Abstract—The packets in a mobile wireless ad-hoc network (MANET) are vulnerable to various packet-dropping attacks. Due to the lack of a centralized monitoring mechanism, it is a challenging problem to identify the attackers that launch the packet-dropping attacks in MANET. The existing DoS defensive techniques have not provided a scheme to efficiently and effectively solve this challenging problem. Hence we present a scheme, called CATCH, which constructs cryptographically-verifiable proofs for packet transmissions and identify the attackers by systematically investigating the packet transmission proofs from the nodes on a route. CATCH is a distributed scheme, and a source node in the network maintains the packet-dropping metric for every node on the routes to the destinations. The packet-dropping metric is computed based on the past packet forwarding history and provides an important information for a node to evaluate the reliability of a node in forwarding packets. We have evaluated the performance of CATCH protocol in terms of its accuracy of identifying packet droppers in simulated wireless ad-hoc networks using ns-2 network simulator under different network topologies, packet dropping rates, and mobility conditions. In these experiments, CATCH successfully identified more than 85% of packet droppers.

I. INTRODUCTION

Wireless ad-hoc networks (MANETs) have many applications, such as providing communication among a disaster relief team deployed to a place without a network infrastructure. In a MANET, a packet may traverse multiple hops until reaching its destination, making it vulnerable to various packet-dropping attacks. The packet droppers can censor the packets forwarded to them and drop the packets based on the packet content, or selectively drop the packets [1] based on the source and the destination of the packets.

Due to the lack of a centralized packet transmission monitoring mechanism and a reliable network infrastructure to support global monitoring, it is a challenging problem to identify malicious packet droppers. For instance, a malicious packet dropper may claim that it has transmitted all packets forwarded to it while in fact its downstream node receives none of these packets. A malicious packet dropper may also drop the acknowledgment packet of a received data packet, causing the source to resend the packet. It is difficult for the source to identify which nodes on the route to the destination are malicious droppers as every one of them can be a malicious node. Furthermore, wireless transmissions may fail due to the node mobility and poor channel condition, which adds to the difficulty of identifying the malicious nodes.

The existing DoS defensive techniques have not provided a scheme that can efficiently and effectively solve this challenging problem. The gray hole detection scheme in [1] relies on the cooperative nodes and probe packets to check whether a node is launching packet dropping attacks. The probe packets incur extra overhead and the results from cooperative nodes may not be trustworthy. The scheme in [2] utilizes the acknowledgment packets to detect the packet droppers under the assumption that the malicious packet droppers will not drop acknowledgment packets. A malicious node in wireless network may drop all types of packets, rendering this scheme less effective in real networks.

We present a scheme, called CATCH, which constructs cryptographically-verifiable proofs for packet transmissions and identify the packet-dropping attackers by systematically investigating the packet transmission proofs from the nodes on a route. CATCH requires a node to cache and provide proofs for every packet it forwarded. To reduce the network overhead of transmitting packet investigation requests and proofs, when replying an investigation request, a node aggregates all the proofs corresponding to an investigation, computes a hash of those proofs, and only returns the hashed value to the source. This proof aggregation mechanism saves the network bandwidth and increases the investigation efficiency. CATCH is a distributed scheme; a source node in the network independently computes the packet-dropping metric for the node on the route to the destinations. The packet-dropping metric is computed based on the past packet forwarding history and provides an important information for a node to evaluate the reliability of a node in forwarding packets. As it is unlikely to for the source to tell the difference between a malicious packet forwarder dropping a packet or the packet being lost due to wireless transmission errors, a node can also be regarded as a dropper by other nodes if it has bad wireless connections to other nodes.

CATCH provides very useful information for MANET applications. For instance, using our CATCH protocol, a routing protocol can identify nodes having a high packet-dropping metric and preclude these nodes from network routes. Thus, a malicious packet dropper will cause much less damages to the network communication.

The rest of the report is organized as follows. In section II, we describe the state-of-art packet dropping detection techniques and IP traceback techniques. Section III presents the packet transmission proof and the investigation mechanism of CATCH protocol. Section IV describes how the packet-dropping metric is computed. Section V presents the experiment results of CATCH protocol in simulated wireless ad-hoc networks under different network topologies, packet dropping rates, and mobility conditions. We will discuss relative issues and possible extension of CATCH in section VI and conclude our work in section VIII.
II. RELATED WORK

This section we review the existing work on detecting and defending the packet dropping attacks.

The scheme in [1] relies on probe packets and the cooperative nodes to detect whether a node is launching gray hole attack by selectively dropping packets. If the packet initiator finds that a packet sent to a cooperative node is not received by the cooperative node, the packet initiator increases the suspicion value of the node checked. The first problem of this scheme is that the cooperative nodes may be malicious and send bogus probe packet reception information to the packet initiator. Another problem is that sending probe packets consumes wireless network bandwidth and the packet dropper may forward the probe packets while selectively dropping the data packets from the initiator.

The scheme in [2] utilizes the acknowledgment packets to detect the packet droppers under the assumption that the malicious packet droppers will not drop acknowledgment packets. A malicious node in wireless network may drop all types of packets, rendering this scheme less effective in real networks. In addition, this scheme forms different groups on a route and the ACK packets transmitted among the groups incurs extra messaging overhead.

The REAct system [3] tries to identify individual malicious nodes who conduct packet drop attacks. In REAct, when a significant packet drop ratio is detected, packet drop ratio, the source node would cast a random audit request to ask for a behavioral proof of successful packet reception. Through this mechanism, malicious nodes are identified by the proofs provided by honest nodes. However, it assumes that there exist at least two independent paths between any pair of nodes in the network, and that a source node shares pairwise secret keys with the nodes in the source-destination node path. These assumptions would introduce high overhead when the network size gets large and not feasible in real network. In addition, REAct cannot detect colluding attackers while our detection system works no matter the attackers are colluding or not.

IP traceback technique traces the IP packets to identify the packet originators [4], [5], [6], [7], [8]. Savage et al. [4] proposed probabilistically marking the packets at IP routers and combining the marks in the packets to construct the path tracing back to the packet-flooding attackers. The FIT scheme proposed by Yaar et al. [6] improves the packet-marking-based tracing by exploiting the network topology information extracted from the packet markings. Compared with prior packet marking schemes, FIT is able to construct the path faster with fewer sampling packets. Belenky et al. [7] addressed the drawbacks of probabilistic packet marking schemes and proposed a deterministic solution. Their solution moved the responsibility of packet marking from distributed in all the routers to the edge router near the source. To a certain extent, it solved the problem that every packet may take different a path from the source to the destination so that the effectiveness of probabilistic packet marking schemes could be diminished. Rayanchu et al. extended the work in [7] to reduce false positives by applying multiple hash functions to the marks. However, these packet-marking method are not applicable to catching the packet droppers in wireless ad hoc networks. In the IP networks, the IP routers can be viewed as trustworthy. But in wireless ad-hoc networks, it is difficult to know which nodes are trustworthy. In addition, the packet marking schemes are used to find out the origins of flooding traffic, while our work focuses on the opposite side – catch the guys who “kill” network traffic. Consequently, a challenge is to use packet marking method of IP traceback in detecting packet droppers in wireless ad hoc networks is that malicious nodes may forge, remove the markings, and drop the packets altogether.

Instead of marking packets, the technique proposed by Snoeren et al. [5] stores a hash digest for a packet at the routers. If a packet is regarded as offensive, the routers can construct a path tracing back to the source of the packet based on the hash of the packet. Such technique is not directly usable in our scheme since it requires a node to trust the information from other nodes.

III. CATCH PROTOCOL

A wireless mobile ad-hoc network (MANET) consists of nodes that are wirelessly connected to each other and a packet from a source is forwarded by the intermediate nodes until reaching the destination. Figure 1 illustrates a packet transmission in a MANET.

The packet transmission in MANET relies on all nodes on the route from the source to the destination. Every node is effectively a router and the packet droppings of a node can cause the network disruption. The goal of CATCH is to identify packet droppers in a MANET. The packets could be dropped by malicious nodes or by a packet forwarder that has a poor wireless link to its downstream node. If the network has duty-cycling nodes that switch between power-saving mode (i.e., sleeping state) and active mode, a duty-cycling node will also drop packets if it is in power-saving mode when its upstream node forwards packets to it. Moreover, some nodes in
a wireless network may be “selfish” in that they only forward the packets from its “friends” while drop the packets from other nodes. In CATCH, a source treat all packet droppers indiscriminately since all packet droppings are detrimental to network communication and the source is unable to trust another node since this node may lie about the reason of it dropping packets.

Based on the ACK packet generated by the destination node, a source node in CATCH is able to know whether a packet is delivered or not. When a source node in CATCH detects a fraction of its packets are not delivered, it launches an investigation of the nodes on the packet forwarding route to identify the packet droppers.

First we define the adversary model and the assumptions used in this project. We assume that a malicious node drops packets in a random fashion only, with an arbitrary dropping probability. Furthermore, we assume a malicious node can not modify the content of a packet. We limit the power of malicious nodes to focus our work in this project on identifying packet droppers, and we will pursue possible extensions of this work in future work. Some additional assumptions are listed as follows:

1) Two nodes in the network share a pre-distributed pairwise symmetric key so that the messages between them can be authenticated by either party of them.
2) All wireless links in the network are symmetric; that is, node A can hear node B implies node B can hear node A.
3) In the network, the malicious nodes are assumed to account for minority.
4) The packet transmission and forwarding is based on source routing so the source node knows the whole route of a packet. DSR [9] is used as the routing protocol in this work; however, any source routing protocol would be compatible to our scheme.

We assume that there are pre-distributed shared keys in the network, but we do not give detailed mechanisms how this can be realized in real world. It is an important issue to find a way to compute and distribute the keys. In centralized network, this is not a big problem because we can reasonably assume the central entity has the power to generate, distribute, and maintain the keys for every node. On the other hand, in MANET, there might be no centralized controller. Asymmetric cryptography could be used in MANET for shared key setup, but it also brings higher computation overhead. Some previous research work [10], [11], [12] has been proposed to address this issue, and our system can utilize these work. In this project, we focus on detecting packet droppers in the network.

The notations used in this article are defined as follows.

- \( S \): the message source.
- \( D \): the message destination.
- \( seq \): the sequence number of a packet.
- \( K_{a,b} \): the shared secret key between node \( a \) and node \( b \).
- \( HMAC_k(M) \): the message authentication code (MAC) of message \( M \) generated using the HMAC-SHA1 algorithm and key \( k \).

### A. Packet Transmission Proof

In CATCH protocol, a packet forwarder obtains a transmission proof from its downstream node, which, if necessary, is presented to the source node to show that this forwarder has indeed forwarded the packet to the next hop. The details of packet transmission proof are described as follows. When an intermediate node \( I \) receives a packet, it forwards the packet to the next hop \( J \) based on the route information in the packet, and wait for the ACK (MAC layer ACK) from \( J \). If \( J \) successfully receives the packet, it replies the ACK to \( I \) with the packet transmission proof embedded in the ACK packet. The proof from \( J \) to \( I \) indicates that \( I \) has successfully delivered the packet to \( J \), which is stored by \( I \) in case the source investigates this packet in future.

Since the source may investigate every packet, an intermediate node has to store the proof of each packet, which is an overhead. Probabilistically storing the proofs may reduce the storage overhead but can lead to wrong investigation result if a node forwarded a packet but did not store the proof. In CATCH system, a node only stores the proofs within past \( T_{\text{proof}} \) seconds since a source will only investigate a packet that is generated less than \( T_{\text{proof}} \) seconds ago. The nodes in mobile ad-hoc networks are cellphones and laptops; the size of their storage is nowadays in GBs. Since each proof has a size of tens of bytes, storing the proofs should not consume too much of a node's storage space. In future work of this project, we may try to find ways to further reduce the proof storage overhead.

The format of the proof from node \( J \) to \( I \) is as follows.

\[
\text{Proof}_{\text{hop}} = \{ S, D, seq, I, HMAC_{K_{S,I}}(S, D, seq, I) \}
\]

The message authentication code in the proof is computed using the shared secret key between the source \( S \) and node \( J \). Therefore, unless \( I \) has indeed forwarded the packet to \( J \), \( I \) is unable to obtain the proof of the packet as it is unable to forge the message authentication code. In addition, since different sequence numbers yield different hashed values, \( I \) cannot forge a new proof based on the proofs it has seen before.

Once the destination node receives a packet, it would send an ACK for the data packet to the source node via the reverse route. Similar to the MAC layer ACK above, this \textit{end-to-end} ACK is also digitally signed by the destination \( D \) using the key shared between \( S \) and \( D \).

\[
\text{Proof}_{\text{end}} = \{ S, D, seq, HMAC_{K_{S,D}}(S, D, seq) \}
\]

Every node on the route receiving this ACK would also keep it as a proof. If the source does not receive the ACK in a given amount of time, it considers a packet/ACK loss event has occurred. If packet loss events happen too often, i.e., higher than a threshold, an investigation would be held by the source node to find out where the packets are lost/dropped.
Investigation:  
\[ \text{source} \quad \text{destination} \quad \text{start seq.} \quad \text{end seq.} \]

Reply:  
\[ \text{Hash(proofs)} \]

Fig. 2. Illustration of investigation request and the aggregated proof

B. Investigation

Based on the number of data packets generated and the number of ACKs received, a source node can detect whether the packet loss rate is higher than a threshold. If so, it starts an investigation to find out where the packets are dropped. During the investigation, the source requests the intermediate nodes on the route one by one to provide the proofs for the packets under investigation, starting from the downstream node of the source.

To reduce the message overhead, an investigation packet is sent to the nodes on a route to request them to present the proofs of a series of packets, specified by the starting sequence number and the ending sequence number of the packet. An investigation packet also includes the address of source and destination. Upon receiving an investigation request, an intermediate node \( J \) looks up its proof cache to search the corresponding proofs, concatenates them together, and then takes a hash on the concatenated string using the HMAC-SHA1 algorithm. The hash is computed using the shared key between \( S \) and \( J \) so that the source \( S \) can verify the validity of the proofs. Moreover, node \( J \) also embeds a bitmap in the investigation reply to indicate which packet proofs are included. Figure 2 illustrates the format of investigation packet and the investigation reply packet.

This aggregated investigation mechanism greatly reduces the network message overhead for sending the investigation packets and the investigation reply packets. For instance, to investigate 100 packets, the message overhead is only one investigation request and one investigation reply, which is 16 bytes and 33 bytes (160-bit hash string plus 100-bit bitmap), respectively.

After receiving an investigation reply from a node \( J \), source \( S \) verifies the validity of the proofs from \( J \) by computing the hash of the proofs of all packets indicated in the received bitmap. \( S \) is able to compute all proofs since it knows the shared secret keys with the nodes on the route. If the investigation reply from node \( J \) is invalid, \( S \) regards both \( J \) and its downstream node as suspicious nodes and increases their dropping metric. This is because the invalid proofs indicate that either \( J \) fabricated its proofs or its downstream node gave \( J \) invalid proofs. If the investigation reply from \( J \) is valid, \( S \) updates the dropping metric of \( J \) based on the number of packets forwarded by \( J \) and the number of non-forwarded packets. For a specific packet, if \( S \) receives the proofs from all upstream nodes of \( J \) but does not receive the proof from \( J \) and the downstream nodes of \( J \), then either \( J \) dropped the packet or its downstream neighbor did not provide a proof to \( J \). Based on this rationale, \( S \) increases the dropping metric of \( J \) and its downstream node. In addition, the source also utilizes the information of the nodes receiving the ACK from the destination to reduce the scope of possible droppers.

For instance, suppose the route from \( S \) to a destination \( D \) traverses \( A, B, \) and \( C \) in order. If \( B \) is unable to provide a proof for a packet whereas \( A \) is able to provide the proof of this packet, then the source increases the dropping metric of \( B \) and \( C \). If \( B \) provides the proof of receiving the ACK of a packet from the destination while \( A \) is unable to provide, then the source increases the dropping metric of \( A \) and \( B \).

By updating the dropping metric in this way, even though the dropping metric of the neighboring nodes of a dropper is also increased when the dropper drops a packet, the dropping metric of the dropper will be larger than that of other nodes since the dropping metric of the dropper increases every time it drops a packet where as the dropping metric of its neighboring nodes only increases when they are implicated by the dropper.

IV. THE PACKET-DROPPING METRIC

A. Basic Concept

A source node maintains a packet-dropping metric, or dropping metric in short, of the nodes on the routes, which is used by the source to evaluate the packet forwarding performance of other nodes. Since there is no trust relationship among nodes, a node does not share its dropping metric information with other nodes.

Whenever a node on the route to the destination is considered to have lost or dropped a packet or ACK, its packet-dropping metric would be incremented. On the other hand, on successful packet/ACK transmission, all the nodes on the route would get their metrics decreased. A node with high packet-dropping metric would be considered as a malicious node or bad node, i.e., a node running out of power or getting high interference nearby. If possible, a source node would not select a node with high metric when deciding the route.

B. Details

In order to maintain the dropping metric, a source node must create a mapping between the identifier of a node and its corresponding dropping metric. Here we assume that each node has a unique identifier, and each node can obtain the identifier of another node in the packet forwarding process. For simplicity, we use IP address as the identifier of a node and assume a nodes never changes its IP address.

At the beginning, a source node \( S \) would assign an initial metric \( m_{\text{init}} \) to all the node in the network. The metric is then updated based on the following exponentially weighted moving average formula:

\[
\text{metric} = \text{metric} \times (1 - \alpha) + \text{(metric change)} \times \alpha,
\]

where \( \alpha \) is the constant smoothing factor between 0 and 1, and it stands for the importance of the latest observation. Upon a successful packet delivery, the source decreases the dropping metric of every node on the route of the delivered packet; that is, the value of \( \text{metric change} \) would be \( m_{\text{dec}}. \)
which is negative. On the other hand, metric change would be a positive value $m_{inc}$ when the source discovers that a node is suspicious of dropping a packet. The minimum dropping metric of a node is configured as $m_{min}$. Setting a minimum dropping metric prevents a malicious node from performing as a normal node at beginning, accumulating a very low dropping metric, and dropping packets without being regarded as a packet dropper until its dropping metric is higher than other nodes.

Meanwhile, a source node also periodically decreases the dropping metric of the nodes so a node with high dropping metric will have smaller dropping metric if it no longer drops packets. This enables a node to begin utilizing previous packet droppers when they no longer drop packets.

V. EVALUATION

A. Implementation on Ns-2 Network Simulator

We have implemented the CATCH protocol on the ns-2 network simulator. The proof of data packet transmission and the proof of receiving the ACK from the destination are both generated and collected in the MAC layer. When a node receives a data packet from its upstream node, it generates a proof for the data packet received and piggybacks the proof in the MAC layer ACK packet to the upstream node. Also at the MAC layer, a node stores the ACK proof originated from the destination of a data packet.

The investigation mechanism is implemented in network layer based on the DSR codes, in particular. We choose DSR because a source in DSR routing protocol is aware of the route of a data packet, thereby enabling it to investigate the nodes on the route if necessary. The investigation is flow-based, meaning a source will investigate the nodes on a route if a certain fraction of the packets on this route are not acknowledged. In the simulations, a source launches an investigation when it detects 10% packets are not acknowledged by a destination.

In the simulations, the parameter $\alpha$ is configured as 0.2. The initialized metric $m_{init}$ of each node is set to $-1$. For a delivered packet, the parameter $m_{dec}$ is $-1$. For a failure of presenting the proof of a packet, the parameter $m_{inc}$ is 3. The minimum dropping metric $m_{min}$ of a node is $-1$ so a node will not have a dropping metric so small that enables it to drop many packets before being detected. We have tried different configurations of these parameters and did get different numerical dropping metric results. However, the droppers caught by the source were the same. The reason is, what matters of the dropping metrics is not their absolute numerical values themselves, but their relative relationship. The source regards the node with the highest dropping metric among a route segment as a dropper.

B. Chain Topology Simulation Results

First we evaluate the CATCH in a 10-node-chain topology, in which the leftmost node (node 1) is the source, the rightmost node is the destination (node 10), and the node 5 in the middle of the chain is the dropper. The nodes use random-waypoint mobility model and the moving speed of a node is 1m/s. Each experiment was conducted 10 times and the average results are presented.

Figure 3 shows the dropping metrics of the nodes when the dropping rate of the node 5 is 100%. The dropping metric of a node is normalized by adding 1 to it to make it easier to study the metric. Since node 5 dropped all packets, it is unable to present the packet proofs when it is under investigation. Node 6 ends up having a same dropping metric as that of node 5 because the source is unable to tell whether the failure of node 5 presenting a proof is due to node 5 dropping a packet or node 6 refusing to issue a proof to node 5.
C. Grid Topology Simulation Results

As shown in Figure 3, two nodes may end up having the same dropping metric even though only one of them is the real dropper. This section evaluates the false positive rate (FPR) and false negative rate (FNR) of the CATCH protocol in a 49-node grid network. The FPR measures the ratio of the number of false positives (i.e., nodes that are not the droppers but are regarded as the droppers by the sources) to the sum of the number of false positives and the number of non-droppers. The FNR measures the ratio of number of false negatives (i.e., nodes that are the droppers but are regarded as the non-droppers by the sources) to the sum of the number of false negatives and the number of real droppers.

The topology of the network is $7 \times 7$. There are 7 traffic flows and 7 droppers in the network. The node moving speed ranges from 1 m/s to 5 m/s. Each experiment was conducted 10 times and the average results are presented.

Figure 5 shows that the false negative rate is about 15% in the tested topology and as the nodes move faster, the false negative rate increases. Due to the mobility, the source may not be able to identify a dropper because the dropper moves to another traffic flow before the dropper’s dropping metric increases to the level that makes it distinguishable from non-droppers. When the droppers move in the network, they also cause the increase of the dropping metric of non-droppers, which also makes it harder to determine the droppers.

Figure 6 shows that the false positive rate is about 10% on average. The false positives are mainly due to that the packet dropings of the droppers cause the increase of the dropping metric of non-droppers. As nodes move in the network, source sometimes may mistake a non-dropper with high dropping metric as a dropper.

VI. DISCUSSION

In this project, we implemented the basic scheme mentioned in previous sections on ns-2 network simulator. However, additional features can be added to CATCH to improve the accuracy of dropper detection. Here we discuss one of these features, which could be an extension of our work in the future.

Dr. Wallach suggests an excellent idea of disambiguating the packet droppers by utilizing multiple routes to the destination. As source routing is used in CATCH, a source node can select any available routes returned from the route discovery (i.e., the routes in the route reply messages of DSR). When encountering some ambiguity in deciding who is the actual dropper on a route, the source can find another route to the destination by bypassing one of the suspects, if such route exists, then do the investigation and dropping metric updating again as in CATCH. This might give the source more knowledge to disambiguate who are misbehaving. For instance, if after bypassing a suspect, there is no packet dropping anymore, the source can increase the dropping metric of the suspicious node as it is likely this suspicious node has caused previous packet dropings.

Another insightful comment from Dr. Wallach is that after a source node finds out a misbehaving node, how can it leverage
this knowledge to improve the packet transmission quality or improve the network connectivity? Definitely, the source would not want to include the droppers on its routes in a period of time. However, should the source also drop packets to or from a packet dropper? As we mentioned before, there could be multiple reasons a node is regarded as a packet dropper. If it just gets bad channel condition in some direction or at some position during moving but has good link quality on other traffic flows, blocking traffic through it on other flows might impede network communication. In addition, the source who drops packets to/from a dropper (based on the source’s view) may also be regarded as a malicious dropper by other source nodes. Stopping the traffics flows to/from a node might end up with disrupting normal packet transmissions in the network. In our design, every source node maintains its own view of the dropping metric of nodes in the network. If a misbehaving node moves around and drops packets everywhere, it will eventually be detected by other source nodes and its damages are thus limited.

VII. ACKNOWLEDGMENT

We would like to thank Dr. Dan Wallach and Dr. Dave Johnson, for their insightful comments and excellent suggestions on improving our scheme.

VIII. CONCLUSION

Identifying a packet dropper in wireless ad-hoc network is a challenging research problem. In this report, we have presented the CATCH, a protocol that constructs cryptographically-verifiable proofs for packet transmissions and identify the packet droppers by systematically investigating the packet transmission proofs from the nodes on a route. CATCH is fully distributed as each node independently computes the dropping metric of other nodes. CATCH devises an aggregated investigation mechanism to control the investigation message overhead. We have implemented the CATCH protocol and evaluated its performance in simulated wireless ad-hoc networks using ns-2 network simulator under different network topologies, packet dropping rates, and mobility conditions. In these experiments, CATCH identified the droppers with a success rate about 85%.

REFERENCES