International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2021/209994 Volume 8, Issue 5, September – October (2021)

RESEARCH ARTICLE

COLSR An Efficient Mechanism to Secure OLSR Protocol Against Multipoint Relay Attack

Abderrahim HAJJI SOUALFI
Department of Science Computer, University of Sciences and Technologies Errachidia, Morocco
hajji777@gmail.com

Said AGOUJIL
Department of Science Computer, University of Sciences and Technologies Errachidia, Morocco
agoujil@gmail.com

Received: 04 September 2021 / Revised: 07 October 2021 / Accepted: 17 October 2021 / Published: 27 October 2021

Abstract – A MANET (Mobile Ad Hoc NETwork) can be considered as a collection of self-organizing mobile devices that can communicate without the aid of an integrated infrastructure or a back-end administration. Defense, crisis management, telemedicine, tele-geoprocessing, virtual navigation, commercial and civil environment, etc. are multiple applications of this type of network. However, MANETs present several significant constraints, including restricted processing capabilities, limited bandwidth, short battery life and vulnerability to multiple attacks. This paper introduces the negative effect of the Multipoint Relay (MPR) attack against an effective routing MANET protocol, OLSR (Optimized Link State Routing). In this attack, a malicious node can broadcast altered control messages over the network to cause loss of messages and connectivity. After a study of existing solutions, a security extension, named COLSR (CMAC for OLSR) and based on CMAC (Cipher-based Message Authentication Code) is proposed and validated by simulations under the OMNeT++/INETMANET environment with the integration of the Crypto ++ library. Our solution offers a good compromise between robustness in terms of security and protocol performance. It ensures the integrity of the OLSR control messages, nodes authentication and has the advantage of a modest impact on the network performances.

Index Terms – MANET, OLSR, MPR Attack, Wireless Security, COLSR.

1. INTRODUCTION

With the expanding use of mobile devices, MANETs cover more applications fields in wireless communications like military service, emergency operations, disaster relief, vehicle and sensor networks, smart cities, Internet of Things (IoT), education, etc. [1]. The nodes forming the network can move, join or leave freely at any time. Because wired routing protocols are limited, many wireless routing protocols have been developed especially for. From many MANET protocols, Optimized Link State Routing (OLSR) protocol is used in wireless large and huge dense networks [2]. The protocol is designed to work in an entirely distributed process and does not depend on any dedicated or central authority. To provide the shortest path routes to all destinations, the OLSR protocol uses three different control messages: Hello, Topology Control (TC) and Multiple Interface Declaration (MID) messages [3]. The protocol tolerates a reasonable loss of some control messages due to wireless transmission current problems. To handle this issue, each control message includes a sequence number increased every sending message. In this manner, the receiving node can identify the most recent topology information, even if the sent messages have been reverted during transmission [4]. To reduce the quantity and the size of control traffic information inundated in the network, the concept of the MultiPoint Relay (MPR) has been introduced. Only MPR nodes can create the link state information and may choose to notify links between themselves and their selectors. A malicious node can take this improvement to lead an MPR attack by diffusing or forwarding altered control messages to its neighbors [5]. Since there is no protection strategy against this attack -in the literature- the OLSR protocol is still exposed. Our goal is to propose and validate (by simulations) a simple, effective security solution for the OLSR protocol against the MPR attack in the Ad Hoc environment and which considers the security constraints (dynamic topology, modest performance, lack of cooperation, etc.).

In the next section, we will introduce the MPR attack. After, we propose an efficient mechanism -COLSR- to secure the OLSR protocol against this attack. In the last section, we discuss the simulation results of our approach compared to the attacked network and the standard OLSR network, before concluding.

2. MULTIPOINT RELAY ATTACK

To assure the packets' delivery, routing protocols achieve two compliments steps. The first phase consists of discovering the network topology when nodes exchange information about

RESEARCH ARTICLE

the network layout. In the second stage, a source node sends a message(s) to a destination with the help (transmission) of the other nodes network [6]. Counter to the wired network, when the routing operations are usually managed by a physical dedicated and administrated resources, MANET information topology is transmitted by nodes forming the network. Many security issues were raised in this case. OLSR is one of the proactive routing protocols of MANET. The three control messages of the protocol are transmitted or broadcasted clearly in the network. A malicious node can use this property to cause a security problem with a multipoint relay attack. A malicious (non-MPR) node can aim at an MPR attack by flooding or forwarding normal or altered control messages.

In this scenario (Figure 1), the malicious node X (non-MPR) broadcasts, in its Topology Control (TC) message, that node A is not on its MPR selector list. Nodes E, F and G receiving this message, and with an ANSN (Advertised Neighbor Sequence Number - A sequence number associated with all the neighbors of a sending node) greater than those of nodes B and C, will declare a lost link with node A. The consequences in this case are certainly a loss of messages and connectivity.

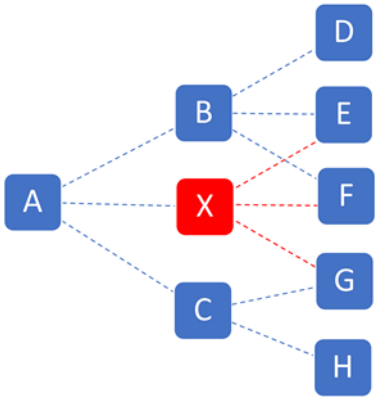


Figure 1 Node X Performs an MPR Attack

The MPR attack causes connectivity or/and message loss and, in some cases, unnecessary network overload. We have demonstrated, by simulations, the negative effect of this attack [5].

3. EXISTING SECURITY APPROACHES

A lot of works has been elaborated offering protection to OLSR protocol against several attacks considering confidentiality, integrity, availability, authentication, and non-repudiation security requirements.

Raffo et al. [7] deploy, in each node, an embedded directional antenna and a GPS device, to include the geographical position of the sending node and to evaluate the reliability of the links. Combining node location and signatures, the solution prevents the wormhole attack and the circulation of false messages. Signatures with timestamps assure shielding from the incorrect traffic generation or relaying.

The authors of [8] use a hybrid version of the IPSec protocol, including both authentication header and encapsulating security payload modes, to deliver a green result concerning the energy consumption security solution for MANETs.

In [9], Clausen et al. propose a security extension built on signatures algorithms like ECDSA, HMAC, DSA and RSA. They launch the concept of admittance control for OLSRv2 (OLSR second version) protocol. After the extension evaluation, the authors found that the HMAC message signature needs extensively little time than RSA, DSA, and ECDSA algorithms for the creation and the verification of control messages.

The authors of [10] use a new technique reducing the messages overhead compared to the classic flooding mechanism. The solution is based on four-way handshaking in the middle of two nodes adopting the HIP (Host Identity Protocol).

With a secured OLSR framework, K. Tamil Selvi and S.Kuppuswami [11] secure and minimize the number of MPR nodes using threshold cryptography and distribution of a shared encrypted secret key. The goal is to bring the veracity for routing messages, particularly Topology Control messages, preventing by this way altered forwarded routing messages. The simulation results confirm the framework implementation.

The authors of [12] use another kind of security against a camouflaging wormhole attack. It is a specification based Intrusion Detection System (IDS). The experimental performance results, using the NS2 simulator, present the positive efficiency of the proposed algorithm.

Sonam Gadekar and Sujata Kadam [13] modify the OLSR standard protocol against the node isolation attack by enhancing its MPR selection policy, to suggest a performant protocol in terms of the required MPR nodes, TC messages, energy consumption, and Packet Delivery Ratio (PDR).

The DCFM (Denial Contradictions with Fictitious Node Mechanism) [14] supports anticipation, in opposition to some vulnerabilities, and countermeasures, to treat bad behaviors, by proceeding to the same attacker's strategy. This technique creates virtual(imaginary) nodes and exploits the victim node’s data to shield the network.

The authors of [15] increase the security of the protocol with the help of Q-Learning (Reinforcement Learning technique) by choosing reliable nodes to forward messages. Those nodes are elected counting on the input features. The OLSR protocol is modified to select the most trusted set of MPR. This security solution costs processing power, delay, and memory

RESEARCH ARTICLE

of the network devices. With a minimum number of nodes in the network, R. Bhuvaneswari and R. Ramachandran [16] suggest an implementation of Elliptic Curve Cryptography (ECC) for OLSR protocol (OLSR-ECC), in opposition to DoS attacks. Since the ECC is strongly based on elliptic curves (specialized mathematics), it is harder to break. The results show the efficiency of the application.

Hamela Kanagasundaram and A. Kathirvel propose an energy-efficient and security-based model for the protocol including an enhanced intellects-masses optimizer (EIMO) [17]. The result of this work is a combination of an effective MPR selection algorithm with the less ergonomic energy model. The network performance metrics like the average network lifetime, the total remaining time, the variance of energy and the energy consumption are respected.

In summary, most of the solutions cited above have attempted to counter one or a few attacks, while others have looked at the security problem as a whole. The results of these outcomes differ depending on the proposed protection architecture. In most cases, developers are looking to find the performance/security tradeoff of the Ad Hoc network. Among these developers, some succeeded in the challenge while others sacrificed the performance side for the sake of security. No solution has specifically addressed the MPR attack against the OLSR routing protocol, hence the need to find a technique to counter this kind of attack.

4. COLSR SOLUTION

4.1. MAC ALGORITHM

In front of those multiple security solutions, our work must be focalized on how to secure OLSR protocol from the MPR attack, since there is no literature answer remedying this issue. This attack becomes effective when a malicious node forwards or broadcasts altered control messages. A Message Authentication Code (MAC) must be used to ensure message integrity (validity of a transmitted message) and authentication (authenticity of the originator message) [18]. The message authentication code appends an authentication tag to a message by the exploitation of a symmetric key (unlike digital signature) for both generating and verifying the tag. Since the MAC is much faster than a digital signature, given the fact it is based on both block ciphers (CMAC, VMAC, …) or hash functions (HMAC, SipHash, …) [18], it (MAC) becomes more suitable to provide security to a MANET in which the topology changes frequently and the end-to-end delay metric is crucial. The MAC function exploits a symmetric key k to generate an m tag for a message x. We note:

$$m = MAC_k(x)$$

After computing the m tag using the secret key k, the Sender node sends both the message x and the m authentication tag. While the MAC is based on a symmetric key, the Receiver node will recompute the authentication tag m' combining the received message x and the same shared key k. the last step involves the verification of the received m tag with the regenerated m' tag. Figure 2 shows the MAC calculation and verification.

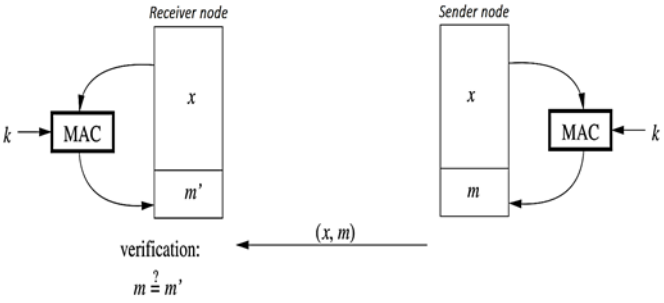


Figure 2 MAC Calculation and Verification

The MAC technique can detect the attacker's behavior when he performs an MPR attack. The MAC recomputation, in reception, will yield an incorrect result if the OLSR control messages were generated or altered by a malicious node.

For each control message (Hello, TC, and MID), the originator (Sender node) will append the *m* (m 1, m 2, …, m 1, m 2, …, M 1, M 2, …) tags (Figure 3):

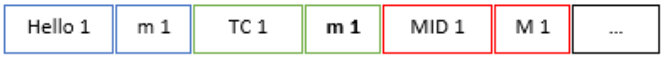


Figure 3 Packet Format in MAC-OLSR

The m tags are used by the Receiver node to validate (or not) the related OLSR control message: every control message without the identical peer tag is rejected.

To secure OLSR control messages with the MAC algorithms, we use the Crypto++ library. Crypto++ is an open-source free C++ class library, enriched by many tools of common and uncommon used cryptographic algorithms [19]. The library contains multiple choice of MAC algorithms and supports the GCC compiler included with the OMNeT++ simulation environment in which we implement our network.

The Crypto library presents two kinds of MAC algorithms based on:

- Block ciphers: CMAC, DMAC, VMAC, GMAC (GCM), Poly1305;
- Hash functions: BLAKE2b, BLAKE2s, Two-Track-MAC, HMAC, SipHash.

The block cipher algorithms of MAC are based on the performant Advanced Encryption Standard. AES is a NIST (National Institutes of Standards and Technology) approved block cipher specified in Federal Information Processing

RESEARCH ARTICLE

Standard (FIPS) Pub. 197. Nowadays, the AES algorithm becomes the reference in encryption. It proves so effective in terms of design, implementation, and hardware, granting high performance in many applications. It is considered like the faster (from four to ten times) cipher, on most modern platforms, compared to the previous Triple Data Encryption Standard (3DES -- [SP800-67]) cipher [20]. This makes the AES symmetric cipher, not only the practical choice but the perfect one.

4.2. Benchmark

The next step consists of choosing the best MAC algorithm, based on AES block cipher, to secure OLSR protocol from MPR attack, taking into consideration the speed of generating the **m** tag, the security assurance, and the MANET properties. To determinate the speed of MAC algorithms, a benchmark (Table 1) has been realized with Crypto++ library:

Attribute	Value
Algorithm	Message Authentication Codes (MACs)
Benchmark time	3s for each algorithm
CPU architecture	64-bit (Intel Core i7)
CPU frequency	2GHz
Crypto++ version	8.2
Simulation environment	Windows 10 64-bit

Table 1 Benchmark Crypto++ Setting

Table 2 shows the performance, in term of speed, of MAC algorithms based on a block cipher:

Algorithm	Tag gener-ation speed (MiB/s)	Tag gener-ation speed (cpb)	Time to Setup Key and IV (ms)	Time to Setup Key and IV (cpb)
VMAC	4637	0.4	0.848	1696
GMAC	4339	0.4	0.476	953
Poly1305	845	2.3	0.329	658
CMAC	541	3.5	0.174	348
DMAC	509	3.7	0.658	1316

Table 2 MACs -Based Block Cipher- Speed Benchmark (Cypto++ Library)

VMAC, a block cipher algorithm MAC using universal hashing proposed by Dai and Wei Ted Krovetz, is the fastest described MAC on the present 64-bit processors. It is designed for high efficiency in software on the 64-bit architectures [21]. Table 2 highlights this performance: the algorithm needs only 6.4 clock cycles (0.4 cycles per byte) to generate a 128-bit digest with an average speed of 4637 MiB/s.

GMAC is a special case of GCM (Galois/Counte Mode) where there is only data to be authenticated (not encrypted). Standardized by the NIST, GMAC is very efficient in hardware and software [22]. The performance of GMAC algorithm is comparable with VMAC ones in terms of speed: 4339MiB/s for GMAC against 4637MiB/s for VMAC with the same speed generation tag (0.4 cycles per byte): the algorithm needs 6.4 clock cycles to generate the **m** tag. We can notice that GMAC is even faster to set up the Key and the Initialization Vector (IV): 0,476ms for GMAC against 0,848ms for VMAC.

Poly1305 is a modern secret-key MAC suitable for a vast diversity of applications that have been introduced by Daniel J. Bernstein. Poly1305-AES guarantees cipher replaceability (from AES to another function), high speed (cf. table 1), low per-message overhead, security (if AES is secure), parallelizability, and incrementality [23]. We observe (Tab. 2) that the Poly1305 algorithm is slower compared to the two previous algorithms (VMAC and GMAC): Poly1305 requires 36.8 cycles (with an average speed of 845Mio/s) to generate the tag. While it takes less time for the initialization of the key and the initialization vector (0.329ms).

For each sent message, the three authentication functions (VMAC, GMAC and Poly1305) need a secret nonce (an arbitrary number that is only ever used once), beside the secret **K** Key, to generate the **m** tag. In this case, the receiver node should know this nonce to create the correct **m'** tag. This solution will generate a lot of traffic in the network and make the generation and verification operations more complex. In terms of security, the integrity can be compromised if the MANET nodes provide the same value for the nonce more than once [18]. For example, in the case of the VMAC algorithm, an attacker can easily forge VMAC tags if he can notice two generated tags with the same key and the same nonce for distinct messages [24]. Regarding GMAC function, if the network uses short or truncated tags (32 bits long), it is possible to forge the GMAC after 2^{16} tries [25]. In the case of the Poly1305 algorithm, the nodes must specify a mechanism of nonce generation and maintenance preventing duplication [23].

CMAC (Cipher-based Message Authentication Code), corresponding to One-Key CBC MAC1 (OMAC1), is a keyed hash algorithm built on the AES symmetric key block cipher. NIST recommends the exploitation of CMAC for operating a

RESEARCH ARTICLE

block cipher-based authentication [26]. Compared to the other MACs algorithms, CMAC shows the minimum time to set up the Key and the Initialization Vector (cf. Tab. 2). With a moderate average speed of 541MiB / s, the CMAC algorithm generates the **m** tag in 56 cycles (56 = 3.5 × 16). Contrary to the three previous MAC algorithms (VMAC, GMAC, and Poly1305), CMAC does not take a nonce as input to generate the tag [22].

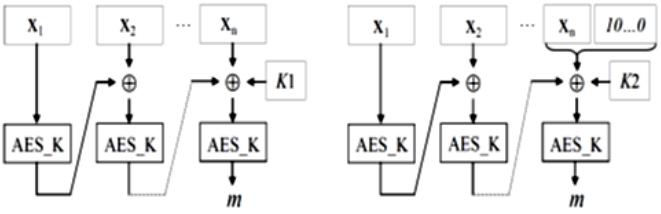
CBC-MAC (Cipher Block Chaining Message Authentication Code) is a basic and textbook algorithm that builds a MAC for predetermined-length messages out of a block cipher or fixed-length bit strings. DMAC (Double MAC) is a variant of the CBC-MAC, used to manage messages of variable unknown lengths. To assure security for unpredicted-length messages, the final block is encrypted lately with a distinct key. Since the block length is short related to the schemes based on a hash function, the DMAC algorithm is more desirable to authenticate short messages [27].

To secure OLSR protocol from the MPR attack, CMAC is the suitable choice. The CMAC authentication function combines speed generating (of the authentication tag) and security assurance (of the different OLSR control messages). We call our implementation COLSR.

4.3. CMAC ALGORITHM

In the CBC-MAC algorithm, a message to authenticate is divided into a series of blocks. Every block depends on the block cipher result of the previous block. This dependence ensures that a modification to any of the message bits will alter the rest of the MAC in an unreliable manner. Unfortunately, CBC-MAC is secure only for fixed-length messages and has security deficiencies [28]. Black and Rogaway submit and demonstrate the security of some derived of CBC-MAC known as FCBC, ECBC, and XCBC. The XCBC (eXtended Cipher Block Chaining) mode resolves the security issues of CBC-MAC but needs three keys for generating the MAC tag. Iwata and Kurosawa present an enhancement of XCBC, termed One-Key CBC-MAC (OMAC). Later, they introduce a refinement of OMAC named OMAC1, the corresponding CMAC [29]. CMAC is designed to identify intended, illegitimated or unintentional modifications of the data, providing thereby a powerful assurance of data integrity than a checksum or an error-detecting code [26].

The authors of [29] formalize a proof of unforgeability for CMAC in the EasyCrypt library. This proof provides very similar safety limits to Iwata and Kurosawa for CMAC and relies on the inability to distinguish CMAC from the FCBC variant. The CMAC algorithm relies upon the choice of an underlying symmetric key block cipher. AES-CMAC algorithm -implemented in the Crypto++ library- uses the Advanced Encryption Standard [NIST-AES] as a building block. It takes a secret key ‘K’, a message of variable length ‘X’, and the length of the message ‘L’ as inputs and returns a 128-bit string MAC tag ‘m’. There are two cases of operation in CMAC (see Figure 4): the message ‘X’ to verify is divided into ‘n’ blocks. If the size of ‘X’ is equal to a positive multiple of the 128 bits block size (case a), the final block should be XORed with the subkey ‘K1’ before generating the ‘m’ tag. Else (case b), the last block must be padded (to adjust its length to the block limit) with 10^i then exclusive-OR’ed using the subkey ‘K2’ before its encryption. The Subkeys K1 and K2 are derived from the secret ‘K’ key and used in both MAC generation and verification algorithms. AES_K is the encryption algorithm assured by AES standard with the confidential key ‘K’:



Case a: positive multiple block length || Case b: otherwise

Figure 4 Illustration of the Two Cases of MAC Tag Generation

The following Figure 5 describes the MAC generation algorithm [30]:

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                      Algorithm AES-CMAC                                   +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                           +
+   Input    : K    ( 128-bit key )                                         +
+            : X    ( message to be authenticated )                         +
+            : L    ( length of the message, in octets )                    +
+   Output   : M    ( Message Authentication Code tag )                     +
+                                                                           +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+   Constants: const_Zero is 0x00000000000000000000000000000000            +
+              const_Bsize is 16                                            +
+                                                                           +
+   Variables: K1, K2 for 128-bit subkeys                                   +
+              X_i is the i-th block (i=1..ceil(L/const_Bsize))             +
+              X_last is the last block xor-ed with K1 or K2                +
+              n      for number of blocks to be processed                  +
+              r      for number of octets of last block                    +
+              flag   for denoting if last block is complete or not         +
+                                                                           +
+   Step 1.  (K1,K2) := Generate_Subkey(K);                                 +
+   Step 2.  n := ceil(L/const_Bsize);                                      +
+   Step 3.  if n = 0                                                       +
+            then                                                           +
+                 n := 1;                                                   +
+                 flag := false;                                            +
+            else                                                           +
+                 if L mod const_Bsize is 0                                 +
+                 then flag := true;                                        +
+                 else flag := false;                                       +
+                                                                           +
+   Step 4.  if flag is true                                                +
+            then X_last := X_n XOR K1;                                     +
+            else X_last := padding(X_n) XOR K2;                            +
+   Step 5.  A := const_Zero;                                               +
+   Step 6.  for i := 1 to n-1 do                                           +
+                 begin                                                     +
+                   B := A XOR X_i;                                         +
+                   A := AES-128(K,B);                                      +
+                 end                                                       +
+            B := X_last XOR A;                                             +
+            M := AES-128(K,B);                                             +
+   Step 7.  return M;                                                      +
+                                                                           +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 5 AES-CMAC Algorithm

RESEARCH ARTICLE

After generating the subkeys K1 and K2 from K (Step 1), the number of blocks 'n' is calculated (Step 2). In the Third Step, the algorithm will check the length of the input message (Hello, TC, or MID message in the OLSR protocol case) to set the flag as false (if the input length is null or if the last block length is less than 128 bits) or true (if the last block length has the 128 bits). The flag state (Step 4) determines if the last block (X_last) will be XORed with the subkey K1 (flag=true) or padded then XORed with the subkey K2 (flag=false). In Step 6, the Cipher Block Chaining (CBC) technique, using the AES encryption algorithm and the shared secret Key 'K' with the zero block 'A' as the initialization vector (Step 5), is applied to the formatted message. The final step (Step 7) consists of generating the 'M' tag.

To verify the received tag, the receiver node will use the same algorithm AES-CMAC recomputation. The verification algorithm takes four inputs, the secret key 'K', one of the OLSR control messages 'X', the length of the corresponding (Hello, TC or MID) message 'L', and the received tag 'M' ' to be verified. The output of the MAC verification is either VALID (if M'=M) or INVALID (if M'≠M). If the output is VALID, the OLSR control message is authentic and not corrupted in transit between nodes. If not (INVALID output), the control message must silently discard, because it was sent from a malicious node or/and altered during the transition.

4.4. Key GENERATION

To guaranty, the authentication and the integrity of the OLSR control messages, all of the MANET participant nodes must keep the shared secret key 'K' safe and generated in the randomness requirement way of RFC 4086 [31]. Taking a very complex algorithm to calculate a cryptographic key, selecting a quantity from a large database, using simple numeric or logical operations to produce pseudo-random sequences are ideas that can lead to insecure pseudo-random number generation. Many public standards are widely expanded to produce keys or other cryptographically random varieties. Some standards build on an entropy source. Others bring the pseudo-random number strong-sequence generator but consider the input of a random seed or input from a source of entropy [32].

We use an AutoSeededRandomPool incorporated in the Crypto++ library to generate a pseudo-random key for CMAC. The design idea of the AutoSeededRandomPool was proposed by Leonard Janke. The algorithm uses a PGP-style generator based on AES to extract entropy from the OS pool. It is suitable for all cryptographic purposes including generating keys and Initialization Vectors (IVs).

4.5. CLOSR ALGORITHM

To summarize, the COLSR (CMAC-OLSR) algorithm will take the following steps (Figure 6):

- Generating the pseudo-random secret Key (AutoSeededRandomPool/Crypto++).
- Distributing the shared key to the trusted nodes (MANET initialization).
- Using CMAC-AES to verify the integrity (of the control message) and the authenticity (of the originator message).

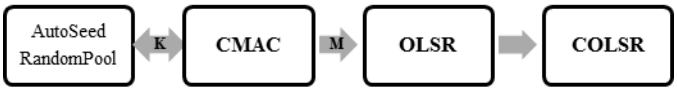


Figure 6 COLSR Solution

5. SIMULATION

This section presents the validation simulation of our approach "COLSR" compared with the standard OLSR protocol in two different circumstances (in the presence and absence of the MPR attack). The first simulation result (Figure 6) concerns the normal running of the network (using OLSR protocol for routing). In the second case (Fig. 7), we observe the MANET performance when it is submitted to the MPR attack. In the presence of the malicious nodes, we inspect our solution effect (Fig. 8) when COLSR is reacting against the attack. We use OMNeT++ simulation environment with the INETMANET framework and the integration of Crypto++ library to verify our security proposition. Unlike [5], in which a non-MPR node can flood or forward normal control messages, this work will be concentrated on the repercussion of misbehavior nodes flooding or forwarding altered control messages. The MPR attack damage is less effective as the network size increases [32]. The table 3 regroups the simulations setting:

Attribute	Value
Nodes number	100
NB of malicious nodes	10 (in the attacked and protected network)
Routing protocol	OLSR
Mobility	Stationary
Simulation time	The 30s
IP Addressing	IPv4 (192.168.1.0/24)
Wireless standard	802.11g
Link bandwidth	6Mbps
Transmission range	100m
Simulation area	1000m×1000m

Table 3 Network Simulations Setting

RESEARCH ARTICLE

The experimental evaluation is carried out with three different parameters like Packet Delivery Ratio (PDR), end-to-end delay, and throughput. Also, comparative results will be discussed.

The following graphs present the simulation results in a normal environment:

In the case of an attacked network, the simulation results are shown in the Figure 7. In the case of an attacked network, the simulation results are shown in the below Figure 8.

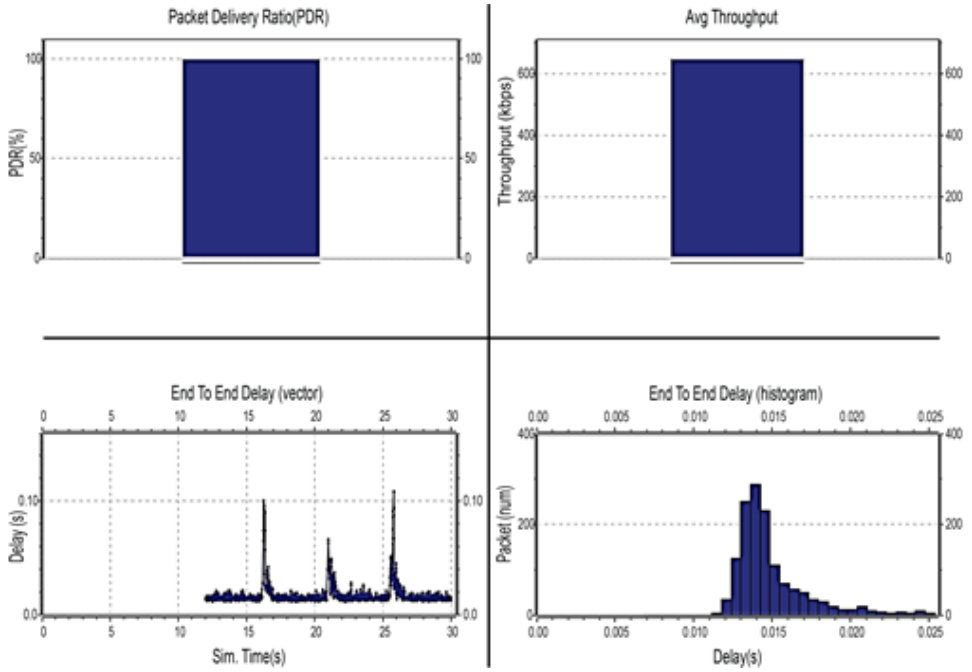


Figure 6 Simulation Results in Normal Network

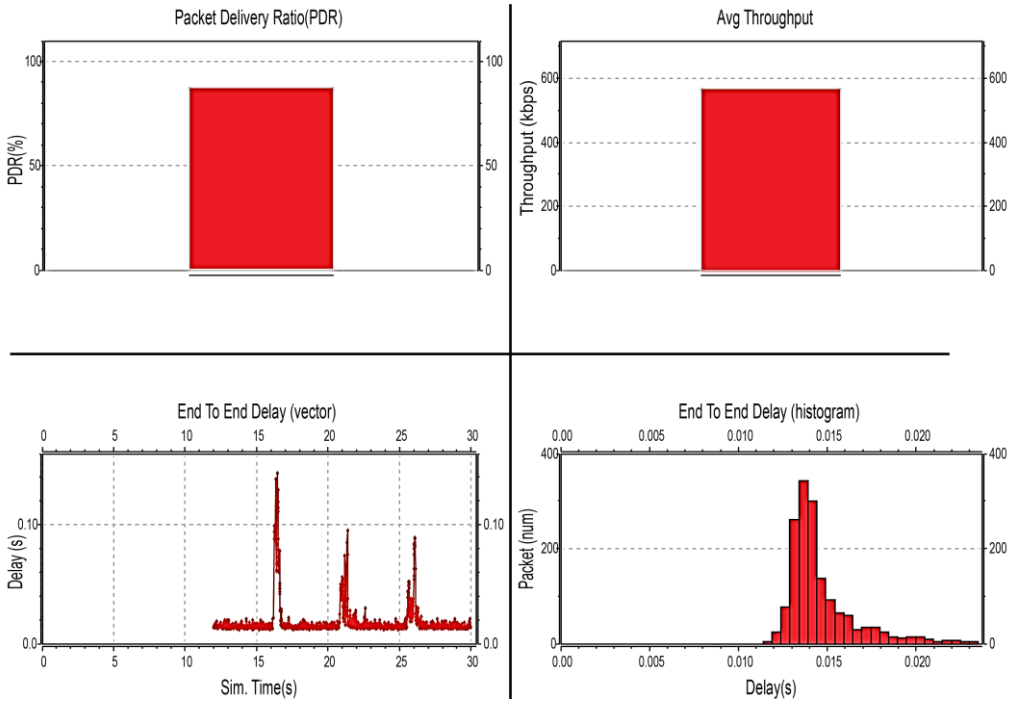


Figure 7 Simulation Results in Attacked Network

RESEARCH ARTICLE

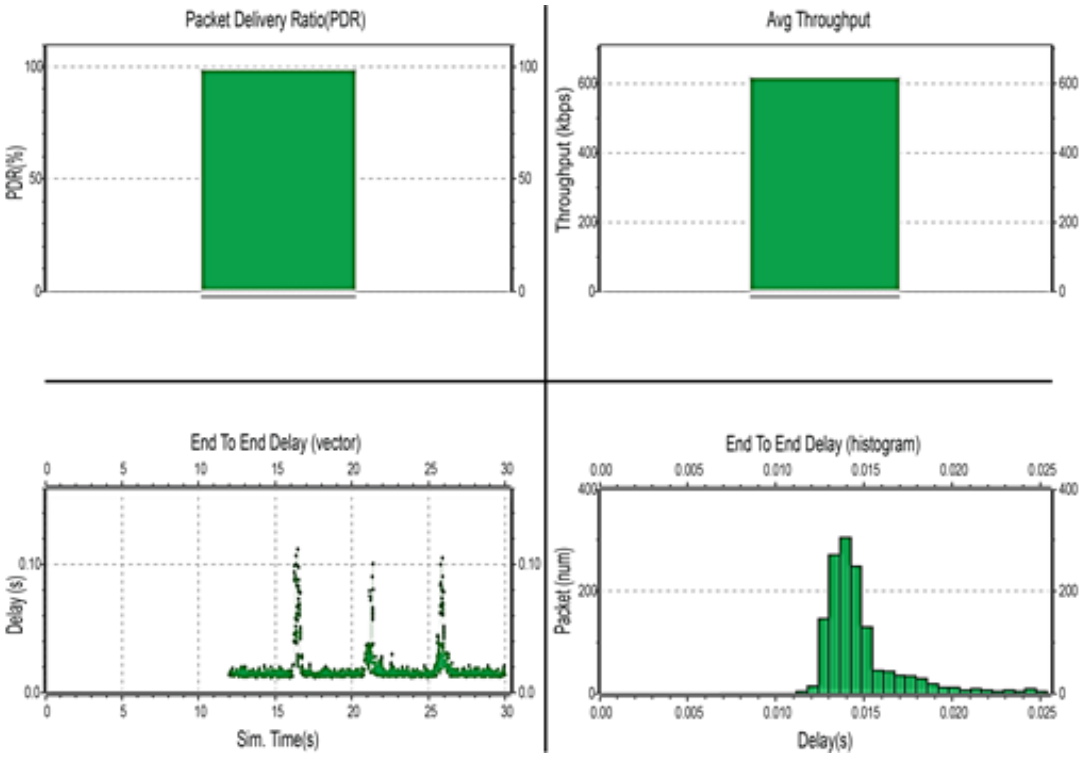


Figure 8 Simulation Results in the Protected Network

6. DISCUSSION

6.1. Packet Delivery Ratio (PDR)

The PDR metric is very necessary to evaluate the performance of a routing protocol in any network.

The PDR is defined as the ratio of data packets successfully arriving at all the destinations compared to the total sent data packets from the different sources. The performance is better when the packet delivery ratio is high (ideally, PDR = 100%). It allows us to verify if the protocol extension has an impact on the successful transfer of data packets.

This metric is presented by each graph in the top left of each precedents Figures (Figures 6, 7, and 8). We can see the negative effect (Attacked MANET) when the MANET is under MPR attack (cf. Table 4). The misbehavior nodes represent only 10% (10 among 100) of the global MANET nodes. When the COLSR implementation (Protected MANET) is taking effect, the PDR metric is back approximately to the normal (cf. Table 4). The following table regroups the PDR metric statistics in the three cases:

	Normal MANET	Attacked MANET	Protected MANET
PDR (%)	100	87,3	98,2

Table 4 PDR Metric in Three Cases

Compared to the network in its normal behavior, the reduction of the PDR metric in the case of the attacked network is evaluated at:

$$PDR_{att\_red} = 12,7\%.$$

The COLSR solution tries to reduce this large gap in order to stabilize the functioning of the network, without affecting the PDR metric too much:

$$PDR_{pro\_red} = 1,8\%.$$

After the packets have been sent, some of them will not have enough time to be delivered to their destinations, because the implementation of the COLSR extension will generate an additional (necessary) time for the verification of the TC topology control messages, while the time of the different simulations is limited (30 seconds). This will have an impact on the PDR metric with a slight decrease.

6.2. Throughput

Throughput (in bits/s) expresses the quantity of data successfully delivered in a given amount of time through a communication channel. This metric is impacted by many factors like the node mobility, number of hops, and transmission range.

Table 5 regroups the different statistics in the three cases:

RESEARCH ARTICLE

	Normal MANET	Attacked MANET	Protected MANET
Throughput (kbps)	638,2	562,2	620,8

Table 5 Throughput in Three Cases

Like the PDR metric, the attack harms the throughput metric. In the attacked scenario, the average throughput is reduced by:

$$Throughput_{att\_red} = 11,9\%.$$

The throughput metric depends on the number of bits successfully delivered. This amount will reduce significatively when the network is under MPR attack.

The COLSR remediation has a small cost on the throughput parameter. When the network is protected, the throughput metric is decreased by:

$$Throughput_{pro\_red} = 2,72\%.$$

This slight deduction is caused by the needed additional time to confirm (or not) the integrity and the authenticity of the topology control messages.

6.3. End to End Delay

End-to-end delay measures the time between a packet sent by a sender and its reception by a receiver.

This metric can characterize the effectiveness of the protocol in terms of optimal path selection and response time.

The table 6 highlights the attack affects the end-to-end metric:

End to end delay(ms)	Normal Network	Attacked Network	Protected Network
Min	10,4	13,6	11,2
Max	105,8	120,2	106,1
Mean	16,3	24,5	18,7

Table 6 End to End Delay (vector charts)

In the case of the attacked network, the average end-to-end delay will rise compared to the normal network. This increase is estimated to be:

$$Delay_{att\_inc} = 33,47\%.$$

Our COLSR response narrowed this large gap considerably, attempting to reach the perfect value of the average delay in the standard network.

$$Delay_{pro\_inc} = 12,83\%.$$

The histogram representation of the delay metric confirms this difference in delay. We note that there is a significant number of packets received with a greater delay in the case of the attacked network compared to the normal network. The COLSR solution provides protection for the network while affecting the delay in a moderate way:

	Received Packets	End to end delay(ms)
Normal MANET	283	[13,6 14,3]
Attacked MANET	346	[14,2 14,9]
Protected MANET	312	[13,8 14,6]

Table 7 End to End Delay (Histogram Charts)

The table 7 shows the end to end delay. The augmentation -in time- of the "latency for route discovery" and "transit time in intermediate node queues" caused by the malicious non-MPRs nodes, which broadcast false information topology, will increase the end-to-end delay metric in the attacked network.

7. CONCLUSION

OLSR is a proactive protocol distinguished by the use of nodes called MultiPoint Relays (MPRs). The MPR nodes optimize the control messages flooded in the network. Malicious nodes can take this privilege to diffuse incorrect topology information, which we call an MPR attack. The attack affects the global performance of the MANET. To manage this situation, we have implemented a solution based on CMAC -a MAC variant- to ensure the integrity and the authenticity of the control messages. Thus, by using our solution, called COLSR, the network behavior has back to normal with a slight impact on the performance network. Easy to implement, our application does not need any time synchronization or location information. An exciting future work would be the combination of our extension with an advanced key management system that will upgrade the secured MANET to an autonomous network.

REFERENCES

[1] Moudni H., Er-rouidi M., Faouzi H., Mouncif H., El Hadadi B. "Enhancing Security in Optimized Link State Routing Protocol for Mobile Ad Hoc Networks". In: Sabir E., García Armada A., Ghogho M., Debbah M. (eds) Ubiquitous Networking. UNet 2017. Lecture Notes in Computer Science, vol 10542, pp. 107-116, 2017.

[2] Azeez A.A.A., Isaac E., Thampi S.M. "Anonymous and Secured Communication Using OLSR in MANET". In: Abraham A., Mauri J.L., Buford J.F., Suzuki J., Thampi S.M. (eds) Advances in Computing and Communications. ACC 2011. Communications in Computer and Information Science, vol 193, 2011, pp. 145-154. Springer, Berlin, Heidelberg.

RESEARCH ARTICLE

[3] Clausen Ed. and Jacquet Ed., "Optimized Link State Routing Protocol". IETF: The Internet Engineering Task Force, RFC 3626, p. 8, October 2003.
[4] Hajare, Priyanka and P. Tijare. "Secure Optimized Link State Routing Protocol for Ad-Hoc Networks". (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (1), 2012, pp. 3053 - 3058.
[5] Abderrahim Hajji Soualfi and Said Agoujil. "Performance Impacts Of Multipoint Relay Attack Against OLSR Protocol", International Journal of Scientific & Technology Research 9, 2020, pp. 1662-1666.
[6] Ferrag, M. and M. Nafaa. "Securing the OLSR routing protocol for Ad Hoc Detecting and Avoiding Wormhole Attack". Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), April Edition, 2011, pp. 51-58.
[7] D. Raffo, C. Adjih, T. Clausen, and P. Muhlethaler. "Securing OLSR using node locations". In Proceedings of 2005 European Wireless (EW 2005), pages 437-443, Nicosia, Cyprus, April 10–13 2005.
[8] E.A. Panaousis, G. Drew, G. Millar, T.A. Ramrekha, C. Politis, "A Test-bed Implementation for Securing OLSR in Mobile Ad-hoc Networks", International Journal of Network Security & Its Applications, Vol. 2, No. 4, pp. 141-160, October 2010.
[9] T. Clausen, U. Herberg, and J. Milan. "Digital signatures for admittance control in the optimized link state routing protocol version 2". Research Report RR-7216, INRIA, February 2010.
[10] Azeez, A.A., Isaac, E., Thampi, S.M.: "Anonymous and secured communication using OLSR in MANET". In: Abraham, A., Mauri, J.L., Buford, J.F., Suzuki, J., Thampi, S.M. (eds.) ACC 2011. CCIS, vol. 193, pp. 145–154. Springer, Heidelberg (2011). doi:10.1007/978-3-64222726-4_16.
[11] Selvi, K.T., Kuppuswami, S.: "Enhancing security in Optimized Link State Routing protocol for MANET using threshold cryptography technique". In: 2014 International Conference on Recent Trends in Information Technology (ICRTIT), pp. 1–6. IEEE, April 2014.
[12] Dutta, C.B., Biswas, U. ."Specification based IDS for camouflaging wormhole attack in OLSR". In: 2015 23th Mediterranean Conference on Control and Automation (MED), pp. 960–966. IEEE, June 2015.
[13] S. Gadekar and S. Kadam, "Secure optimized link state routing (OLSR) protocol against node isolation attack", 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, 2017, pp. 684-687.
[14] INDRAUSHA RANI, V., REDDY, K. ."To Improve The Security Of OLSR Routing Protocol Based On Local Detection Of Link Spoofing". IJSEAT, North America, 5, pp. 652-655. jun. 2017.
[15] H. Priyadarshani et al., "Enhancing the security of OLSR protocol using reinforcement learning", 2017 National Information Technology Conference (NITC), Colombo, 2017, pp. 49-54.
[16] R. Bhuvaneswari and R. Ramachandran, "Comparative Analysis of E-OLSR Algorithm in the Presence of Routing Attacks in MANET", International Journal of Sensors, Wireless Communications and Control (2018) 8: 65.
[17] H. Kanagasundaram and A. Kathirvel, "EIMO-ESOLSR: energy efficient and security-based model for OLSR routing protocol in mobile ad-hoc network", in IET Communications, vol. 13, no. 5, pp. 553-559, 19 3 2019.
[18] Christof Paar and Jan Pelzl. 2009. "Understanding Cryptography: A Textbook for Students and Practitioners (1st. ed.)". Springer Publishing Company, Incorporated.
[19] Majed Alrowaithy and Nigel Thomas. 2019. "Investigating the Performance of C and C++ Cryptographic Libraries". In Proceedings of the 12th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2019). Association for Computing Machinery, New York, NY, USA, 167–170.
[20] ChaCha20 and Poly1305 for IETF Protocols. Y. Nir, A. Langley. May 2015. (Format: TXT, HTML) (Obsoleted by RFC8439) (Status: INFORMATIONAL) (DOI: 10.17487/RFC7539).
[21] Krovetz, Ted. (2006). "Message Authentication on 64-bit architectures". IACR Cryptology ePrint Archive. pp. 327-341, 2006.
[22] Kohno, T., Ferguson, N., & Schneier, B. (2010). "Cryptography engineering: Design principles and practical applications". Indianapolis, IN: Wiley Pub., Inc.
[23] Bernstein D.J. (2005), "The Poly1305-AES Message-Authentication Code". In: Gilbert H., Handschuh H. (eds) Fast Software Encryption. FSE 2005. Lecture Notes in Computer Science, vol 3557, pp. 32-49. Springer, Berlin, Heidelberg.
[24] Ted Krovetz, CSU Sacramento, Wei Dai, "VMAC: Message Authentication Code using Universal Hashing", Internet Draft, October 2007.
[25] Niels Ferguson. "Authentication weaknesses in GCM". Public Comments to NIST, 2005.
[26] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", May 2005(includes updates as of 10-06-2016: page II), NIST Special Publication 800-38B.
[27] eRightSoft. "MAC (Message Authentication Code) applied on files and typed-strings". https://www.erightsoft.com/help/HMACAlgo.htm.
[28] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", CRC Press, Inc., Boca Raton (1996).
[29] C. Baritel-Ruet, F. Dupressoir, P. Fouque and B. Gregoire, "Formal Security Proof of CMAC and Its Variants", 2018 IEEE 31st Computer Security Foundations Symposium (CSF), Oxford, 2018, pp. 91-104, doi: 10.1109/CSF.2018.00014.
[30] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <https://www.rfc-editor.org/info/rfc4493>.
[31] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.
[32] A. H. Soualfi, S. Agoujil and Y. Qaraai, "Performance Analysis of OLSR Protocol under MPR Attack in Progressive Size Grid MANET", 2019 International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 2019, pp. 1-5, doi: 10.1109/WINCOM47513.2019.8942557.

Authors

Abderrahim HAJJI SOUALFI

Ph.D Student

E3MI Team

Master's degree: Microelectronics Systems of Telecommunication and Industrial Computing

Ad Hoc Network, MANET routing protocols, Security.

Said AGOUJIL

Ph.D Professor

E3MI Team

Numerical Analysis, Linear Algebra, structured Matrices, Signal and image processing, Mobile Network Ad Hoc and DTN, E-learning, speech coding.

RESEARCH ARTICLE

How to cite this article:

Abderrahim HAJJI SOUALFI, Said AGOUJIL, "COLSR An Efficient Mechanism to Secure OLSR Protocol Against Multipoint Relay Attack", International Journal of Computer Networks and Applications (IJCNA), 8(5), PP: 671-681, 2021, DOI: 10.22247/ijcna/2021/209994.