Smart attacks based on control packets vulnerabilities with IEEE 802.11 MAC

Abderrezak Rachedi, Abderrahim Benslimane

Smart Attacks based on Control Packets Vulnerabilities with IEEE 802.11 MAC*

Abderrezak Rachedi and Abderrahim Benslimane
LIA/CERI, University of Avignon, Agroparc
BP 1228, 84911 Avignon, France
Email: {abderrezak.rachedi, abderrahim.benslimane}@univ-avignon.fr

***Abstract*—In this paper, we show new smart attacks which were not dealt with in the solutions proposed recently. We focus on the Medium Access Control (MAC), particularly the IEEE 802.11 and we study some hidden vulnerabilities based on the control packets. The malicious nodes can exploit these vulnerabilities to reduce the network's performance, to disturb the monitoring, routing processes and to escape the Intrusion Detection System (IDS). Furthermore, we show how vulnerabilities can be exploited and how these attacks can be implemented by the attacker. Moreover, attacks' algorithms and the security analysis are presented. We investigate on the effect of these attacks with the simulations and the experimentations. The simulations' results and their analysis illustrate the negative impact of these attacks on the network. In addition, the experimentation results demonstrate the feasibility to real exploitation of these attacks and they confirm the simulation's results.**

I. Introduction

With the growing popularity of MANETs, it is reasonable to expect that users will demand a high security level for them. Many recent works proposed a security solution based on the monitoring mechanisms such as: Watchdog mechanism [5] and others [6] [1]. The monitoring process is defined as the set of actions that are useful to supervise the behaviour of nodes. These actions depend on the services we want to monitor (routing, authentication, integrity, etc). The monitoring process is a part of the Intrusion Detection System (IDS). Therefore, without any IDS of misbehaving nodes, it has been shown that the results of misbehaviour have shown that they dramatically decreased the performance of the network [**?**] and produced the denial of service (DoS). However, the IDS solutions proposed for MANETs [5] [6] are not efficient for the certain hidden vulnerabilities, particularly the cross-layer attacks. These attacks are based on the MAC layer and propagate to the upper layer. Some researchers already studied MAC vulnerabilities and the cross-layer attacks [3] [2], but they only focused on the back-off, DIFS and SIFS manipulations. In this work, we do not focus on the back-off vulnerability, but we show new MAC layer' vulnerabilities based on the packet control format and on the new challenge for IDS that is the detection of these smart attacks.

The contribution of this work consists on the illustration of the new hidden vulnerabilities at the MAC layer, particularly the control packets which have a negative impact on the network's performance. These vulnerabilities are different than the problem of false RTS [4]. However, in our case we focus on the control packets format vulnerabilities and the different attacks which can be exploited against the network and particularly the monitoring process. Furthermore, we discuss the difficulty to detect the attackers and the attacks based on the CTS and ACK packets. Moreover, we show the impact of the attacks on the network and we discuss their effect on the monitoring process based on the forwarding approach (Watchdog mechanism [5]). The attackers' goals can be focused on the network's performance, that means they want to disturb the monitoring process to punish the nodes that well-behave by reducing their reputation. The attacks' implementation and their simulations are presented. In addition, we do not limited to the simulations' result, but the real experimentations are done and the obtained results confirm the simulations' results. Almost all works in the literature propose only analytical analyses and/or simulations rare are those which implement their proposals for a security issues in a testbed. In spite of the problems related to the hardware and system, our initiative to implement our proposals deserves to be introduced into this paper and constitutes a beginning with other experiments.

The rest of the paper is organized as follows. In section 2, we present the hidden vulnerabilities with RTS-CTS handshake protocol: control packets format vulnerabilities, the false CTS and the false packet's validation based on the false ACK are illustrated. In section 3, we show the impact of these attacks on the network with the simulations. The fourth section is devoted to the experimentations' results and to their analysis. In section 5, we propose a security analysis. The last section is the conclusion and we also present our future works.

II. Hidden vulnerabilities

In this section, we illustrate the hidden vulnerabilities at the MAC layer which have a negative impact on the monitoring mechanism. We show how these vulnerabilities may be exploited by the attackers.

A. The RTS-CTS handshake mechanism

The RTS-CTS mechanism is used to avoid the hidden node's problem [7]. The basic idea of the RTS-CTS mechanism consists in transmitting the RTS packet from the sender to the receiver node. When the receiver node receives the RTS packet, it answers by the CTS packet, in order to inform its

* This work is supported by the ANR "Agence Nationale de la Recherche - France" within the project framework ARA/CLADIS.

neighbours about the transmission duration and to avoid the hidden nodes' problem. As illustrated in figure 1(a), node C overhears the RTS packet and defers its own transmissions by the *NAV(RTS)*. So, the NAV (Network Allocator Vector) is the transmission duration which is calculated by sender node A.

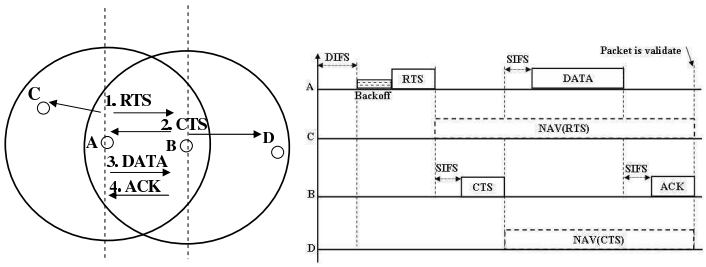
$$NAV(RTS) = 3.T_{SIFS} + T_{CTS} + T_{DATA} + T_{ACK} \quad (1)$$

where T_{CTS}, T_{DATA} and T_{ACK} are the propagation times of transmission packets CTS, DATA and ACK respectively. The T_{SIFS} is the Short InerFrame Spacing [1].

Node D defers its own transmission when it receives the CTS packet with *NAV(CTS)* in the duration packet field, as illustrated in figure 1(a). The *NAV(CTS)* is calculated by receiver node B, and is based on the *NAV(RTS)* in the RTS packet

$$NAV(CTS) = NAV(RTS) - (T_{SIFS} + T_{CTS}) \quad (2)$$

Sender node A does not start the transmission of the data before it receives the CTS packet. The receiver node answers by the ACK packet, once it has received the DATA packet correctly, as indicated in figure 1(b).



(a) (b)

Fig. 1. The RTS-CTS mechanism

B. Control packets' format vulnerabilities

Figure 2 shows the RTS, CTS and ACK packets' format [8]. We notice that, in order to optimize the packets' size, the CTS and ACK packets do not contain the source address of the transmitter node. However, with an RTS/CTS mechanism, when the node sends the RTS packet to the destination node, all nodes in its transmission range become silent except the destination one, that will answer by CTS without its address in the packet. When the sender node receives the CTS packet, it deduces that the packet is from the destination node without checking the address (identity) of the source node, that sends the CTS. In IEEE 802.11, the node cannot communicate with more than one node at the same time [8]. Furthermore, the ACK packet cannot be authenticated by the node which receives it. The malicious nodes can exploit this vulnerability by producing some attacks without being identified by the IDS. The attacker can generate the false RTS in order to block its neighbours. According to the RTS packet format, many solutions may be proposed to detect this attack, like the RTS validation proposed in [4]. However, this solution is limited to the RTS packet and cannot avoid the problem. For example, the attacker can only generate the false CTS and sophisticated false ACK, in order to disturb the network's operation. The scenario of these attacks is explained hereinafter.

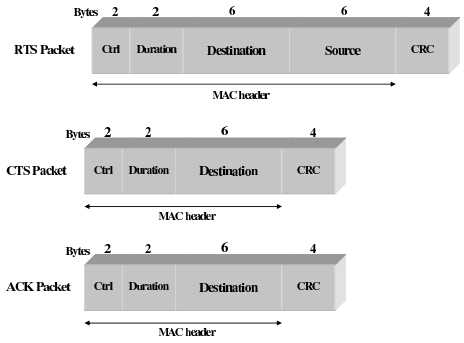


Fig. 2. The RTS, CTS and ACK packets' format

C. Virtual jamming based on the false CTS

The false CTS packet can be generated by the attacker, in order to create a false blocking situation (virtual jamming) [4]. When the nodes in the transmission range of the attacker node receive a false CTS packet, they are futilely blocked during the presumed transmission duration *NAV(CTS)*, even if they did not receive the RTS packet (it considers itself as a hidden node). The false CTS attack can block the nodes in the transmission range of the attacker but it can also block the nodes outside the transmission range of the attacker and inside the interference range of EIFS (Extended Inter-Frame Spaces)[2] [8]. That means that the nodes receive a false CTS but cannot decode it correctly. We can say that the impact of this attack is not limited to the transmission range of the attacker node but is extended to its interference range. Another important problem with this attack is that even if the monitor node detects the false CTS, it is not able to detect the attacker according to the packet format's vulnerability. That means that the attacker can easily escape the punishment procedure, for example, by reducing its trust level or reputation level. That's why the attacker node can easily exploit this vulnerability and frequently attack its neighbors.

In figure 3 we show the false CTS attack algorithm as flow chart. We notice that the attacker checks if the NAV matches zero, then, it will check the channel status. If the channel is busy, it will wait for $DIFS + Backoff$, otherwise it will create the false CTS with a false destination address and start the attack.

D. False packet validation based on the false ACK

The false ACK packet can be exploited by the attacker, in order to disturb the network's operation such as the routing process or the monitoring mechanism, etc. The false ACK attack cannot be detected and the negative impact is more

[1]In IEEE 802.11, the $T_{SIFS} = 10us$

[2]The EIFS is estimated at $364\mu s$ when using a 1 Mbps channel bit rate

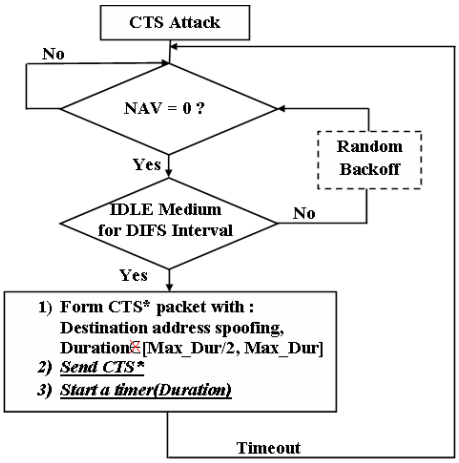


Fig. 3. The false CTS flow chart

significant. The idea consists in validating the packet of the sender although the packet is not received correctly by the receiver node. The impact of this false validation of the packet is that the sender will not retransmit the packet because it received the ACK packet.

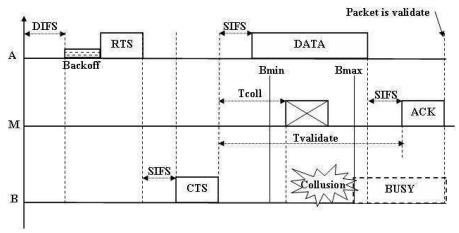


Fig. 4. The false ACK disturbs the monitoring process

Figure 4 illustrates the scenario of the false ACK. Node A wants to send a packet to node B, node M is an attacker located in the transmission ranges of node A and node B. Node M needs to know the address of nodes A and B and also needs to know the NAV(RTS) or NAV(CTS) to create this attack. When node M overhears the RTS packet, it gets the address of nodes A and B and the NAV(RTS). According to the NAV(RTS) or NAV(CTS), node M can determine the time T_{coll} it will start the attack. The attack is divided into two parts. In the first part, the attacker sends the packet to node B at T_{coll}, in order to create a collision at node B. The T_{coll} is the random value in the interval $[B_{min}, B_{max}]$ where

$$\begin{cases} B_{min} = \frac{NAV(CTS)-(T_{ACK}+T_{SIFS})}{2} \\ B_{max} = NAV(CTS) - (T_{ACK} + T_{SIFS} + T_{JAM}) \end{cases}$$

where T_{JAM} is the propagation time of the packet which will create the collision at node B. Generally, this packet has a small size, like the ACK or CTS packets. Then, T_{JAM} is equivalent to T_{ACK} or T_{CTS}. The reason why we chose a random value for T_{coll} is that we wanted to escape any anomaly detection by the monitor node (an IDS). In the second part, the attacker sends the false ACK to node A at the time T_{valide}. This duration is calculated as follows: $T_{valide} = NAV(CTS) - T_{ACK} - T_{SIFS}$.

When sender node A receives the ACK packet before a certain timeout $TOut_2$, it does not realize the presence of any anomaly or false ACK packet. $TOut_2$ is the maximum time between the time when the sender node starts to transmit DATA and the reception of the ACK packet. The $TOut_2$ is calculated as follows:

$$TOut_2 = T_{DATA} + MPD + T_{SIFS} + T_{ACK} + MPD \quad (3)$$

where MPD is the maximum propagation delay [3]. So, the T_{valide} must be less than $TOut_2$, otherwise the sender can detect the problem and retransmit the packet.

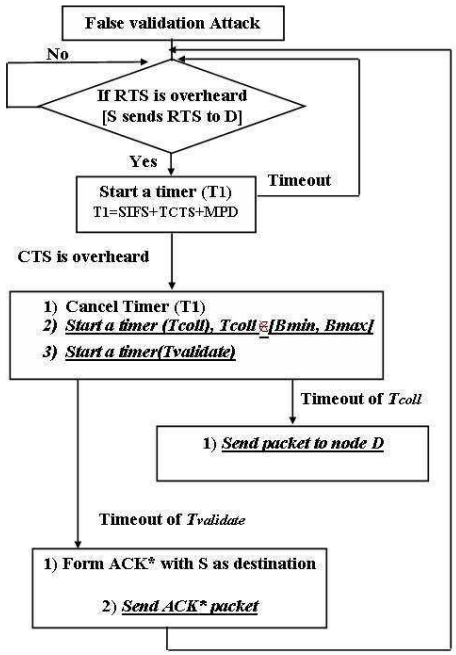


Fig. 5. The false packet validation flow chart

Figure 5 illustrates the flow chart of the false packet's validation based on the false ACK attack. When the attacker overhears the RTS packet, it determines the transmitter's and the receiver's address an then waits for the CTS packet for maximum T_1. T_1 is defined by the T_{SIFS} time, the propagation time of CTS packet (T_{CTS}) and the maximum propagation delay (MPD), $T_1 = T_{SIFS}+T_{CTS}+MPD$. Once the attacker overhears the CTS packet before T_1 timer finished, then, it cancels the T_1's timer and it selects the T_{coll} and $T_{validate}$ as illustrated above. Otherwise, it restarts the attacks's algorithm. Once the T_{coll} timer is finished, the attacker sends the packet to node D in order to create a collision. Then, then the $T_{validate}$ timer is finished the attacker send the false ACK to the sender in order to validate the DATA packet transmission.

[3]In the IEEE 802.11 with the direct sequence spread spectrum (DSSS), the $MPD = 2us$

III. Performance evaluation by simulation and experimentation

A. Attacks' simulation and impact's evaluation

In this section, we investigate on the impact of the false CTS and false ACK (false validation) on the network. We implemented these attacks in an NS2 [12] and we simulated them in different cases.

First, we simulate the simple topology illustrated in figure 6. In this scenario, we have two sets of nodes, $S_1 = \{0, 1, 2, 3\}$ and $S_2 = \{4, 5, 6, 7, 8\}$. In each set, the nodes can hear and communicate directly with nodes belonging to the same set but the nodes in S_1 are not reachable by the nodes in S_2 and vice versa. In both sets, we have two CBR flows (the packet's size equals 1000 bytes and the rate reaches 50 packets/second). However, in set S_1, there is no attacker and in set S_2 ; node 8 reacts as a malicious node and it can use the false CTS and false packet validation's attacks (false ACK).

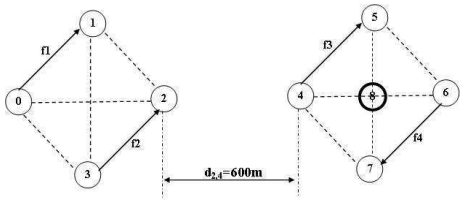


Fig. 6. Simple topology

We chose the throughput as metric, in order to show the impact of these attacks on the network. In figure 7, we plot the average throughput of the network according to the simulation time. In the case of the false ACK, we remark a significant difference between a throughput with no attacker in the network and one with one attacker in the network. However, in the case of a false CTS, we note the same observation as in the first case: the throughput decreases with the attendance of the attacker in the network. However, the false ACK has a more negative impact on the throughput than the false CTS, because the false CTS just creates a blocking situation for a certain time around the attacker node, whereas the false ACK attack creates a collision at the receiver node and then sends the false ACK packet, in order to get the packet validated by the sender node and to avoid the retransmission. In order to compare both attacks, we can say that the false ACK attack is more complex to implement than the false CTS, and the false ACK has a more negative impact on the network than the false CTS.

In order to study the impact of these attacks in the general case, we simulate a network with 50 nodes uniformly and randomly distributed in the area of $800 \times 800m^2$ with 25 CBR connections and with a different number of attacker nodes in the network. Figures 8 and 9 show the throughput in the normal case and in the case of 10 and 20 attackers in the network with both cases false ACK and false CTS. We note that, when the number of attackers increases, the throughput quickly decreases. However, the case of false CTS is illustrated in figure 9, and we remark some difference with the case of

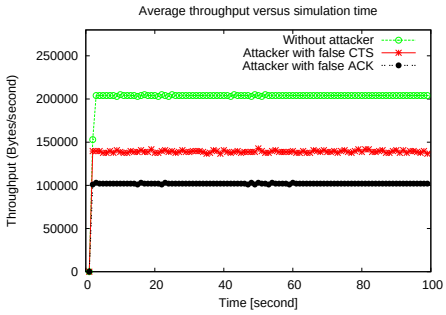


Fig. 7. The average throughput versus time

false ACK: the throughput decreases less rapidly when we introduce 10 and 20 attacker nodes. The difference in the case of 10 and 20 attackers is not significant if we compare it with the case of false ACK. In figure 10, we illustrate the impact

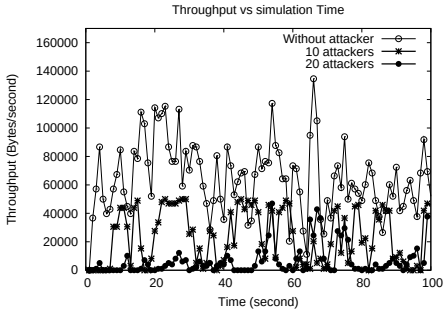


Fig. 8. The throughput versus time with false ACK

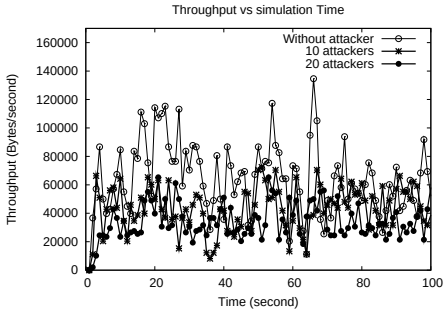


Fig. 9. The throughput versus time with false CTS

of the different numbers of attackers from 0 to 25 malicious nodes (corresponding to 0 to 50%) in the network with a simulation time of 150 seconds. In the case of 5 attackers in the network, we note that the throughput decreases in the same way in both cases false ACK and false CTS. However, when the number of attackers is 10, we note that the throughput of the network decreases more than in the case of false CTS. We have the same observation when the number of attackers is between 15 and 25. We can summarize as follows: the number of false ACK attackers has a more negative impact than the number of false CTS attackers.

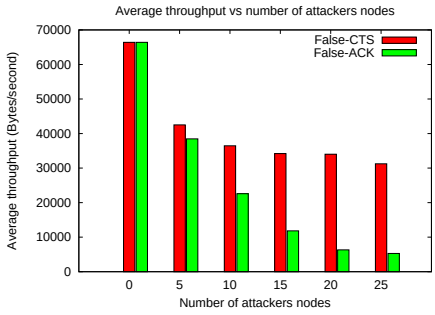


Fig. 10. The impact of the attackers number on throughput

B. Real attacks' experimentation and their impacts

In order to illustrate the feasibility to exploit the described vulnerabilities, we decided to realize these attacks and experimented them. To reach this goal, we experienced the simple attack scenario by using the open source MadWifi drivers (Multiband Atheros Driver for Wireless Fidelity) [9] available for Linux. For compatibility with MadWifi, we used wireless network interface cards based on the Atheros chipsets [10]. For experimentation' equipments, we used the Atheros AR5005G 802.11abg NIC Chipset of TP-Link as wireless network interface and Intel Pentium 4 2.4GHz, 512KB cache L2 and 512MB of RAM memory for attacker and others computers. Furthermore, the CTS and RTS attacks algorithms are implemented on MadWifi. In figure 11, we show the simple experimentation scenario, where the machines A and B want to communicate and M is a malicious machine.

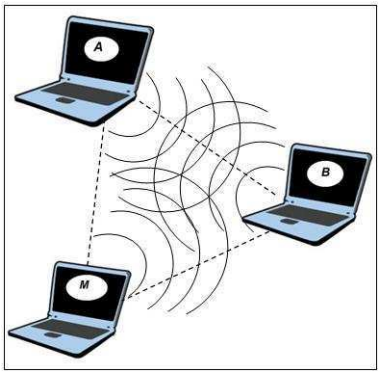


Fig. 11. Experimentation scenario

1) Case of false CTS attack: Node A is repeatedly trying to ping node B (by using or not the RTS/CTS mechanism). The ping process consists in two ICMP messages: a request from A and a reply from B. After a successfully request/reply message from A to B, node M overhears the communication and obtains the transmission duration (NAV), and then, it starts to manage its attack. Once, the channel is idle, it sends a false CTS packet with $NAV(CTS)$ in the duration field as described in CTS attack's algorithm. The $NAV(CTS)$ can be either a constant or a random value, but it cannot exceed $32767\mu Sec$ [8]. Using the analyzer Wireshark [11] to listen and monitor the network traffic, we obtained result which is plotted in figure 12. We notice that after the first successful communication between nodes A and B, node M starts its attacks at 30 seconds and it successfully sends a false CTS. Then, we remark that neither node A nor node B can communicate during false $NAV(CTS)$. The attacker node M continues to transmit the false CTS and dominates the channel by creating the blocking situation. We focus on the period between 30 and 35 seconds and we plot in figure 12 the number of packets according to experience times. Then, we illustrate that the first false CTS packet is transmitted and after false NAV duration, others false CTS are transmitted by the attacker in order to keep the blocking situation as long as possible. When attacker M stops to send the false CTS at 320 seconds, node A can send its packet to node B and the channel's situation is back to normal.

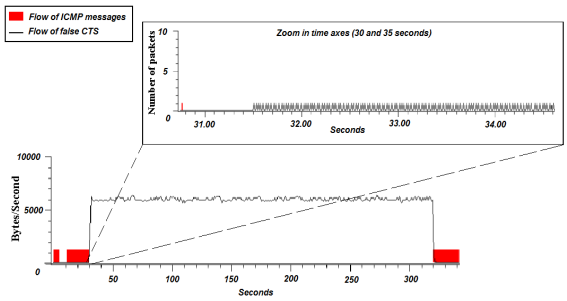


Fig. 12. Impact of false CTS attacks on ICMP message between A and B

In order to study the false CTS attack with TCP communication, we introduced the SSH connection between nodes A and B. Figure 13 shows the result obtained by using the Wireshark analyzer in the case of the SSH connection. The SSH connection is represented by TCP flows in figure 13. We remark that when the attacker manages to have access to the channel and successfully sends the false CTS packet at 39 seconds, then the communication between nodes A and B is broken during the false NAV. Since, the attacker can transmit successfully the false CTS and it continues to disturb the communication between nodes. With these results, we illustrate the efficiency of the false CTS attack and its real negative impact in the network.

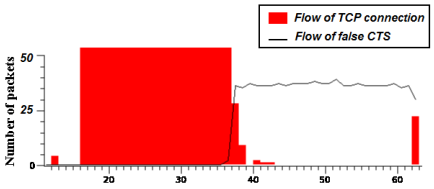


Fig. 13. Impact of false CTS attacks on SSH connection between A and B

2) Case of false RTS attack: In order to illustrate the impact of the false RTS by the experimentation, we implemented this attack with MadWifi and the results obtained are shown in figure 14. We notice that when the attacker success to

transmit the false RTS, it blocks nodes A and B for the random $NAV(RTS)$ chosen by the attacker. At 68.5 seconds node A success to access to the channel but few after node M success to send its false RTS and blocks again the nodes A and B. The attacker M is in competition with nodes A and B to transmit the false RTS. In comparison between the results of false RTS and false CTS attacks, we remark that the false CTS has more negative impact than the false RTS, because the collision with CTS packet is less than the collision with RTS packet. However the CTS packet' size is 38 bytes and it is smaller than the RTS packet's size which is 44 bytes. As conclusion, we can say that the false CTS is more efficient and difficult to detect than the false RTS attack.

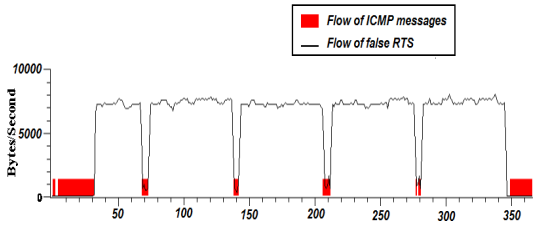


Fig. 14. Impact of false RTS attacks on ICMP message between A and B

According to the experimentation's results, we illustrated the feasibility of implement these attacks and study their real impact on the network. Furthermore, we plan to extend this experimental platform to more complex scenarios.

IV. Security analysis

The design of the detection and the reaction mechanisms against described attacks without modification of IEEE 802.11 standard is a real challenge. The problem is related to the vulnerability of the CTS and ACK packets' format. Furthermore, these attacks have a negative impact on the network. For instance, with a false packet validation, the routing protocol can easily be disturbed: the nodes cannot establish the routing table. Another important negative impact of the false packet validation attack is that it disturbs the monitoring process.

The problem with the false CTS and false ACK, is that the attacker does not need to spoof the source address, because the conception of these packets does not require the source address. In addition, the attacker can easily escape the punishment procedure and continue its attack frequently. Unlike the false RTS attack, the attacker creates a false RTS in order to create the virtual jamming. In this case, the jammer can be detected by the IDS and reduces the effect of its attack. The random RTS validation is proposed by Ashikur [7] to alleviate the impact of this attack. If the attacker chooses the false RTS attack, it needs to spoof the source address in the RTS packet sent by the victim node's address, in order to escape the punishment reaction of the IDS.

Every of false CTS and false ACK can create suspicion situations, each node in the attacker's transmission range suspects other nodes in its transmission range. In order to avoid any false RTS, false CTS and false ACK (or false validation packet), we need to answer the following question. How authenticate and check the integrity of the control packets such as: RTS, CTS and ACK? Many answers and solutions with and without cryptography concept can be proposed to deal with this problem. However, the investigation of the trade-off between the security cost of the secure control packets and the quality of service (QoS) is needful which is the extension of this work.

V. Conclusion

In this paper, we have shown some new hidden vulnerabilities in IEEE 802.11 and the possible attacks that could exploit them. These vulnerabilities at the MAC layer are analysed, like the false CTS and false packet validation attacks which exploit the CTS and ACK packets format vulnerability. The negative impacts of these attacks on the network are illustrated by the analytical, simulation's and experimentation's results. Another important negative effect of the false validation packet based on the false ACK attack affects not only the the network performance but also the monitoring process. The attacker can easily reduce the reputation of well-behavioring nodes and disturb the existing trust model. Furthermore, the routing operation can be dramatically affected by the false ACK attack or false packet validation. The real implementation of these attacks particularly false RTS and false CTS demonstrate the feasibility to exploit the vulnerabilities on IEEE 802.11.

As future work, we plan to study some efficient solutions with and without cryptography concept and their security cost estimation including energy consumption and throughput.

References

[1] A. Rachedi and A. Benslimane and Lei Guang and Chadi Assi, A Confident Community to Secure Mobile Ad-Hoc Networks. *In the IEEE International Conference on Communications (ICC 2007), Glasgow, Scotland, UK*, 2007

[2] L. Guang, C. Assi, A. Benslimane On MAC Layer Misbehavior in Wireless Networks: Challenges and Solutions, *IEEE Wireless Communication Magazine (Accepted, April 2007)*

[3] S. Radosavac, N. Benammar, J. S. Baras Cross-layer attacks in wireless ad hoc networks *Conference on Information Sciences and Systems*, 2004

[4] S. Ray, D. Starobinski On False Blocking in RTS/CTS-Based Multihop Wireless Networks *Vehicular Technology, IEEE Transactions*, Volume 56, PP. 849-862, 2007

[5] S. Marti, T.J. Giuli, K. Lai and M. Baker, *Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*, In Proceedings of the Sixth annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 255–265, 2000.

[6] Yongguang Zhang and Wenke Lee, *Intrusion detection in wireless ad-hoc networks*, In Proceedings of the ACM MobiCom'2000, pages 275-283, 2000.

[7] Rahman, A. Gburzynski, P., *Hidden Problems with the Hidden Node Problem*, in 23rd Biennial Symposium on Communications, pages 270-273, 2006.

[8] The editors of IEEE 802.11, *Wireless LAN Medium Access Control(MAC) and Physical Layer (PHY) specification*, 1997.

[9] *http://madwifi.org/*

[10] *http://www.atheros.com/*

[11] *http://www.wireshark.org/*

[12] UC Berkeley and USC ISI, *The network simulator ns-2*, Part of the VINT project. Available from http://www.isi.edu/nsnam/ns, 1998.