message. Nodes also use ERR messages to report links in active routes that are broken due to node movement. All ERR messages must be signed. For a route between source A and destination X}, a node B generates the ERR message for its neighbor C as follows:

$$B \to C: [ERR, IP_A, IP_X, N_B]K_B -, Cert_B$$

This message is forwarded along the path toward the source without modification. A nonce ensures that the ERR message is fresh. It is extremely difficult to detect when ERR messages are fabricated for links that are truly active and not broken. However, the signature on the message prevents impersonation and enables non-repudiation. A node that transmits a large number of ERR messages, whether the ERR messages are valid or fabricated, should be avoided.

- **Key Revocation**

In the event that a certificate needs to be revoked, the trusted certificate server, T, sends a broadcast message to the ad hoc group that announces the revocation. Calling the revoked certificate cert X, the transmission appears as:

$$T \to brdcst: [revoke, cert_T]K_T -$$

Any node receiving this message re-broadcasts it to its neighbors. Revocation notices need to be stored until the revoked certificate would have expired normally. Any neighbor of the node with the revoked certificate needs to reform routing as necessary to avoid transmission through the now un trusted node.

III.RELATED WORK

Swastik Kopparty et al. [4], has proposed that for any TCP connection, certain nodes along the route take up the role of being proxies for that connection. The proxies buffer packets upon receipt and administer rate control. The buffering enables dropped packets to be recovered from the most recent proxy. The rate control helps in controlling congestion on inter-proxy segments. Thus, this work concludes that shorter TCP connections achieve better parallelism in the network.

In [7] the main idea behind the proposed mechanism is to notify the sender when the packets of a Transport layer flow change their route. In this work sender can benefit from this information when deciding whether to retransmit a missing segment or to wait, when estimating the RTT (Round Trip Time), and when deciding whether to change the congestion window.

Nizar et. al. [8] suggested the techniques introducing awareness of the physical medium into TCP are typically implemented using different explicit notification techniques. One of the first proposals in this category presented in [9] is Explicit Congestion Notification (ECN). It reserves a specific bit inside the IP header, which brings indication of network congestion back from a router to the sender node. This allows TCP sender to select its congestion control actions differentiating between congestion and link error related losses.

In [10] Sarolahti et. al. proposed explicit signaling algorithm allowing network routers to increase TCP startup performance over high-speed network paths. Having the core algorithms controlling TCP functionality such as congestion control and error recovery implemented at the sender node turns the design of optimization algorithms towards explicit notification solutions, which usually demonstrate considerable performance advantages. However, the main drawback for such solutions is the requirement for the modification of TCP sender code - traditionally implemented inside the operating system kernel, making the deployment of these schemes difficult on the wide scale.

In [8] aims at overhead reduction deriving from the multilayer ARQ employed at the link and transport layers. It introduces ARQ proxy [11],[12] at the base station and ARQ client at the mobile node agents, which substitute the transmission of the TCP ACK packet with a short link layer request sent over the radio link. As a result, ARQ proxy releases radio link resources required for TCP ACK packet transmission - which can be used by other transmitting stations.

In [13] proposed that approaches that rely on explicit feedback from intermediate nodes, like ECN can face problems, since no direct access for the IP header is allowed for such nodes. In order to mitigate such a problem, some effort has to be put on that, but a really robust solution seems to be absent.

Ding et. al. [14] proposed TCP-MANET to detect malicious packet drop attack based on RTT of next acknowledged packet. Upon inferring a malicious attack, TCP-MANET trigger the routing protocol to find a new route to connection, and locate the malicious node in the network.

In [15] to defeat all identified attacks on AODV and DSR using ARAN has been proposed. ARAN can secure routing in environments where nodes are authorized to participate but untrusted to cooperate, as well as environments where participants do not need to be authorized to participate. This work evaluates ARAN and shows that it is able to effectively and efficiently discover secure routes within an ad hoc network.

Jonny Karlssson et. al.[16] proposed that due to heavy asymmetric cryptographic operations and large routing packets, ARAN has a high computational cost for route discovery. ARAN is also vulnerable against selfish nodes e.g. drop routing packets. In particular, if the selfish node is an authenticated node, then ARAN is unable to detect this type of attack.

Kimaya Sanzgiri et. al. [17] proposed ARAN, a routing protocol for ad hoc networks that uses authentication and requires the use of a trusted certificate server. In ARAN, every node that forwards a route discovery or a route reply message must also sign it, (which is very computing power consuming and causes the size of the routing messages to increase at each hop). A proposal that only require originators to sign the message has been proposed in [18]. In addition, it is prone to reply attacks using error messages unless the nodes have time synchronization. Harsh Sadawarti et. al. [19] proposed security model based on ARAN to handle the DoS attacks. All the routing messages are authenticated at every hop from source to destination as well as on reverse path from destination to source.

IV.PROPOSED SECURITY AWARE AND CONGESTION CONTROL MECHANISM

The proposed work provided the security and performance enhancement by controlling the congestion at transport layer. The above said work embeds ARAN over SPLIT-TCP at transport layer that not only prevents congestion but also provide secure data communication in MANET. This work takes the following assumptions:

- The scheme is based on public key cryptography using offline certification authority (CA).
- Proxy nodes are the trusted nodes and know the public key of other proxy nodes.
- The encryption/decryption takes place at the proxy nodes.
- Only proxy nodes can be the source and destination nodes.
- All links are bidirectional.

Each node gets digital certificate from Certifying Authority (CA) in a secure fashion before communication. Since the intermediate nodes will act as only forwarding nodes. All the security checks will be carried out at proxy nodes using ARAN. Let P_1 P_2 be the Proxy Nodes and F_1 , F_2 be the intermediate nodes. Here R_{P1} U_{P1} are the Private and Public key of node P_1 and R_{P2} U_{P2} are the Private and Public key of node P_2. The packets will have to pass through the nodes which can be in an arrangement among the following cases:

Case 1: (Secure communication between two proxy nodes)
In this case at proxy node P_1 the message is encrypted with R_{P1} and further encrypted with public key U_{P2}. At the proxy node P_2 this combination is decrypted with R_{P2} and further decrypted with private U_{P1}.

Figure 4: Secure Communication between two proxy nodes

Case 2: (Secure communication Through Intermediate Node)

In this case a proxy node P_1 knows R_{P1} and U_{P2}. The message is encrypted with R_{P1} and further encrypted with U_{P2}. The next node is an intermediate node F_1 which will only forward the message to the neighbour node. It does not perform any verification and testing.

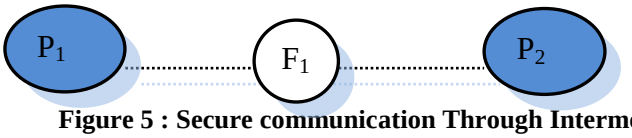

Figure 5 : Secure communication Through Intermediate Node

Case 3: (Communication Between Forwarding Nodes)

The first intermediate node F1 will forward the message to next intermediate node F2 without performing any verification and testing which will also forward the message to the neighbour node.

Figure 6: Communications Between Forwarding Nodes

If the message is tampered proper decryption of the encrypted message cannot take place. If a message is unable to reach the next proxy node in the first attempt then the message is retransmitted. If the message is unable to reach the destination node in three attempts then a negative acknowledgement is sent to the source proxy node. An alternate route is then chosen with minimum number of intermediate nodes using the information that is present in the cache of nodes.

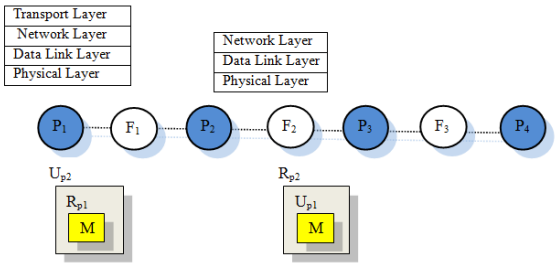

Figure 7: illustrates the operation of SPLIT-TCP

Figure 7 illustrates the operation of split-TCP where a split-TCP connection exists between source node P_1 and destination node P_4. A proxy node receives the TCP packets, reads its contents, stores it in local buffer, encrypts/decrypts the message and sends an acknowledgement to the source (or the previous proxy). This acknowledgement called local acknowledgement (LACK). In the fig the P_1 initiates a TCP session to node P_4. Node P_2 is chosen as next proxy node after the source node. The number of proxy nodes in a TCP session is determined by the length of the path between source and destination nodes. The following mechanism takes place:

Step 1: The node P_1 encrypts the message with R_{P1} and further encrypts with public key U_{P2}. Double encryption takes place at the proxy node and the message is forwarded to intermediate node F_1.

Step 2: Node F_1 does not perform any verification and simply forwards the message to next neighbor node (proxy node P_2).

Step 3: Proxy node P_2, upon receipt of each TCP packet or message from node F_1, carries out decryption with U_{P1} and one more decryption is carried out with $R_{p1.}$ At the first level of decryption authentication, non-repudiation and integrity is achieved. Then at the second level of decryption we are able to achieve secrecy. If proper decryption takes place then proxy node P_2 acknowledges the previous proxy node (P_1) with a LACK packet, and buffers the received packets.

Step 4: The buffered packet is forwarded to next neighbor node which is an intermediate node 2. It forwards the received message to next node which is proxy node (P_3).

Step 5: The process in step1 to step3 is repeated. If proper results are not obtained on decryption of the encrypted message at proxy node (P_2) then the information in the message is tampered.

4.1 Performance Analysis

The network scenario consists of a proxy nodes followed by an intermediate node. The proxy node can be source and destination node. Our simulation environment consists of 5 proxy and 4 intermediate nodes as illustrated in figure 4. For simulation purpose we assume P1 as the source and P3 as the destination. Node P1 sends cipher text by applying double encryption using RSA with its private key for first encryption and RSA with private-public key of next proxy node (P2) for second encryption. The intermediate node 1 receives the data and forwards the cipher text to P2 which carries out decryption using the same algorithm and obtains the original message. Now for transferring the message further, it again encrypts. This process repeats till the message reaches the destination node. The cipher text obtained on first level decryption matches that of the one obtained after first level of encryption which has been obtained using private key encryption. The attacker can only attack at any of the intermediate node since we have assumed all the proxies as trusted nodes. Interruption attacks are launched to deny routing messages from reaching the destination nodes by modifying the message.

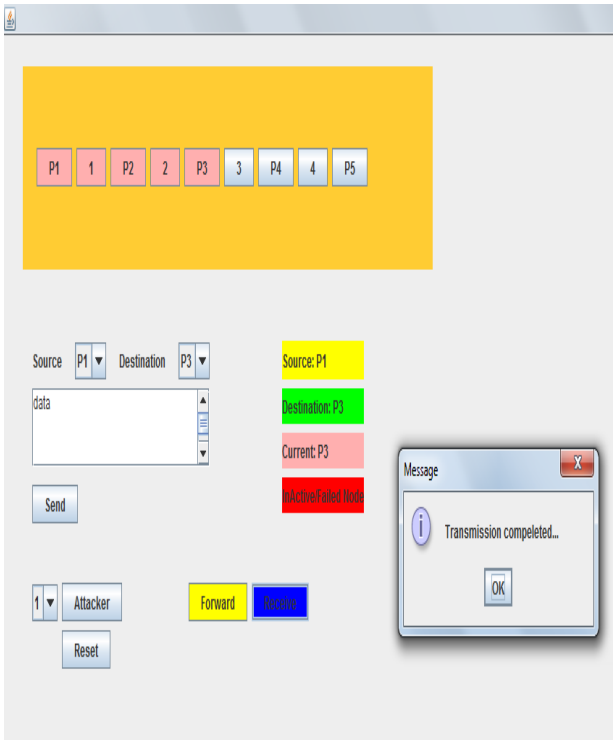

Figure 8: Simulation Scenario (when a between source and destination is complete in graphical mode)

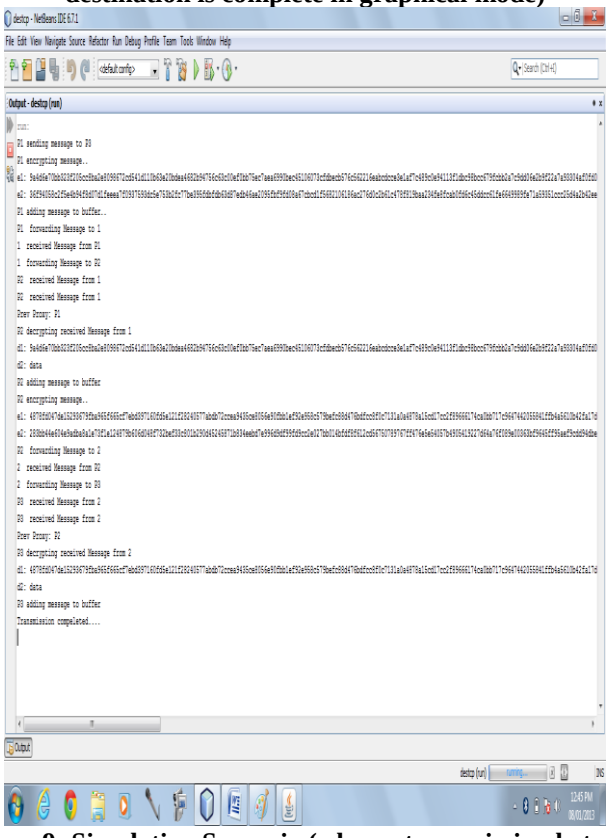

Figure 9: Simulation Scenario (when a transmission between source and destination is complete in text mode)

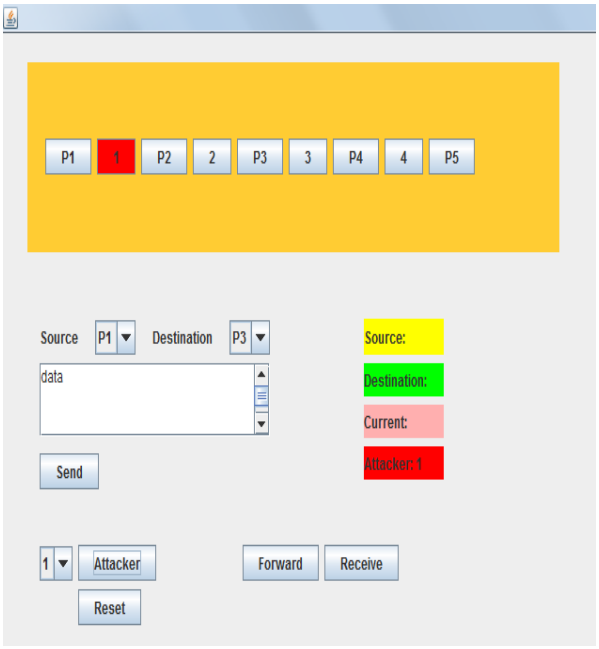

Figure 10: Simulation Scenario to select the attacking node

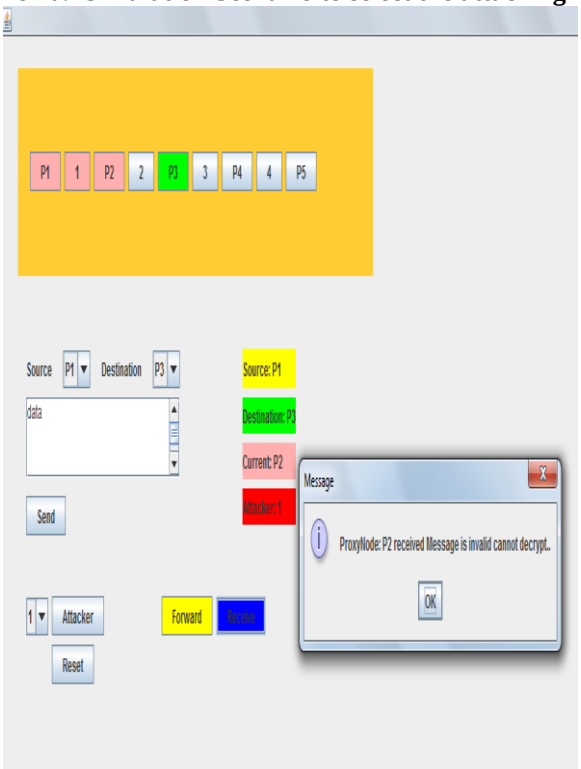

Figure 11: Simulation Scenario (showing transmission in case of attack in graphical mode)

So, if there is an attacking node present in the network then the message is unable to travel further in the network and is thus unable to reach the destination node. In figure 10 and figure 11, node 1 is causing interruption and this is detected by proxy P2 which is unable to decrypt the message since, it has been modified at previous node (node 1).

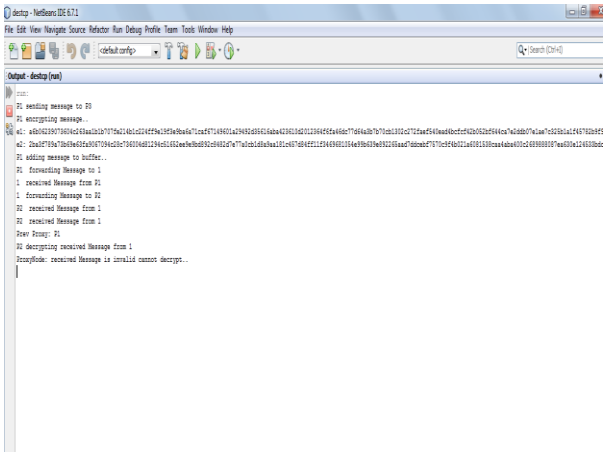

Figure 12: Simulation Scenario showing transmission in case of attack in text mode

V. RESULT AND DISCUSSIONS

The proposed work provide Security and congestion control on SPLIT-TCP. In this work the number of proxy nodes can be obtained from the given equation:

$np = n - (n/2)$ where, n is total number of nodes

The implementation of the network consists 9 nodes in which 4 intermediate nodes and 5 proxy nodes. Following table is used in the graph analysis. It has been assumed that the network topology consists of an alternate combination of proxy and intermediate nodes. T is the time taken to travel from source to destination at the Transport layer.

Table 1: Computational Times of ARAN at Network layer and SPLIT-TCP

Number of Nodes	ARAN at Network Layer	Proposed (ARAN +Split-TCP)
50	50T	25T`
100	100T	50T`
150	150T	75T`
200	200T	100T`

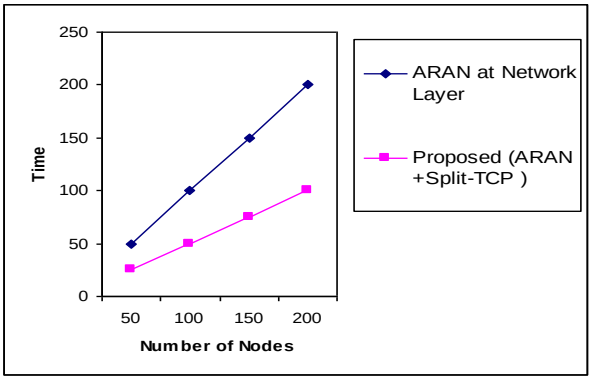

Figure 13: Computational Time of ARAN at Network Layer Vs. Transport Layer

In Figure 13 comparison of computational times of ARAN over routing protocols of network layer and transport layer. As we can see in the figure starting with 50 nodes in the network the computational time taken by ARAN over Network layer is 50T and over Transport layer is 25T, with 100 nodes the computational time taken by ARAN over Network layer is 100T and over Transport layer is 50T, with 150 nodes the computational time taken by ARAN over Network layer is 150T and over Transport layer is 75T and with 200 nodes the computational time taken by ARAN over Network layer is 200T and over Transport layer is 100T.

VI. CONCLUSION AND FUTURE SCOPE OF WORK

The research work embed ARAN on Split-TCP at the Transport layer results in a security aware congestion control mechanism which also reduces delay and thus enhances performance. It is secure since the proxy node does not forward the tampered message to the next node. And it enhances the performance as security checks are not implemented at every node of the network. Instead security is analyzed only at the proxy nodes. As per proposed scheme when the proxy node is unable to carry out decryption successfully then it sends a negative acknowledgement to the source. In the future work, this can be extended by making three attempts for retransmission, then considering this scenario as congestion. The work can also be enhanced by finding an alternate route by using DSR in case source receives a negative acknowledgement thrice.

REFERENCES

[1] H. Hsieh and R. Sivakumar 2002, Transport Over Wireless Networks.Handbook of Wireless Networks and Mobile Computing, Edited by Ivan Stojmenovic. John Wiley and Sons, Inc.

[2] Bing Wu, Jianmin Chen, Jie Wu, Mihaela Cardei 2006 "A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks", WIRELESS/MOBILE NETWORK SECURITY, pp. 2-38.

[3] C.Siva Ram Murthy and G.Mohan, 2001. WDM Optical Networks:Concepts, Design and algorithms, PrenticeHall PTR, New Jersey.

[4] Kopparty, S.; Krishnamurthy, S.V.; Faloutsos, M.; Tripathi, S.K.; 2002 , 'Split TCP for mobile ad hoc networks,' Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE , vol. 1, pp. 138- 142.

[5] C. Siva Ram Murthy and B. S. Manoj, 2004. "Ad Hoc Wireless Networks, Architectures and Protocols", Low Price Edition, Pearson Education.

[6] Swastik Kopparty, Srikanth V. Krishnamurthy, Michalis Faloutsos, Satish K. Tripathi, "Split TCP for Mobile Ad Hoc Networks", Department of Computer Science and Engineering, University of California, Riverside.

[7] Reuven Cohen and Anna Levin, 2009. "A Route-Control Mechanism for Improving the Performance of Transport Protocols in a MANET", IEEE Conference, 34th Local Computer Network, pp-546-553.

[8] Dzmitry Kliazovich , Fabrizio Granelli 2010. ' Why Cross Layer?Advantages and Disadvantages,' University of Trento, Italy,pp:1-33

[9] K. Ramakrishnan, S. Floyd, and D. Black, 2001. The Addition of Explicit Congestion Notification (ECN) to IP.RFC 3168, 2001.

[10] P. Sarolahti and S. Floyd, 2007 "Cross-layer Indications for Transport Protocols," Internet draft draft-sarolahti-tsvwg-crosslayer-00.txt, 2007.

[11] D. Kliazovich, F. Granelli, S. Redana, and N. Riato 2007, "Cross-Layer Error Control Optimization in 3G LTE," IEEE Global Communications Conference (GLOBECOM), Washington, DC, U.S.A.

[12] D. Kliazovich, N. Ben Halima, and F. Granelli, 2007. Cross-Layer Error Recovery Optimization in WiFi Networks in Tyrrhenian International

Workshop on Digital Communication (TIWDC), Ischia island, Naples, Italy.

[13] Wei-Qiang Xu and Tie-Jun Wu, "TCP Issues in Mobile Adhoc Network: Challenges and Solutions" , J. Computer Science and Technology, vol. 21(1), pp.72-81

[14] Ding J,Medidi S R. 2004 "Distinguishing congestion from malicious behaviour in mobile ad-hoc networks" In Proc, Digital Wireless Communications VI, Raghuveer M Rao, Sohail A Dianat, Michael D Zoltowski(eds.), In Proc. SPIE, vol. 5440, Bellingham, WA. USA, pp.193-203.

[15] Kimaya Sanzgiri Daniel LaFlamme Bridget Dahill 2005. "Authenticated Routing for Ad hoc Networks" in IEEE Journal On Selected Areas In Communications: Special issue on Wireless Ad hoc Networks(JSAC),23(3):598-610, March 2005.

[16] Jonny Karlssson ,Laurence S. Dooley ,Goran Pulkkis, 2012. Routing Security in Mobile Ad hoc Networks, Issues in Informing Science and Information Technology vol. 9,

[17] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields and E.M. Royer, 2002. "A Secure Routing Protocol for Ad hoc Networks", Proc. 10th IEEE Int'l. Conf. Network Protocols (ICNP'02), IEEE Press, pp. 78-87.

[18] Jitender Ahlawat, 2012 "SECURE REACTIVE ROUTING PROTOCOLS", International Journal of Research in Science And Technology (IJRST), vol. 1(5) , Apr-Jun 2012

[19] Harsh Sadawarti and Anuj K. Gupta 2009. "Secure Routing Techniques for MANETs", International Journal of Computer Theory and Engineering, vol. 1(4).

AUTHORS

First Author – Sakshi Bhatia, Asst. Professor, Deptt. of Computer Engg., M. .M University Mullana (Ambala), Haryana
Second Author – Sanjeev Rana, Professor, Deptt. of Computer Engg, M. .M University Mullana (Ambala), Haryana
Third Author – Rajneesh Kumar Gujral, Professor, Deptt. of Computer Engg, M. .M University Mullana (Ambala), Haryana