Coping with Packet Replay Attacks in Wireless Networks

Zi Feng*, Jianxia Ning*, Ioannis Broustis*,
Konstantinos Pelechrinis‡, Srikanth V. Krishnamurthy*, Michalis Faloutsos *

*UC Riverside, ‡University of Pittsburgh

{*zfeng, jning, broustis* }@*cs.ucr.edu, kpele@pitt.edu,* {*krish, michalis*}@*cs.ucr.edu*

Abstract—In this paper, we consider a variant of packet replay attacks wherein, an attacker simply replays overheard frames as they are, or with minor manipulations in the packet header; we refer to this as the copycat attack. When routers forward such replayed packets, the levels of congestion and interference increase in large portions of the network. Our experiments indicate that even a single attacker can degrade the route throughput by up to 61%. While simple to use techniques such as digitally signing every packet can stem the dissemination of such packets, they are resource intense. Thus, we design a lightweight detection and prevention system, COPS (for Copycat Online Prevention System), that intelligently uses a combination of digital signatures and Bloom filters to cope with the attack. With our system, the task of identifying and discarding replayed packets is distributed across a plurality of nodes on a route. We implement COPS on real hardware and perform experiments on our 42 node wireless testbed. Our measurements indicate that COPS achieves its objective; it can efficiently contain the effects of replayed packets to a local neighborhood without incurring high resource consumption penalties. Specifically, we show that COPS reduces the route throughput degradation by up to 66%.

I. INTRODUCTION

A simple, yet effective strategy for wireless DoS is to replay locally overheard data packets. These packets are then carried by other forwarding nodes resulting in increased levels of congestion on a wider scale. There are variations of the attack, where either control or data packets are replayed. In this work, we are focus on adversaries that replay *data* packets either without modifying them or after manipulating their contents (typically the header); we refer to this attack as a copycat attack. The objective of the attacker is to make the packet to look like a legitimate unit avoiding at the same time detection. The intelligence of such an attack lies in convincing (i) the MAC level recipient(s) of a packet to accept and forward it and, (ii) the final destination into believing that this was a legitimately retransmitted packet and that no attack is being launched.

To illustrate this attack strategy, consider the simple topology depicted in Fig. 1. Here, Alice has established a 3-hop route to Bob via the relays R_1 and R_2. Without loss of generality, the attacker is in Alice's vicinity and overhears her packets. Let us examine the following two cases.

a. The attacker does not manipulate any packet contents: Let us assume that in the topology of Fig. 1, the PDR (Packet delivery Ratio) on all of the links is equal to 1. Alice first transmits packet P_1, which is successfully received by R_1 and overheard by the attacker. Thereafter, when Alice observes the medium to be idle, she transmits packet P_2 to relay R_1, which is also acknowledged. Relay R_1 will place P_1 and P_2 in its MAC output queue and will forward them to R_2 whenever it gains
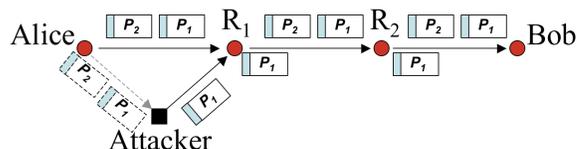
Fig. 1. The data copycat attack: Alice's packets towards Bob are replayed by the attacker; routers R_1 and R_2 forward the replayed packets.

access to the medium. When the attacker observes that relay R_1 has received P_2, the packet P_1 is replayed and received by R_1 again. This replayed packet has Alice's source address and appears to be new from the perspective of the MAC layer of R_1 (there are no pending ACKs for the replayed P_1). Hence, R_1 is deceived into forwarding it to R_2, *again*. Note here that in order for the attack to be successful, the attacker has to make sure that R_1 will not discard the replayed P_1 packet. This is possible only if R_1 successfully receives subsequent packets (e.g. P_2) from Alice. Otherwise, R_1 will assume that Alice has not received the ACK for P_1, and simply discard the replayed P_1 [1] (until a new packet such as P_2 is received). Note here that the attacker can temporarily store and replay packets that are quite old, thereby effectively circumventing this constraint.

b. The attacker edits the packet header: The above attack strategy can be easily detected. In particular, the original transmitter of the packet (e.g., Alice) can easily detect the malicious node, when overhearing a replayed packet by inspecting the source MAC address. In addition, even if the above is not possible in some topologies, an "unmatched" MAC layer ACK (e.g., sent from R1 to Alice due to the replayed packet from the attacker) will trigger the detection. Upon detection, this attack strategy can be overcome by using a Bloom filter to ascertain that a newly arriving packet has not been received in the past. To bypass such safety countermeasures, the attacker may slightly modify a packet in a way that it still looks legitimate. A simple approach would be to spoof the MAC address of a legitimate node [2] that is not part of the Alice-Bob route, and replace Alice's address with the spoofed address in the header of the data packet. R_1 and R_2 are thus misled into believing that this is a new legitimate packet and Alice cannot detect the attack.

Our measurements on a wireless testbed, show that the impact of the copycat attack on performance can be devastating. In particular, the existence of a single attacker can degrade the route throughput by up to 61%, while multiple attackers can further reduce the total network throughput.

The above effects can be completely overcome by using simple, previously proposed techniques. In particular, a **basic scheme** that utilizes digital signatures and Bloom filters can mitigate copycat attacks. Our prototype implementation, presented later, demonstrates the robustness of the basic scheme (up to 66% throughput improvement).

However, performing signature and filtering operations on each data packet introduces considerable processing overheads, especially at high data rates. This in turn, can significantly hit performance, especially under benign settings. Our measurements with different devices verify this fact, particularly when the available CPU and memory resources are limited. COPS balances the trade-off between the incurred processing overhead (due to the use of digital signatures) and the level of achieved protection against DoS, by only requiring signatures on a randomly chosen subset of packets. This subset is determined a priori by the source of the packets, and the processing load of the verification operations is distributed among all the nodes of a route. In brief, the main contributions of our work can be summarized in the following:

- We assess the trade-off between the processing overhead and attack resilience via extensive experimentation. Our experiments show that by signing just 40% of the packets the throughput on a 2-hop route is increased by up to 56% with one attacker and by up to 95% with two attackers.
- Based on the above assessments, we design COPS. COPS is an adaptive scheme that includes a detection and restrainer mechanism to counter copycat attacks. In brief, a very limited number of randomly generated packets are signed and verified in benign settings but upon the detection of a copycat attack, a more aggressive signing/verification policy is adopted.
- We implement and experiment with COPS on our indoor/outdoor 802.11 testbed. Our measurements show that our scheme can effectively contain the copycat attack.

The remainder of the paper is structured as follows. In section II we provide brief background on digital signatures and Bloom filters; we also discuss previous related studies. In section III we describe the attacker model and we quantify the impact of an attack. In section IV we present our measurement guidelines that lead to the design of COPS which is presented in section V. In section VI we evaluate the effectiveness of COPS. Our conclusions form section VII.

II. BACKGROUND AND RELATED WORK

In this section, we first provide brief background on packet authentication towards fighting DoS attacks. Subsequently, we discuss relevant previous studies.

A. Using Bloom filters and digital signatures

As discussed earlier, with the copycat attack the attacker can manipulate the headers of overheard packets, by using spoofed addresses. The use of digital signatures can help prevent the propagation of such manipulated packets. Routers may utilize Bloom filters to determine whether a newly arrived packet is original or replayed; while Bloom filters can catch packets that are replayed as is, they cannot catch manipulated packets. Thus, COPS intelligently employs both Bloom filtering and digital signing functionalities. We briefly explain the default operations of Bloom filters and digital signatures, in what follows.

Bloom filters in a nutshell: A Bloom filter is an array of bits of size m for representing a set $B = j_1, ..., j_n$; initially all the bits are set to 0. A Bloom filter uses k independent hash functions $f_1, ..., f_k$ with range $1, ..., m$ [3]. For every $i \in B$, the bits $f_i(j_i)$ are set to 1 for $1 \leq i \leq n$ if j is to be indexed using the filter. Although a location can be set to 1 multiple times, the first change is the only one that has an effect. To check whether indeed an element $a \in B$ is indexed, one needs to examine whether $f_i(a)$ are set to 1, $\forall i$. If not, then $a \notin B$. Otherwise, $a \in B$ with some probability. The probability of error is controlled by choosing an appropriate size for the data structure relative to the size of the set of elements to be represented. A detailed overview of how Bloom filters have been previously used in a plurality of networking problems can be found in [3]. In essence, each packet that is seen, is indexed using the filter. In other words, for a packet p, $f_i(p)$ is set to 1, $\forall i$. When a new packet p' is received, the filter checks to see if $f_i(p') = 1$, $\forall i$. If yes, it is classified to be a replayed packet.

The use of digital signatures: Digital signatures are used for authenticating the identity of message senders. Each node in the network has a private and a public key. The former is needed in order for a *digital signature* to be created, while the latter is used towards verifying this signature. The signature creation and verification operations typically use the Secure Hash Algorithm (SHA-1) [4], [5]. Since a private key is unique and held secret, attackers cannot reconstruct the same digital signature, unless they compromise the node. If a message is digitally signed, any change in the message will invalidate the signature. Any node that has the public key (typically made known) that corresponds to the signature of a received message can verify the message. In this work we assume that the identity of each legitimate node is a priori bound with a private and a public key by a trusted authority. Further details on signature procedures and algorithms can be found in [5].

B. Previous studies

In what follows, we discuss related studies on replay attacks.

Attacks on crypto-based key establishment: Such attacks target accessing secret information exchanged among legitimate nodes. The work in [6] describes an interesting classification of such attacks. Aura et al. [7] present a set of design principles to avoid replay attacks during crypto-based key establishment. Malladi et al. [8] propose a method that uses hashed values of random numbers in conjunction with the identities of all nodes to protect information. These approaches cannot mitigate copycat attacks, since data replaying takes place *after* secure communication establishment (perhaps achieved by the aforementioned schemes). In our work we assume that authenticated identity information exchange among legitimate nodes in the network has been established a priori.

Replay attacks related to wireless routing: Papadimitratos and Haas [9] present a secure route discovery protocol based on a message authentication code that can only be verified by the end nodes of a route. This, however, makes the protocol vulnerable to route and data frame replay attacks at intermediate nodes. Zhen and Srinivas [10] propose an approach to cope against a routing message replay attack that generates multiple, redundant RREQ packets. Winjum et al. [11] propose a scheme to address replay attacks in OLSR (Optimized Link State Routing protocol). However, none of the above approaches consider data replay or copycat attacks.

Replaying broadcast packets: Perrig et al. [12] propose a broadcast authentication protocol, TESLA, which uses one-way hash chains and delayed key releases to authenticate broadcast traffic. This idea has been utilized by various other studies, such

as [13], [14]. However, as discussed in [15], TESLA induces a security vulnerability: if a secret hash key is released before forwarding nodes authenticate a packet, the newly arrived packets signed by an attacker with this hash key will be falsely deemed legitimate.

Securing data transmissions: Although there exist previous studies on secure data transmissions, they are inadequate in mitigating copycat attacks. Heer et al. [16] provide an adaptive and lightweight security protocol, ALPHA. Before every new (typically large) data packet is to be routed, a small path reservation packet is sent to the final destination in order for all nodes on a path to examine the integrity of the (larger) data packet that will follow. However, since copycat attacks might not modify the contents of the overheard packets, ALPHA cannot efficiently block the propagation of replayed packets.

Using time-stamping and counters to address replay attacks: The use of crypto-synchronization has been widely used in CDMA/EVDO networks for protection against replay attacks [17]. Similar time-stamping strategies for mitigating replay attacks have also been proposed in [18] and [19]. Such techniques, however, require that nodes are very strictly synchronized, and this can typically be only achieved with specialized hardware, especially in dynamic environments.

Anti-replay techniques for wireline networks: There has been some work on replay packet detection in wireline networks. However, these studies do not take into account the inherent properties of the wireless medium. The IP security protocol (IPSec) [20] includes an optional technique for the detection of duplicate IP datagrams. Gouda et al. [21] propose a variation of the IPSec anti-replay mechanism. However, since IPSec establishes a shared symmetric key between the source and the destination, replayed packets are only detected by the end destination, i.e., after they have already travelled along the route. For the same reason, any end-to-end symmetric key based approach is inadequate. Our proposed framework adopts an asymmetric key cryptographic technique, based on digital signatures, as we discuss in the following section.

To the best of our knowledge, our study is the first to provide a complete and effective software framework to mitigate copycat attacks while inducing low processing overhead.

III. THE COPYCAT/REPLAY ATTACK

In this section, we begin with defining the attacker model that we consider in this study. Subsequently, we demonstrate how the attack can impact network performance.

A. The Attacker model

The goal of the copycat attack is to mislead routers into forwarding replicas of previously transmitted packets. However, the use of Bloom filters can effectively block the forwarding of packets. The attacker can bypass such blocking by slightly manipulating the packet header; with this, the Bloom filter decision engine will infer that the packet is new. However, if the manipulated packet is signed, the signature verification process at the next router will fail. Note also that Bloom filters are not of infinite size and thus, they are flushed as soon as they cannot store more information. Hence, a copycat attack may still bypass a Bloom filter.

In particular, we consider that the adversarial device has the following capabilities; these can be easily implemented in most commercial wireless cards nowadays :

- It has a wireless interface using which it can overhear packets . It stores the packets locally and retransmits them after an arbitrary time. For each overheard packet, the attacker decides randomly to either perform modifications in the packet header, or replay it without modifications.
- It has sufficient processing and memory capabilities, to store and process large volumes of packets.
- It is not an authenticated device, i.e., it does not have a private or a public key assigned by an authority. However, it can spoof the credentials of legitimate nodes.
- It can replay every packet an arbitrary number of times.
- It adheres to the 802.11 MAC protocol rules.

We assume that public and private keys needed for device authentication have been distributed before the application of our scheme. We also assume that two legitimate nodes can negotiate a shared secret key using their public/private keys, which allows them to communicate secret information.

B. Demonstrating the impact of the attack

Next, we present some of our testbed measurements that demonstrate the throughput degradation due to the data copycat attack.

Testbed description and experimental methodology: We conduct our experiments on our 42-node wireless testbed, which is deployed on the 3rd floor of the Engineering Building Unit II, at UC Riverside. The testbed configuration is such that the network consists of both indoor and outdoor links; we depict the layout in the left part of Fig. 2. The nodes are based on the Soekris net5501 hardware configuration [22], and run a Debian Linux distribution with kernel v2.6.16.19 over NFS. Each node is equipped with 500 MHz CPU, 512 Mbytes of RAM, and a WN-CM9 wireless mini-PCI card, which carries the AR5213 Atheros main chip. Every card is connected to a 5 dBi gain external omnidirectional antenna.

Our measurements encompass an exhaustive set of links and routes of different lengths. We experiment with both 802.11a and g modes of operation (unless otherwise stated our observations are consistent for both modes of operation). The experiments are performed late at night in order to avoid interference from co-located WLANs. All devices (legitimate nodes and attackers) set their transmission powers to 20 dBm.
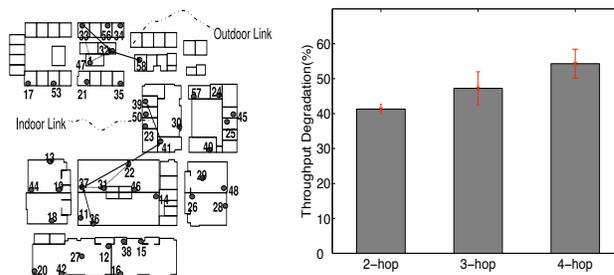

Fig. 2. The impact of the copycat attack. Our experimental testbed layout (left). The effect of the data copycat attack routes of different lengths (right).

In order to demonstrate the effectiveness of the considered attack strategy, we perform the following set of experiments. We consider different 2, 3 and 4 hop (overlapping) routes on our testbed and we measure the end-to-end performance degradation due to the presence of an attacker. We repeat each experiment 10 times and Fig. 2 (right part) depicts the average degradation observed, along with its standard deviation. The attacker is positioned such that it affects the first hop of the route (the effect of the attacker's position will be studied in the following sections). The attacker overhears packets for 120 seconds and replays them for the following 120 seconds. Our results show that the degradation can be as high as 54%. In addition, we observe that increasing the hop count on the route increases the degradation. This is an artifact of the attacker position; since the attack is at inception, the replayed packets traverse a larger portion of the network with longer routes, thus causing higher levels of degradation. Going forward, we examine the reasons behind the performance degradation, considering different settings, and we propose our countermeasure.

IV. DERIVING GUIDELINES FOR SYSTEM DESIGN

In this section, we first describe the implementation details of digital signatures and bloom filters; both are used in our system. Next, we describe experimental results with a baseline system where these features are employed to various extents. The results from these experiments provide insights that guide the design of our system, COPS.

A. Implementation details

Our implementation is built on a combination of the Linux Click modular router platform v1.6 [23], the OpenSSL library [24] and a C++ Bloom filter library from Google [25]. We implement five new Click elements: *AddSQN, AddSID, AddSign, CheckSign, Bloomfilter* and *CheckRand*. For each packet, the element *AddSQN* adds a 32-bit monotonically increasing sequence number; *AddSID* adds a 32-bit nonce, and *AddSign* adds a 48-Byte digital signature. The element *CheckSign* verifies the signature of each incoming packet. We use the SHA1 function and DSA_sign/DSA_verification functions from the OpenSSL library to generate the message digest of the data as well as to sign/verify the digest. The element *Bloomfilter* uses 8 hash functions to produce hash values for the tuple <SQN, SID> (SQN is the sequence number and SID is the source identifier) in every packet. The Bloom filter size is set by default to 200000 entries. The source node uses the *CheckRand* element to determine whether to sign the packet or not; receivers use this element to determine whether the incoming packet is on the list for verifying the signature. This function can be used to sign and verify only a fraction of the packets generated.

B. The basic system

We first consider a basic scheme that employs a combination of digital signatures and Bloom filters towards addressing copycat attacks. The scheme does not account for the security-performance tradeoff and its objective is to simply constrain the replayed packets to their local neighborhood; the scheme provides insights on the design of our adaptive approach later.

As an example we consider the 4 hop path 37-11-36-38-16, the 3 hop path 37-11-36-38 and the two hop path 37-11-36.

Source Authentication : The basic scheme authenticates the packets as follows. First an additional header field is inserted into each packet by the source; we call this the BS header. The field includes a sequence number (SQN), a nonce (SID) and a digital signature (SIG). The 32-bit sequence number (SQN) is assigned for each data packet sent by the source. When the SQN space is used out, the SQN wraps around. In the case where two packets with the same SQN arrive at a receiver it will not be able to determine if there is a copycat attack or not. Since we want the COPS header of each packet to be unique, we use a 32-bit randomly-generated nonce in conjunction with the SQN for every packet. With this, it is almost impossible for two or more packets to have the same pair of SQN and SID.

The source node also uses its private key to produce a digital signature (SIG), which it appends to the header. In order to render the signing process efficient, a one-way hash function is first used on the packet contents to generate a fixed-size bit string. The private key is then used to sign this returned string. We use the popular SHA-1 as the cryptographic hash function [4], and we use DSA as the digital signature algorithm. SHA-1 takes the SQN, SID and the payload of the packet as input and returns a 20-byte output . Then, DSA generates a 48-byte string by signing the 20-byte hashed data.

Packet validation: In order to prevent the propagation of replayed packets Bloom filters are employed. Each intermediate node R maintains a Bloom filter $Filter_{S,R}$ to keep track of data packets from every source S that uses R as a relay. Upon receiving a data packet from S, router R first checks the tuple <SQN, SID> of this packet by passing it to $Filter_{S,R}$. If the packet fails this check, R decides that the packet is replayed and discards it. Otherwise, R proceeds with authenticating the data packet. R uses the public key of S to verify the signature in the packet. If this packet fails the signature verification, R considers it to be a spoofed packet and drops it. Otherwise, R forwards the packet to the next hop. The Bloom filter is flushed as soon as no more information can be stored. The final destination of a data packet performs the same operations as the intermediate nodes; it conducts the two verification steps for each incoming data packet and only accepts a packet if it passes the verifications. The steps of this procedure are shown as a block diagram in Fig. 3.

Note here that the design of BS, as well as COPS, adopts the use of small-size Bloom filters (order of a few KBytes) in conjunction with digital signatures. This allows its applicability in networks with devices of limited memory, such as sensor motes. One could employ large-size Bloom filters (order or several Mbytes), which can store packets for much longer times.However, the memory and processing requirements would render this inapplicable in networks that consist of CPU and/or memory limited devices.

Processing Overhead: The default operations of BS, described above, provides an almost 100% warranty that replayed packets are blocked from propagating in the network. We have verified the effectiveness of these joint operations against data copycat attacks through extensive experiments on a 42-node wireless network discussed later in this section. However, our measurements also reveal that these operations induce considerable processing overheads; this is especially the case for packet sources digitally signing and verifying packets.
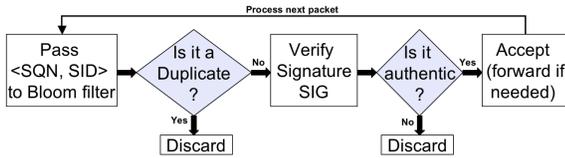
Fig. 3. Data forwarding at each router: the <SQN, SID> tuple of every data packet is checked by the Bloom filter, which is locally maintained at the router. If it passes the check, the SIG is verified using the public key of the source node. If either of the two verifications fails, the packet is dropped.

Soekris	500B/pkt	1000B/pkt	1500B/pkt
nosign	5793	3974	1817
sign10%	723	724	726
sign50%	155	155	154
sign80%	109	109	109
sign100%	69	69	68
Laptop	500B/pkt	1000B/pkt	1500B/pkt
nosign	31269	30460	16766
sign10%	7392	7242	7203
sign50%	2358	2469	2410
sign80%	1821	1802	1741
sign100%	1119	1115	1139

TABLE I

MAXIMUM DATA INJECTION RATES OF A SOURCE NODE ON A SOEKRIS NET5501 BOX OR A DELL LAPTOP WITH DIFFERENT PACKET SIZES.

To quantify the additional overhead that is imposed, we measure how quickly devices with different hardware and with fully-saturated traffic queues, can sign packets (as per the afore-described procedure) and inject them into the network. For this, we observe the ability of two different types of devices to sign and inject packets into the network. Specifically, we test: (a) a Soekris net5501 box [26] with 500 MHz CPU and 512 MB of RAM, and (b) a Dell laptop with 2.4 GHz CPU and 2 GB of RAM. We conduct experiments on 2-hop routes with each of the above devices; in every experiment we sign only a percentage of packets. We also consider various data packet sizes. We measure (a) how many packets/sec each device can inject into the network, and (b) how much time it takes to perform the signature and verification operations. Our measurements are tabulated in Tables I and II. We observe that the processing unit capabilities play a significant role in the ability of the device to inject traffic into the network. In particular we observe that more than a 10 fold reduction in the injection rates is possible in same cases, if a 100% of the packets are signed. This exorbitant processing penalty necessitates the design of a *lighter* scheme, which balances security and performance.

C. Evaluating the performance with the basic system

In the following we will present the evaluations of the BS. The insights from the experimental results drive the design of COPS, presented later.

Towards reducing the processing load due to signing each data packet, we experiment with slight variations of BS. In particular, we utilize *random signing* (RS). BS-RS performs the following actions:

- It determines a subset of packets that correspond to a certain percentage of the packets in the output queue of a source node; only these packets are actually signed.
- The verification load is then distributed among all the

Click + DSA signature	Average Process Time(sec)	
	sign	verify
Soekris	0.014644	0.019361
Laptop	0.000955	0.001508

TABLE II

AVERAGE PROCESS TIME OF SIGNING/VERIFYING A DSA SIGNATURE ON A SOEKRIS BOX AND THE A DELL LAPTOP.

nodes of a route. The set of packets that are verified at each node is decided a priori by the source node.

Before the nominal start of a traffic session, the source node generates (pseudo) random numbers using different seeds. The number of seeds is the same as the number of hops on the route. Packets whose SQN corresponds to one of the random numbers generated, will be signed. The source node sends the seeds to each node on the route separately. Note here that
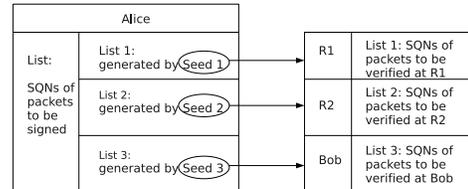

Fig. 4. Random packet signature scheme for the 3-hop route of Fig. 1. Alice has a List consisting of List1, List2 and List3; these are generated by seeds 1, 2, 3 respectively. The List will be the SQNs of the set of packets that are signed by Alice. She sends the three seeds to R_1, R_2 and Bob, respectively. Each of them now knows which packets are to be verified.

the seed information is encrypted by the pairwise secret keys between the source and the intermediate nodes, so that *only the legitimate nodes on the route know which packets are signed.* When a node R gets the seed from source node S, it generates a sequence of random numbers. Packets from S whose SQNs are on the list of this sequence are authenticated by node R.

Packets whose SQNs do not correspond to one of the random numbers have "dummy" signatures (perhaps a random number) inserted in place of real signatures; this prevents the copycat from knowing which packets are really signed and which are not. With this approach, generating and verifying packets becomes faster. Taking the 3-hop route of Fig. 1 as an example, Fig. 4 shows how the random packet signature functionality is executed. Note that this introduces a performance vs. security trade-off. If only a fraction of the packets are signed the attacker can effectively have some of the replayed packets forwarded. However, once a replayed packet is recognized, the attack is detected. A higher percentage of signed packets decreases the detection time, but increases the processing overhead.

In what follows, we discuss our experiments towards understanding BS's performance on our testbed.

Assessing the efficiency of BS-RS in scenarios with short routes: To begin with, we consider 2-hop routes and one attacker. The attacking device is located close to the source (we examine other cases later) and overhears packets for the first 120 sec. Subsequently it launches a copycat attack for another 120 sec. For each new packet, prior to transmission the attacker decides randomly on whether to modify the packet header or not. First, we measure the throughput of 12 different 2-hop flows (a) in benign conditions and (b) with the copycat attack. Fig. 5 shows the average percentage degradation in the end-to-end throughput for different versions of BS; as an example, sign50 indicates a BS-RS version where 50% of the packets are signed and verified. To compute the degradation, we

compare each throughput value with the throughput when (i) no attack occurs and (ii) the network is unprotected i.e., there is no overhead of any sort. We observe that when the attack takes place, if the network is unprotected the degradation in throughput can be as high as 54%. *Surprisingly, we find that signing and validating all the packets (a 100%) in this scenario degrades the throughput to a higher extent than if there was no protection, even under attack.* This is directly attributable to the processing overhead induced by the these operations. On two hop paths, the impact of the attack is constrained to a small portion of the network; the processing overhead incurred in order to prevent the replayed packets from traversing the second hop hurts the performance more than the attack itself. We observe that there is an inherent trade-off in terms of how many packets are signed and verified versus the reduction in degradation in the presence of the attacker. If fewer packets are signed, there is less overhead; however, more of the attacker's packets make it through; if more packets are signed fewer replayed packets are forwarded, but the processing increases. Fig. 5 suggests that BS-RS with 80% of the packets signed provides the best trade-off to yield the highest throughput when under attack; the throughput degradation is only about 5% in benign conditions and only about 20% in the presence of the attacker. Note that it is impossible to completely eliminate the impact of the attacker; the attacker's packets will always affect the local neighborhood. In essence, with protection from BS the attacker is reduced to a local jammer; anti-jamming is considered in [27], [28] and is not the focus of this work.

Experimenting with longer routes and more attackers: Next, we consider routes with length 3 and 4 hops. As before, the attackers overhear and store packets during the first 120 sec of traffic. During the following 120 sec, the attackers launch the copycat attack. First, we consider 12 such 3-hop flows, with one attacker per flow, who overhears and replays packets transmitted from the first *relay* node (near the second hop). In Fig. 6 we observe that in the presence of the attack, due to the trade-off discussed earlier between the processing overhead and the level of achieved protection against DoS, BS-RS with 50% of the packets signed has the lowest throughput degradation. Even in this case, signing and verifying all of the packets is not viable since it can lead to significant overhead penalties.

Next, we show experimental results with 4-hop routes; we also consider 2 active attackers (one of which is placed close to the source while the other is placed by the last hop) at the same time. Fig. 7 demonstrates that with two attackers launching copycat attacks at the same time and without applying the protection scheme, the degradation in throughput can be as high as 61%. The difference in throughput degradation with BS(-RS) is less pronounced as in the previous cases with 2 and 3 hop routes, with a single attacker (Fig. 5 and Fig. 6). This is because the attack in this case is more severe than in the previous cases because: (a) with two attackers the imposed levels of interference are higher and (b) the replayed packets traverse a longer 4-hop route and thus, impact the performance to a larger extent. Signing 40% of the packets provides the best performance versus security trade-offs in this case. We would like to emphasize that similar trends were observed for paths of different hop count and different attacker location as seen in Figs. (5)-(7).

Cases with attackers with differing signal qualities to the relays: Next we observe how the distance between the replay attackers and victim routes affects the network performance; the larger the distance the poorer the signal quality between the attacker and the victim route. While we have performed many experiments to validate our general findings, we present a specific sample scenario with a 4-hop route viz., 36→31→22→23→39. We enable fully-saturated UDP traffic from node 36 to node 39, and we activate an attacker at each of 3 different locations, namely A1, A2 and A3, as shown in Fig. 8; the measured RSSI values from the attacker, as measured at the victim relays are also shown. At each of these locations, the attacker overhears packets for 2 minutes and subsequently, replays them. Since signing 40% of the packets demonstrates good performance in all of our experiments, we choose this percentage. As we observe in Fig. 9, the percentage degradation in throughput is lower as the distance between the attacker and the victims increases. This is expected, since the PDR on the attacker's links decreases due to poorer link quality from increased distance. With this, there are two effects that act in conjunction. First, it becomes more difficult for the attacker to successfully overhear packets; second, replayed packets are not successfully received by the relay.

In the absence of any protection mechanism, when the attacker is close to the source (e.g. location A1 in Fig. 8), the copycat attack pushes a significant number of replayed packets along the route. However, when the attacker is closer to the destination (e.g. location A3), replayed packets travel only a few hops and, thus, the impact of the attack is not so prominent.

Assessing the impact of the copycat attack on the routing performance: The transmission of replayed packets increases the medium occupancy and this leaves less time for legitimate nodes to send packets. This can have a significant impact on the performance of routing operations, which depend on the broadcasting of routing control messages. To determine the effect of replay attacks on the performance of routing, we perform experiments with two routes that are active in parallel and are affected by two attackers. We also randomly choose 20 pairs of legitimate nodes on the testbed that wish to establish routes, using the DSR protocol [29]; note that these flows are not directly targeted by the attacker. Fig. 2 shows the flows being attacked for this experiment. Nodes 31 and 47 are the attackers; node 36 generates traffic towards node 39, while node 33 generates traffic towards node 58. We measure the route discovery time for each of the *other* 20 randomly selected pairs. Fig. 10 shows the cumulative distribution function (CDF) of the average route discovery time: (a) under benign conditions, where BS is not deployed, (b) under the presence of one attacker where BS is not deployed, and (c) under the presence of one attacker where BS-RS (with 40 % of the packets signed) is enabled. We observe that when the attacker is active, the route discovery time in the absence of BS is generally higher; this is because the copycat attack *indirectly* affects the routing performance of the other flows. Since the route discovery and route response packets, in many cases, have to traverse the areas congested due to the replayed packets, the latency incurred increases. With random signing, this problem is significantly

---

We present results for the Sign40 scheme only, since it provided the best performance among the different SignX schemes examined.
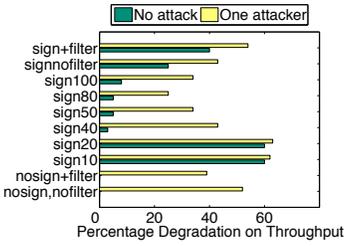
Fig. 5. Percentage of throughput degradation in 2-hop routes under various conditions.
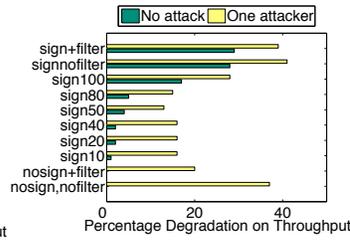

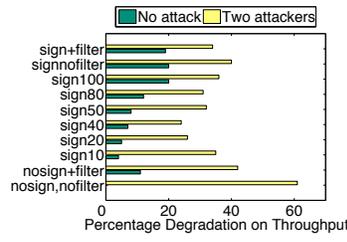Fig. 6. Percentage of throughput degradation in 3-hop routes under various conditions.

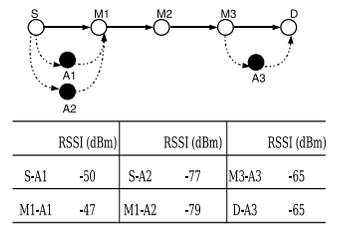Fig. 7. Percentage of throughput degradation in 4-hop routes under various conditions.

	RSSI (dBm)		RSSI (dBm)		RSSI (dBm)
S-A1	-50	S-A2	-77	M3-A3	-65
M1-A1	-47	M1-A2	-79	D-A3	-65

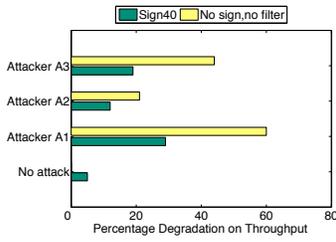Fig. 8. Experimental set-up for assessing the impact of the attack at various locations.

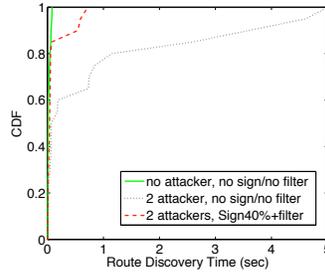Fig. 9. Percentage of throughput decrease for various attacker locations along a 4-hop path.

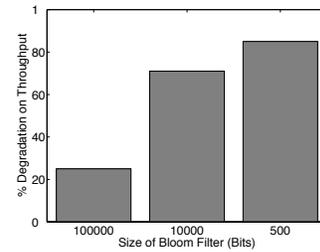Fig. 10. The CDF for the route request time based on our measurements on 20 pairs.

Fig. 11. The performance of COPS-Lite with Bloom filters of different sizes and hash functions.

alleviated. To illustrate this with an example, let us take a closer look at the route 44→57 (see Fig. 2). While node 31 is launching a copycat attack on the flow 36→39, the route query packets from 44 and the route response packets from 57 have to traverse regions that experience high levels of interference and congestion. In fact, when BS is disabled, the route request from node 44 has only a small chance to getting through to 57 before the route query times out. On the other hand, when BS is enabled, replayed packets are blocked to a large extent from traversing beyond node 22 and this alleviates the problem.

Experiments with various Bloom filter sizes: As one may expect, the size of the Bloom filter affects the potency of the attack. If the size is small, the filter has to be frequently flushed and with this, old packets that were overheard before the flushing will be considered as legitimate after the flushing. We perform experiments with different route lengths and Bloom filter sizes. We provide a representative example in Fig. 11, for a 3-hop route, wherein the attacker is located close to the source. Fig. 11 shows the throughput degradation with the Sign40 scheme, as compared to applying no protection. We observe that with the 100-Kbit filter, BS-RS provides the best performance; the attack degrades the throughput by 25% only. With filters of size 10-Kbit and 500-bit, many malicious packets go through, leading to higher levels of throughput degradation. This suggests that, depending on the available memory resources, long-size filters are preferable. Note however that if the attacker is able to store packets for very long periods of time, this may still be inadequate.

V. DESIGNING COPS

The experimental results presented earlier show that different variations of BS achieve different security levels and network performance. Due to the processing overheads of the core functionalities of BS, blindly applying them can have a negative impact on the network performance. In order to achieve the best trade-off between the processing overhead and the protection level against the replay attack, we utilize the understanding obtained from our experimental assessments and design COPS. In brief, COPS works as follows; in benign settings and COPS induces signature and verification of only a small fraction (e.g., 10%) of the packets. However, upon attack detection, the source is notified and the fraction of signed and verified packets is increased drastically. Note here that, COPS distributes the verification operations across the nodes along the route in order to reduce the verification load, in a way similar to BS. In the following we provide details on our design.

Detection mechanism: COPS incorporates a detection mechanism for both variations of the copycat attack. For the case where packets have not been manipulated by the attacker, the original transmitter can detect the attack relatively easily. Considering, without loss of generality, the topology in Fig. 1, once the attacker replays an unmodified packet, the attack can be detected with one of the two (or both) following ways:

- Alice overhears a packet that includes her credentials.
- Alice receives an unexpected/spurious MAC layer ACK (R1 upon reception of the replayed packet will transmit a MAC layer ACK). This helps if Alice cannot directly overhear the replayed packets.

When the attacker modifies the header of the packet (e.g., spoofs the source MAC address), the attack is detected by utilizing digital signatures; the attacker will not be able to sign the modified packets with a valid private key. If every packet is signed, then the attack is detected upon transmission of the first replayed-modified packet. However, as discussed earlier signing every packet induces significant processing overhead. Thus, COPS' detection scheme incorporates random signing i.e., only a randomly chosen fraction of the generated packets are digitally signed and a dummy signatures are inserted in the
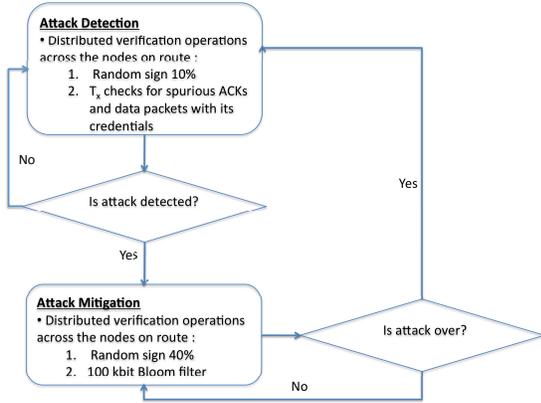
Fig. 12.   The operations of COPS.

other packets (as discussed). In this case, there is a delay introduced in detecting an attack; the attack is detected when the first "signed" packet is modified and replayed. Our measurements indicate that with sign10, the detection time is on average *3 sec*. Our results also indicate, as one might expect, that more aggressive signing schemes (e.g., sign50), reduce the detection time. However, overall, this reduction in detection time cannot compensate for the performance degradation induced by the processing overhead.

The attack mitigation mechanism: In a nutshell, by default (benign conditions are assumed) COPS uses random signing with 10% of packets being signed/verified. Every legitimate transmitter also keeps watching for spurious MAC layer ACKs and/or overheard replayed packets.

When an attack is detected, COPS performs the following steps: **(a)** increases the percentage of signed packets to 40% to contain the modified replayed packets, and **(b)** makes use of a 100 Kbit Bloom filter to contain replays of unmodified packets. Note here that, these values are derived based on our understanding from our previous experimental results. After applying the above countermeasures for restraining the effects of the attack, COPS continually monitors the network behavior for replayed packets. If no replayed packet seen for up to x seconds, COPS restores the default operations. Picking the value of x is not trivial; it heavily depends on the attack strategy. If we were to pick x to be small, the overhead due to signing higher percentages of the packets under benign settings would be "minimized". However, with an attacker who waits for some time between replayed packets transmissions, a small value would yield undesirable effects. On the contrary, picking a high value for x can increase the hit due to the processing overhead introduced by the digital signatures. In the current implementation of COPS, we pick $x = 20sec$ which we find to be a good value for all considered scenarios. A pictorial representation of COPS is in Fig. 12. Our experimental evaluations of COPS are in the following section.

VI. EVALUATING COPS

In this section we present our testbed measurements for evaluating the efficacy of COPS in dealing with replay attacks.

A. COPS through a lens

We first take a close look at the operations of COPS. For this, we demonstrate the benefits of COPS, via a representative experiment in Fig. 13; flow 36→39 is considered with 31 as the attacker (see Fig 2). The abscissa indicates the progression in time during the experiment and the ordinate represents a moving average of the end-to-end throughput measurements. Initially there is no attack. The source signs only 10 % of the packets; the signature is verified by various relays en route, distributively. At around 57 sec into the experiment, node 31 launches the attack. The attack is detected within about 3 sec; after this, the source is notified and it sends out a new message, requiring the verification and authentication of 40% of the messages and the activation of Bloom filter. From this point on, 40% of the packets are signed. Note from the figure that this adaptive version provides the best of both worlds; it provides good performance in benign settings is able to effectively alleviate the impact of an attack. Finally, the COPS returns to signing 10% of the packets and deactivates Bloom filtering if no malicious packets are caught for at least 20 seconds (Fig. 13).
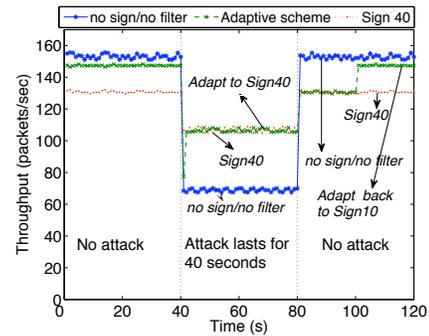

Fig. 13.   A real time trace of the throughput performance, in the presence of COPS.

B. Macroscopic view of COPS

In this section we present our macroscopic results for the efficiency of COPS. In particular, we perform experiments with (i) a large set of routes on our testbed and (ii) a variety of attack models. Each experiment lasts for 600 seconds and we account for different attack loads. Our metric of interest is the average performance degradation when using COPS as compared with the corresponding hit when using static approaches based on BS. In particular we compare COPS with sign10, sign40 and no countermeasure at all.

In a attack scenario corresponding to "attack X%", the attacker is active for X% of the total time. Table III shows the different time slots of the 600 seconds experiment duration, used from every attack model.

Benign settings: We observe from Figs. (14(a))-(14(e)) that in the absence of any attack, the degradation is the lowest when no protection scheme is being employed (as one might expect). In addition, COPS exhibits almost the same performance as with sign10 (recall that under benign conditions COPS randomly signs and verifies 10% of the packets, but does not employ Bloom filter). Employing sign40 and Bloom filters results in at least a 3x higher degradation as compared to COPS.
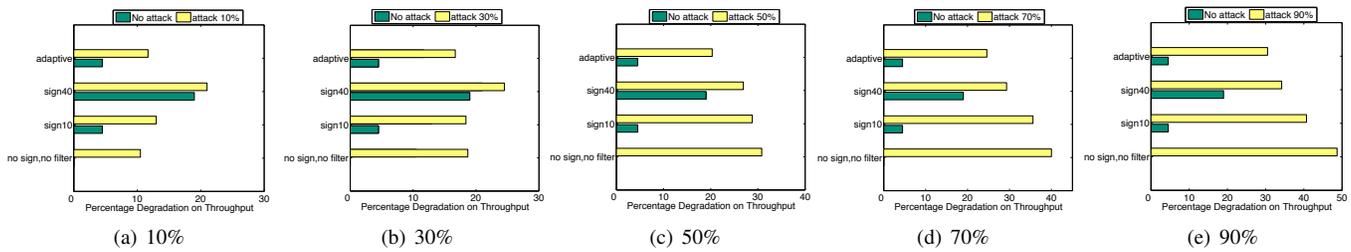
Fig. 14. Performance degradation comparison when using COPS for different attack loads.

Attack model	Time slots (secs)
10%	[60,180]
30%	[60,180], [540,60]
50%	[60, 180], [300, 360], [420, 540]
70%	[60,360], [420,540]
90%	[60, 600]

TABLE III
ATTACK LOAD DISTRIBUTION

Attack settings: In all considered scenarios, COPS delivers the best performance (the least throughput degradation), since it adapts to attack changes. Irrespective of the attack intensity, COPS maintains low overhead under benign conditions. A higher processing overhead is incurred only when under attack; in these scenarios, 40% of the packets are signed as this yielded the best overhead versus detection/mitigation efficiency as demonstrated in our prior experiments. Note here that, estimating the intensity of the attack on the fly and adapting the percentage of packets that are signed with a finer granularity is challenging; it is attacker specific and could increase the complexity of a detection/mitigation system. We will consider such possibilities in the future.

Figs. (14(a))-(14(e)) provide a gist of our experimental results under the copycat attack when COPS is employed. COPS provides a performance improvement of up to 66% (compared with the case with no security solution). As compared with the static schemes derived from BS, COPS provides an improvement of up to 45%.

VII. CONCLUSIONS

In this paper, we consider a variant of the data packet replay attack in wireless networks. Replayed packets increase the levels of contention and interference. An attacker can either replay packets as is, or modify the headers to create the illusion of new packets. Packet signature and verification operations and Bloom filters can help tag packets as authentic or replayed. However, our measurements on a large-scale testbed suggest that a naive application of these basic functionalities, wherein all packets are checked, project significant amounts of processing overhead. Towards reducing this processing overhead, while effectively mitigating the attack, we propose COPS, where (a) only a percentage of packets is signed, and (b) the load of verification operations is distributed along the nodes constituting the route. The fraction of packets that are signed and verified is adaptively varied based on the understanding gained from our measurements. Our experiments demonstrate that our system can efficiently address the attack in all the considered settings.

REFERENCES

[1] ANSI/IEEE 802.11-Standard. 1999 edition.
[2] J. Wright. Detecting wireless LAN MAC address spoofing. Technical report, 2003.
[3] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, pages 636–646, 2002.
[4] FIPS-180-1. In *NIST, FIPS PUB 180-1: Secure Hash Standard*, April 1995.
[5] FIPS-186-2. In *Digital Signature Standard (DSS), FIPS PUB 186-2*. U.S. Department of Commerce/National Institute of Standards and Technology, January 2000.
[6] P. Syverson. A Taxonomy of Replay Attacks. In *Naval Research Lab, Washington DC*, January 1994.
[7] T. Aura. Strategies against replay attacks. In *IEEE Computer Security Foundations Workshop*, pages 59–68. IEEE Computer Society Press, 1997.
[8] S. Malladi, J. Alves-Foss, and R. B. Heckendorn. On preventing replay attacks on security protocols. In *In Proc. International Conference on Security and Management*, pages 77–83. CSREA Press, 2002.
[9] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *CNDS*, pages 193–204, 2002.
[10] J. Zhen and S. Srinivas. Preventing Replay Attacks for Secure Routing in Ad Hoc Networks. In *Ad-Hoc, Mobile, and Wireless Networks*, pages 140–150. Springer Berlin/Heidelberg, February,2004.
[11] E. Winjum, A. M. Hegland, O. Kure, and P. Spilling. Replay Attacks in Mobile Wireless Ad Hoc Networks: Protecting the OLSR Protocol. In *Lecture Notes in Computer Science, ISSN 0302-9743*, 2005.
[12] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5:2002, 2002.
[13] Y. Yang, X. Wang, S. Zhu, and G. Cao. Sdap: A secure hop-by-hop data aggregation protocol for sensor networks. In *ACM MOBIHOC*, pages 356–367. ACM Press, 2006.
[14] S. Zhu et al. Lhap: a lightweight network access control protocol for ad hoc networks. In *Journ. of Ad Hoc Networks, 46 DRDC Ottawa TM*, 2006.
[15] Y. Huang, W. He, K. Nahrstedt, and W. C. Lee. Dos-resistant broadcast authentication protocol with low end-to-end delay. In *INFOCOM Workshops 2008, IEEE*, pages 1–6, April 2008.
[16] T. Heer et al. Alpha: an adaptive and lightweight protocol for hop-by-hop authentication. In *ACM CoNEXT*, 2008.
[17] S. Mizikovsky, Z. Wang, and H. Zhu. CDMA 1xEV-DO Security. In *Wiley Interscience, Bell Labs Technical Journal, 11(4), 291-305*, 2007.
[18] S. Pallicara et al. A Framework for Secure End-to-End Delivery of Messages in Publish/Subscribe Systems. In *IEEE/ACM International Conference on Grid Computing*, 2006.
[19] C. Adjih et al. Securing the OLSR Protocol. In *Med-Hoc-Net*, 2003.
[20] Security Architecture for the Internet Protocol. http://www.rfc-editor.org/rfc/rfc4301.txt.
[21] M. G. Gouda, C.-T. Huang, and E. Li. Anti-Replay Window Protocols for Secure IP. In *IEEE ICCCN*, 2000.
[22] UCR Wireless Testbed. http://networks.cs.ucr.edu/testbed.
[23] Click Modular Router. http://read.cs.ucla.edu/click/.
[24] The OpenSSL Project. http://www.openssl.org.
[25] C++ Bloom Filter Library. http://code.google.com/p/bloom.
[26] Soekris-net5501. http://www.soekris.com/net5501.htm.
[27] V. Navda et al. Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks. In *IEEE INFOCOM Miniconference*, 2007.
[28] R. Gummadi et al. Understanding and mitigating the impact of rf interference on 802.11 networks.
[29] D. B. Johnson et al. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *Ad Hoc Networking, Ch.5, pp. 139-172*, 2001.