# Security Implications of Cooperative Communications in Wireless Networks

Salik Makda†, Ankur Choudhary⋆, Naveen Raman†, Thanasis Korakis†, Zhifeng Tao◇, Shivendra Panwar†

† Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, NY 11201
⋆ Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India
◇ Mitsubishi Electric Research Laboratories, Cambridge, MA 02139

e-mail: smagda@poly.edu, ankurc@iitk.ac.in, nraman01@students.poly.edu, korakis@poly.edu, tao@merl.com, panwar@catt.poly.edu

***Abstract*—Cooperative communications is an innovative technique that is expected to change the behavior of wireless networks in the near feature. In the MAC layer, this technique defines new protocols by enabling additional collaboration from stations that otherwise will not directly participate in the transmission. A typical example of such a protocol is CoopMAC [1], a cooperative MAC protocol that involves an intermediate station or helper in the communication between a transmitter and a receiver. Under this scheme, the transmitter sends its packets to the receiver by forwarding them through the helper. In this way the protocol takes advantage of spatial diversity and faster two-hop transmission, significantly improving the performance of the network. In such an environment, where the sender relies on an intermediate helper to forward its packets to the original destination, numerous security issues may arise. The present security schemes need to be adapted in order to support end-to-end security in the source-helper-destination communication model. In this paper we discuss the potential security issues that cooperation may raise and propose two new security schemes to address those concerns. To evaluate the feasibility of the proposed algorithms, we implement them using open source drivers platform, which is explained in the paper in detail. Moreover, the paper also discusses the design challenges encounterd and share the experience and insights gained during implementation. Our implementations of the suggested techniques allow the WEP, WPA and WPA2 (802.11i) security protocols to successfully operate in the new cooperative environment.**

***Index Terms*—Wireless security, cooperative communications, cooperative MAC protocols, open source implementation**

## I. Introduction

Cooperative communications consists of schemes and techniques that take advantage of spatial diversity among neighboring stations in a wireless network. These schemes, by enabling additional collaboration from stations that otherwise will not directly participate in the transmission, achieve tremendous improvement in the overall performance of the network. The innovation of cooperative communications was initially confined to the physical layer. However, in order for the network to take advantage of cooperative diversity, new higher layer protocols should be devised accordingly. Towards this end, several MAC layer protocols have been proposed to take advantage of the notion of cooperation. A typical example is a protocol called CoopMAC [1], which illustrates how the legacy IEEE 802.11 [2] can be enhanced with minimal modifications to maximize the benefits of cooperative diversity. Under this protocol, a station that experiences a bad channel with an intended receiver, instead of transmitting directly to a receiver at a low transmission rate, can use an intermediate station (usually called a helper) to forward its packets to the receiver. Due to the fact that the helper experiences a good channel with both stations involved in the communication, the two hop transmission is done at high rates. Thus the overal communication consumes significantly less time than the original one. As a result, CoopMAC can substantially improve the performance of the network. The interested reader can find more information about the details of the protocol in [1].

An implementation of the cooperative MAC scheme in the *HostAP* Linux wireless driver has been discussed in [3]. Although cooperative communications enables high performance in wireless networks, its reliance on the premises that some third-party station has to be involved in the communication raises serious concern of potential security compromise. More specifically, the fact that the helper receives the packet and retransmits it to the receiver has the potential of opening holes in the security of the network and hence must be thoroughly investigated.

In this paper we first study the potential security issues that may arise in such a cooperative network. We then propose two new schemes that adapt today's security techniques (i.e., WEP, WPA, WPA2-802.11i) [2] [4] to the new cooperative environment. In order to prove the feasibility of the proposed schemes we implemented them in MADWIFI version number 0.9.2, which is the most recent version of an open source Linux driver platform for Atheros chipsets. In the paper, we describe these efforts as well as the challenges we have faced and the experience gained during this process.

The rest of the paper is organized as follows: In section II we briefly introduce cooperative MAC protocols and enumerate the possible security concerns that the protocol may raise. In section III we provide a brief overview of the 802.11i security framework. In section IV we propose the solutions to the security problems described earlier in section II. In section V we discuss the details of the implementation of the proposed security schemes. Finally, we conclude in section VI by highlighting the experience and insights we have gained during the implementation process.

## II. Security concerns in cooperative MAC protocols

### A. CoopMAC Description

In this section we describe potential security issues that a cooperative MAC protocol can introduce. We were initially inspired to study these issues by observing the security implications of CoopMAC. However, most of the points in the following discussion can apply to any cooperative MAC protocol that uses a relay station to forward packets from a transmitter to a receiver.

In order to make the reader familiar to CoopMAC we give here a brief description. The interested reader can find the details of the protocol in [1], [3].

For this description we assume stations with IEEE 802.11b interfaces. Under the specific technology a station can transmit using 1, 2, 5.5 and 11 Mbps, based on the channel quality. The scheme can be adapted in any wireless technology that offers multirate capability for packet transmission.

Consider a transmitter relaying data to a receiver (Figure 1). Due to bad channel condition, the transmitter is not able to sustain high data rate with the receiver. Thus it transmits in a low rate (1 Mbps in the figure). We modify this scheme by introducing another node called helper. The helper is a station that belongs to the same wireless network and is capable of supporting a higher data rate with both the transmitter and receiver. In the new scheme, the transmitter instead of transmitting the packets directly to the receiver through a slow hop (1Mbps), it uses cooperation and it forwards the packets through the helper by using two fast hops: one from itself to the helper (11 Mbps in the figure) and the second one from the helper to the receiver (5.5 Mbps in the figure). Using this MAC layer scheme, the network gains benefits from spacial diversity due to the existence of the helper in the communication between the transmitter and the receiver. Simulation results as well as experiments in a real implementation [1], [3] show that the proposed scheme boosts the performance of the network up to 5 times comparing is with the existing technology of IEEE 802.11.

### B. Security Issues

The first potential security issue in the CoopMAC protocol is that of the helper deliberately not forwarding frames received from the source. In this case the helper could deny service to the source by simply dropping the packets it receives. It would then be up to the source to realize that this helper is unresponsive and choose another helper. If another helper does not exist, it can then transmit directly to the destination, albeit at a lower rate. The source could detect the responsiveness of the helper or lack thereof, by imposing some kind of a timeout, after which if no acknowledgment from receiver is received, it would blacklist the helper and try to retransmit via a different helper, if available, or directly.

The second potential issue is more serious. The malicious helper may try to deny service to the source by failing to forward data and spoofing an ACK on behalf of the destination, thereby making the source think that the data was received. Here, we may try to combat this problem via the aid

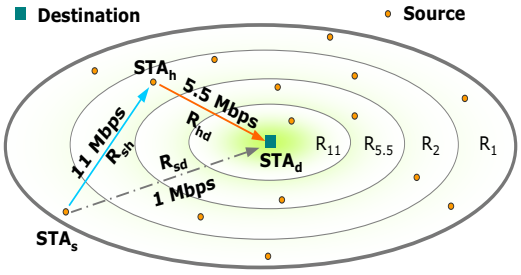


Fig. 1: Illustration of the Cooperative MAC Protocol.

of RTS/CTS. COOPMAC uses some variant of the RTS/CTS scheme. This means that the destination sends the CTS, and is aware that it is an intended recipient of a future frame. Thus, if it does not receive this frame in the allocated NAV period, (due to the fact that the helper did not send it and spoofed an ACK to the source) it can send a NACK or negative acknowledgment to the source, alerting the source that it did not receive the frame. Alternatively, the transmitter can simply listen to the second hop transmission, and if the correct second hop packet is not heard, then blacklist the helper to avoid using it in the future.

The third and the most important potential security issue is a scenario where the helper modifies the payload and then forwards it. The receiver will typically not come to know of this, so it will think that it is only communicating with the genuine sender and may end up voluntarily with replying with privileged information, such as username and passwords. This type of an attack is possible when changes made in the payload will not lead to corruption of the packet, i.e. when no wireless encryption scheme is used or if the WEP scheme is used. If no wireless encryption scheme is employed, then it is obvious that no mechanism exists to detect the alteration of the payload. The modification of payload may also work without corrupting the packet when WEP is used and there is a single shared key [5]. Such an attack cannot be easily avoided unless the transmitter and receiver can themselves find that there is an unusually large delay in the received packets, which will be due to calculations of CRC ,etc, at the helper. At that point they may choose to use some other helper. However if we implement CoopMAC according to the protocol which requires the retransmission of the packet by the helper in a SIFS interval this type of attack will not be possible as the SIFS duration is very small to perform any kind of complex calculations and manipulation of the packet. Finally, 802.11i security protocols are not vulnerable to the modification of payload unless the exact key is known to the helper.

## III. An overview of 802.11i

IEEE 802.11i, also known as WPA2, is an amendment to the 802.11 standard specifying security mechanisms for wireless networks. When Wired Equivalent Privacy (WEP) was shown to have severe security weaknesses [5], Wi-Fi Protected Access (WPA) was introduced as an intermediate solution to WEP insecurities and implemented a subset of 802.11i. 802.11i makes use of the Advanced Encryption Standard (AES) block cipher whereas WEP and WPA use

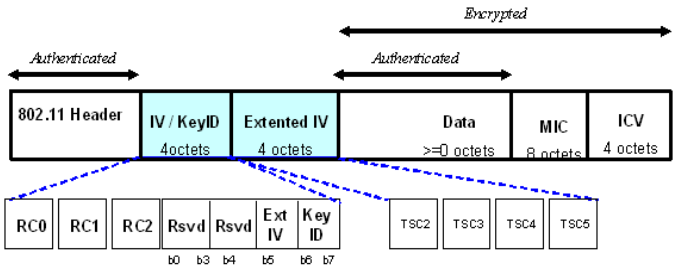


Fig. 2: TKIP MPDU

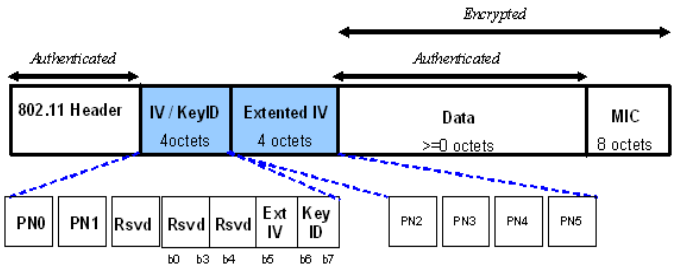


Fig. 3: CCMP MPDU

the RC4 stream cipher. The 802.11i architecture contains the following components: 802.1X for authentication, Robust Secure Network(RSN) for keeping track of associations, and AES-based Counter Mode with Cipher Block Chaining Message Authentication Code(CCMP) to provide confidentiality, integrity and origin authentication.

Like WPA, 802.11i has a pre-shared key mode (PSK), designed for home and small office networks that cannot afford the cost and complexity of an 802.1X authentication server. Each user must enter a passphrase to access the network. The passphrase is typically stored on the user's computer, so it need only be entered once.

### *A. Security protocols used in 802.11i*

**TKIP: Temporal Key Integrity Protocol** is a security protocol used in Wi-Fi Protected Access (WPA). WPA was introduced to patch up the deficiencies in the older Wired Equivalent Privacy (WEP) standard, hence TKIP was designed to replace WEP without replacing the legacy hardware. This was necessary because the vulnerabilities of WEP had left WiFi networks without viable link-layer security, and the solution to this problem could not wait for the replacement of deployed hardware. For this reason, TKIP, like WEP, uses a key scheme based on RC4, but unlike WEP, TKIP provides per-packet key mixing, a message integrity check and a rekeying mechanism. TKIP ensures that every data packet is sent with its own unique encryption key. Key mixing increases the complexity of decoding the keys by giving the attacker much less data than if it had been encrypted using any one key. The message integrity check prevents forged packets from being accepted. Under WEP it was possible to alter a packet whose content was known even without decrypting it. Also TKIP hashes the initialization vector (IV) values, which are sent as plaintext, with the WPA key to form the RC4 traffic key, addressing one of WEP's largest security weaknesses. WEP simply concatenated its key with the IV to form the traffic key.

**CCMP: Counter Mode with Cipher Block Chaining Message Authentication Code Protocol** is an IEEE 802.11i encryption protocol, created to replace, together with TKIP, an earlier, insecure WEP protocol. CCMP uses the Advanced Encryption Standard (AES) algorithm. In the CCMP, unlike TKIP, key management and message integrity is handled by a single component built around AES. Data is encrypted using Counter (CTR) mode AES. Authentication is achieved by using a Cipher Block Chaining Message Authentication Code (CBC-MAC). This combination of CTR and CBC-MAC is what constitutes CCMP. CCMP encapsulations attempt to ensure the confidentiality and integrity of the communications channel, and to prevent replay attacks. Integrity is assured by calculating a MIC (Message Integrity Code) sum to check if a message is altered, protecting data from replay attacks.

## IV. Security in presence of COOPMAC

In order to implement CoopMAC we need to be able to modify without corruption the IEEE 802.11 header of the packet at the helper (transmitter, receiver MAC addresses) for the helper-to-destination transmission. Thus, the current approach to implement CoopMAC will not be compatible with 802.11i [4]. In both TKIP and AES mode, the integrity check covers the MAC header of the packet in addition to its payload. This check calculates a message integrity check (MIC) over the source and destination address as well as the MSDU plaintext data. Thus, if the helper changes anything in the header, the integrity check at the receiver will fail and the packet will be discarded. No ACK will be issued, so the source will try to retransmit. After a few unsuccessful retransmissions, the transmitter will then blacklist this helper to avoid using it in the future which is not desirable. Hence in order to make 802.11i work in a cooperative network we need to make some modifications to the protocol in terms of header management so as to support its encryption and authentication mechanisms.

After a careful study of 802.11i and CoopMAC implementation, we propose two possible solutions in order to make CoopMAC compatible with the IEEE 802.11i architecture:

### *A. Two header format scheme*

*Transmitter*

- In this method before any authentication or encryption is performed, the transmitter, if using cooperation, selects an appropriate helper.
- The transmitter first prepares the second hop packet which will be transmitted from the helper to the destination. This packet is the same as what a direct source to destination transmission packet would have been, and is encrypted and authenticated with the key shared between the source and destination.
- Now this entire MAC level packet with its second hop header is treated as payload and again encapsulated and


Fig. 4: Two header format scheme

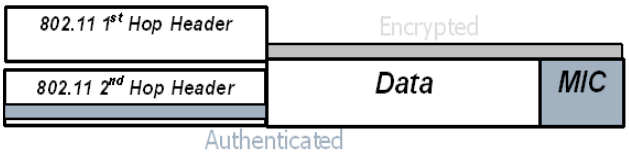


Fig. 5: Single header format scheme

encrypted with respect to the first hop transmission i.e. with the helper being the destination (hence the key is the one which is shared between the source and helper).
- This doubly encapsulated and encrypted packet is now transmitted. Thus this mechanism secures both the source-helper and helper-destination links.

*Helper*

- Identifies if the received packet is using cooperation and is the first hop.
- If it is first hop then remove out the first hop header and transmit the remainder of the packet to the intended destination with no modifications at all. Here helper cannot modify the packet payload without risking its corruption, as it is still encrypted with the 802.11i key shared between the source and destination.
- Else if own packet then accept.

*Receiver*

- Receives the packet and perform decryption.
- Calculates MIC for the packet and compares with the original calculated MIC in the packet.
- Now as there has been no modification to the parts of packet used in the calculation of original MIC, the packet will successfully clear this integrity check.
- Thus the authenticity and privacy of the packet is ensured.

The two header format increases the transmission overhead by one 802.11 header per packet. This overhead can be removed by using the single header format.

### *B. Single header format scheme*

*Transmitter*

- Before any authentication or encryption is performed, the transmitter, if using cooperation, will select the appropriate helper.
- The transmitter again first prepares the second hop packet which will be transmitted from the helper to the destination. This packet is same as what a direct source to destination transmission packet would have been, and is encrypted and authenticated with the key shared between the source and destination.
- Now the Transmitter replaces this second header with the first hop header, and without any further encryption or authentication transmits the packet. It can be noted that the header is still in plaintext. Hence, right now the packet is a corrupt packet because of its incorrect header but the correct header is known to the helper.

*Helper*

- Identify if the received packet is using cooperation and is the first hop. This check should be done before the driver attempts to decrypt the packet after its reception.
- If it is first hop then the helper simply modifies the plaintext header with second hop source and destination fields and transmits the packet. Now the header is the same as that for which the source calculated the MIC.
- Else if own packet, then accept.

*Receiver*

- Receive the packet and perform decryption.
- Calculate MIC of the packet and compare with the original calculated MIC in the packet.
- Now as there has been no modification to the parts of packet used in calculation of original MIC it will successfully clear this integrity check.
- Thus the authenticity and privacy of the packet is ensured.

## V. Implementation of the Schemes

In order to prove the feasibility of the proposed schemes and to test them in a real environment, we decided to implement them. For the implementation we used a Linux platform based on the open source MADWiFi driver [6] for Atheros chipsets. The implementations for both the schemes follow the respective steps outlined in the preceding section but there were certain non trival design choices and challenges that we had to encounter. In order to fully understand and appreciate them, we shall first provide a brief overview of how security is implemented in the MADWiFi driver.

### *A. Security implementation in the Open Source MADWiFi Driver*

Security in MADWiFi can be viewed from two perspectives: a user's and a developer's perspective. As a user, we are interested in being able to use the features of the driver to set up a secure channel for our communication. MADWiFi implements WEP and also the WPA and WPA2 cipher suites: TKIP and CCMP. However, as of now, in built authenticator and supplicant support for only WEP is provided for both ad-hoc and infrastructure mode setups. The use of WPA and WPA2 is more complicated. In order for these schemes to be used in infrastructure mode, the WPA supplicant process should run on the client and a host-ap authenticator daemon should be set up at the AP. Using them in the ad-hoc mode is a bit trickier though. One would have to run both the WPA supplicant and host-ap daemon processes on the nodes; even then multiple authentications are probably not possible.

Now, let us take a look at security from the more interesting angle- that of a developer's. The trace of a packet as it passes through the MAC layer is of key importance in understanding the functioning of security in a driver and subsequently modifying it. While the packet to be sent is being encapsulated with the MAC header, then depending on the destination address, the corresponding unicast/multicast key is retrieved and the header size is allocated according to the type of the key (WEP, TKIP or CCMP). Then, depending on the security protocol being used, a Message Integrity Code (MIC) is added to the packet which covers the payload as well as the source and destination addresses. There is no MIC in the WEP protocol. Now, depending on the packet size and fragmentation threshold, the encrypted packet is fragmented into one or more parts, which are followed by their encryption bits before they are finally transmitted. At the receiver's side, the reverse order is followed. The shared key corresponding to the sender is retrieved and the received packet/fragment is decrypted using it. Then the packet(s) are defragmented followed by MIC checks depending on the protocol being used. Finally if the packet passes the integrity check, the MAC header is removed and the packet is pushed to the upper layer. There is one more thing to be noted before closing this discussion. MADWiFi (as some other drivers) can use hardware encryption for greater efficiency. The type of encryption is determined by a flag in the key itself, so it can be set to enable software encryption during development.

### *B. Design Challenges and Decisions*

As noted in the preceding section, setting up a three node ad-hoc WPA/WPA2 security enabled network is not currently supported by the MADWiFi driver. Thus, we decided to hack around the available infrastructure BSS mode security features to illustrate the implementation of our security schemes. It is imperative to note here that the correctness of our solution does not depend on the stations being in ad-hoc or infrastructure BSS mode. The same implementation with a few minor changes can be used to secure co-operative ad-hoc networks once the full fledged security enabled MADWiFi driver is rolled out.

Since we decided to use the infrastructure mode to implement the security schemes, the entire CoopMAC had to be written for the infrastructure mode. However, before starting the implementation of the protocol, a major design decision had to be made: Who should be the AP in our sender-helper-receiver set-up? This choice would not only determine the form of CoopMAC implementation, but also it would had a critical role on whether we would be able to implement the protocol at all, a fact that we realized only later.

Initially, we were driver to this desition by some implementation limitations in MadWiFi: In the infrastructure mode, only the AP can directly communicate to more than one node. Thus, the helper was chosen as the AP since the helper had to be able to communicate directly with both the sender and the receiver. We implemented CoopMAC with this design choice but soon we realized that this would not work. Under this scheme, sender and receiver both being STAs did not have a direct link between them and hence they did not have a shared key. The sender had only one key which was for its link to the helper (AP). Hence, it could neither perform the two headers double encryption, which required both sender-helper and sender-receiver shared keys, nor the single header encryption which required the sender-receiver shared key.

So the AP clearly had to be the sender and CoopMAC was again implemented with the new design choice. Now the sender had both sender-helper and sender-receiver keys and hence both the schemes could be implemented. The limitation of absence of a direct link between helper and receiver was overcome by altering the header of the packet transmitted by the helper to match the one that would be transmitted if the helper were the AP. In this way the receiver accepted the packet thinking it was sent from the sender (AP) which is the desired result of CoopMAC.

Under the above setup we were able to implement both the proposed schemes and to transmit encrypted packets with IEEE 802.11i security from the sender to the receiver through the helper. The code for the implementation of the security schemes can be found and downloaded from the official CoopMAC site in [7].

### *C. Discussion*

Of all the security issues, the most important concern with the CoopMAC approach where we need the helper to properly forward the packet to the destination is not that the helper being in possession of the packet may try to decode it (this can be even done by passive sniffing), but that it has to be given the ability to modify the header but not the payload of the packet. The above two implementations individually will address all the security concerns we have in a cooperative network due to the high robustness of 802.11i. The following points can be observed with respect to our implementations:

- The helper cannot decrypt the packet as it does not have the appropriate keys. 802.11i uses separate keys for each station and no private keys are shared with the helper.
- The helper itself will be an authenticated station using 802.1X and hence will be a trusted entity. In order for a bad helper to be a part of the network implies that first 802.1X server has to be hacked into.
- The helper may try to spoof some packets and send them to the receiver, but as it does not have the proper keys it will not be able to do so. Similarly any kind of session hijacking will not be possible.
- An attack by an authenticated helper station will be limited to denial of service. On detection of the loss of packets, the transmitter can quickly shift to another helper or transmit directly and blacklist the helper so as not to use it later.
- Other than this there are no possible attacks from the helper due to the inherent security features of 802.11i, which takes care of the following security concerns:
  - access to data (using strong encryption like AES)
  - spoofing (per packet encryption/authentication)

– Replay (as CCMP uses a 48-bit Packet Number (PN) to prevent replay attacks and construct a fresh nonce for each packet. The large space of PN eliminates any worry about PN re-usage during an association) and
– man in the middle attacks (strong mutual authentication)

It can also be noted that our implementations does not open up any other security holes, as the environment will be controlled (by appropriate modification in the driver/firmware) at the source and the helper. These do not require disclosing of any private keys by the source, helper or receiver and hence the data can at no point be decrypted by undesired stations, hence maintaining the confidentiality and integrity of the environment.

## VI. Conclusions

In this paper we study the security implications that a Cooperative MAC protocol introduces in the current WiFi security framework. We conclude that an intermediate relay station that forwards packets to the destination can destroy security when WEP is used, by changing the header or the payload of the packet. However, when WPA or WPA2 (802.11i) is used, the intermediate station can not change the packet since now both the payload and the header are used for the encryption of the packet. Furthermore we propose two schemes for adjusting security (WPA or WPA2) to the new cooperative environment. In order to show feasibility of the proposed schemes, we implemented them using open source drivers. From our implementation experience we obtained insights into the security aspects on a cooperative network.

## References


[1] P. Liu, Z.Tao, and S. Panwar, "A Cooperative MAC Protocol for Wireless Local Area Networks," in *Proceedings of IEEE ICC'05*, June.

[2] "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *ANSI/IEEE Std 802.11, 1999 Edition*, 1999.

[3] T. Korakis, Z. Tao, S. Makda, B. Gitelman, and S. Panwar, "To Serve is to Receive Implications of Cooperation in a Real Environment," in *Proceedings of Networking 2007*, June.

[4] "Amendment 6: Medium access control (mac) security enhancements," *ANSI/IEEE Std 802.11, 1999 Edition*, 1999.

[5] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11," in *The Seventh Annual International Conference on Mobile Computing and Networking*, July.

[6] "MADWiFi: Multiband Atheros Driver for WiFi," *http://madwifi.sourceforge.net/*.

[7] "Website for the Cooperative MAC Implementation," *http://eeweb.poly.edu/coopmac/*.