DoS-Resistant Broadcast Authentication Protocol with Low End-to-end Delay

Ying Huang, Wenbo He and Klara Nahrstedt
{huang23, wenbohe, klara}@cs.uiuc.edu
Department of Computer Science
University of Illinois at Urbana-Champaign
Siebel Center, 201 N. Goodwin Ave.
Urbana, IL 61801

Whay C. Lee
Whay.Lee@motorola.com
Networks and Systems Research Center
Motorola Labs
111 Locke Drive,
Marlborough, MA 01752-7214

***Abstract*—In mission-critical networks, command, alerts, and critical data are frequently broadcast over wireless networks. Broadcast traffic must be protected from malicious attacks, wherein sources are impersonated or broadcast packets are forged. Even though broadcast authentication eliminates such attacks, attackers can still launch Denial-of-Service attacks by injecting substantive false packets, which consume both communication and computation resources. Due to inevitable proliferation of duplicates of broadcast packets, it is especially important to limit false packet propagation range. Evidently, authenticating each packet before forwarding can effectively contain false packets within one hop. But it results in considerable end-to-end delay penalty on authentic packets. In this paper, we propose a randomized authentication scheme, DREAM, which contains most of false packets in one-hop range of attackers and yet keeps end-to-end delay relatively low. Dream also continuously monitors the contextual threat and dynamically adjusts the trade-off among containment and end-to-end delay performance. Extensive evaluations in ns2 validate our idea.**

I. Introduction

Mobile ad hoc wireless networks (MANET) are deployed for many mission-critical applications, such as military operations, emergency response and disaster recovery. Often operating in multi-hop and open wireless environment, they are vulnerable to various attacks, such as packet modification, impersonation, and Denial-of-Service (DoS) attacks. Thus, the primary concern is to consistently assure availability of communication resources, carefully grant authorized access and prevent malicious tamper on network function from outside the administrative domain.

Commands, alerts and data are frequently broadcast network-wide to relay time-sensitive information and support critical decision-making process. If broadcast traffic is not authenticated, sources can be impersonated and message content can be forged by adversaries and rivals. Misleading or false information could cause unnecessary operation or wrong decision. Thus, it is important and common to authenticate broadcast traffic.

However, without careful design, broadcast authentication may suffer from DoS attacks and negatively affect performance of broadcast protocol. In DoS attacks, attackers need neither to compromise legitimate devices nor to know trust and key information. They only flood faked and format-compatible packets so as to consume excessive bandwidth, computation and memory resources *network-wide*. Even though counterfeit packets are rejected after signature validation, they cause serious problems: (1) validating those packets wastes energy and CPU resources; (2) before they are verified, storing those packets temporally wastes buffer space; (3) transmitting those packets wastes limited wireless bandwidth. Therefore, broadcast authentication protocols must *contain forged messages* near attackers to prevent the rest of the network from infection.

A typical place to perform broadcast authentication is at the ends of communication, which means that forwarders do not check the integrity of packets. This broadcast authentication scheme residing at application layer can be considered as a scheme using "forward-first" policy: a node rebroadcasts each packet if needed and then delivers authentic ones to application after signature validation. Pure "forward-first" scheme misses the containment capability. A false packet is broadcast everywhere before its authenticity is verified. In order to contain false packets, an intuitive way is to apply hop-by-hop authentication scheme based on "authenticate-first" policy at routing layer: a node only rebroadcasts authentic messages after validation. In this way, false packets are filtered out at the first hop and devices outside the transmission range of attackers are immune. However, this hop-by-hop authentication scheme imposes remarkable penalty on end-to-end delay of legitimate traffic due to authentication delay at each intermediate hop. The accumulated delay postpones packet delivery to nodes far away from the sources and the maximal delay is proportional to network diameter in hops.

Public key scheme and one-way hash function scheme represented by TESLA [1] are two common cryptographic primitives for broadcast authentication (Other alternatives are extensively surveyed in [2]). TESLA is an efficient protocol that utilizes one-way hash chains and delayed key disclosure to authenticate broadcast traffic. But, TESLA incurs security vulnerability if used together with "authenticate-first" policy. This is because at the time forwarding nodes are able to authenticate a packet Msg, the secret hash key $hKey$ used to sign Msg is already released by the source. Afterwards, no nodes can trust any newly received packets signed by $hKey$. Hence, the intermediate hop has to resign Msg under his identity before forwarding. By manipulating resigning process, a compromised node could flood as many packets as it wants and viciously claim that those packets are sent by other innocent sources.

In this paper, we present a novel broadcast authentication scheme, called *DREAM*, an acronym for **D**oS-**R**esistant **E**fficient **A**uthentication **M**echanism. It effectively limits false data injection via frequently using "authenticate-first" policy based on public-key authentication. It also reduces the end-to-end delay by allowing a small percentage of unverified packets forwarded probabilistically via "forward-first" policy so that remote nodes obtain the broadcast messages quickly. Compared with most pertinent work considering containment of false broadcast injection [3][4][5], DREAM offers the following two advantages: (1) The remote nodes, who receive the unverified packets, are randomly determined for each packet. Hence, DREAM avoids a single point of failure and achieves load balancing. (2) In order to reduce end-to-end delay, not everyone in the neighborhood of a broadcast source has to forward unverified packets. Allowing only a small number of packets to reach remote regions is sufficient to reduce delay, while it effectively restricts contagious areas of false packets.

II. Design Goal and System Model

To guarantee that application receives only authentic packets, there is no doubt that every node needs to authenticate each packet. Resources of nodes in one-hop neighborhood of attackers cannot be protected unless they become inactive. Nevertheless, we want to eliminate faked packets as early as possible to protect resources far away from attackers and still keep average end-to-end delay low for authentic packets. Next, we present our network, broadcast traffic and attacker models.

A. Node and Network Model

We consider an ad hoc multi-hop wireless network supporting communication among a large number of devices. They may be mobile, but without rapid changes of network topology. These devices are all dispatched from a single administrative domain. Before deployment, each device receives digital certificates and necessary keying materials from a trusted server. These devices are thus able to establish pairwise trust relationship and shared secrets in field without contacting the remote trusted server.

B. Broadcast Model

All of the nodes participate in a predetermined broadcast protocol. The protocol can be in any form, ranging from flooding, probabilistic broadcast [6] to tree-base broadcast [7]. Although $DREAM$ is designed based on flooding, it works with other broadcast protocols by minor modification. Any device can originate broadcast messages. A broadcast message is uniquely defined by the ID of the originator and a sequence number. Upon message arrival, the recipients decide about duplication suppression, rebroadcast, and delivery to applications according to the broadcast protocol used. All broadcast messages are sent in clear text. Broadcast packets are authenticated before delivered to applications.

C. Attacker Model

Attackers are unaffiliated with the administrative domain, and they have neither valid certificates nor keys. Via snooping on wireless channels, attackers can gather the details of broadcast and authentication protocol. They intend to launch DoS attacks by flooding false or old packets. Since attackers do not have keys or certificates, they cannot produce valid signatures in false packets and those packets are dropped ultimately. Their common strategy is to let false packets permeate throughout the network as much as possible to aggressively consume communication and computation resources network-wide.

III. DREAM

In order to promptly contain injected false packets, we need to push public-key authentication from application layer down to network layer so that the broadcast protocol only allows authentic packets to be rebroadcast after validation. A hop-by-hop authentication scheme relying on "authenticate-first" policy perfectly contains false packets in one hop range of an attacker. However, it unfavorably imposes significant delay penalty on authentic traffic, especially for the traffic to the set of nodes far away from broadcast sources.

$DREAM$ addresses this end-to-end delay issue by relaxing containment requirement. The main idea is to allow a small and controlled number of packets transmitted to remote locations quickly without authentication in a probabilistic manner. A path segment over which packet P is not validated before forwarding is referred to as an unverified forwarding path for P. Nodes along the unverified forwarding paths forward packets before authenticating; while the other nodes, representing the majority of the network, authenticate packets before forwarding. Those unverified transmissions virtually reduce the network radius from the broadcast source and decrease end-to-end delay. [1]

Because of the diversified suppression policies of broadcast protocols, DREAM is integrated with broadcast protocol and independently runs at each device. The architecture of DREAM contains three modules as shown in Figure 1.

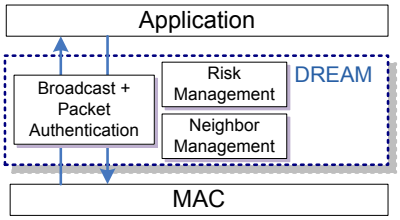

Fig. 1. DREAM Architecture

Packet Authentication Module is responsible for signing and validating packets. There are two operating modes yielding different delay and containment trade-offs.

Risk Management Module continuously monitors contextual threat. When evidence of false packet injection shows up, the module adjusts the operating mode in Packet Authentication Module to a more defensive and secure mode.

Neighbor Management Module periodically exchanges *hello* messages with one-hop neighbors. *Hello* messages not only indicate a node's liveness but also include a field specifying a node's one-hop neighborhood size. In order to prevent malicious tampering, hello messages are signed and verified.

Next two sections are devoted to Packet Authentication and Risk Management modules separately.

[1]This may introduce out-of-order packets with large delay variance. We assume that applications will reorder packets if in-order delivery is required.

A. Packet Authentication

Packet Authentication Module comprises two parts: (a) signing at sources and (b) verification and forwarding at receivers. Whenever a source sends out a broadcast packet, it signs the packet. Upon receiving a broadcast message, a node probabilistically determines to forward it first or to authenticate it first. When a node authenticates the packet first, we call the node an authenticating node. No matter which choice is made, each node will (a) only forward a unique broadcast packet at most once (it does not forward identified false packets); and (b) validate the packet and send the authentic one to applications. The message format is:

$$ID_{src}, Seqno, Msg, PubKeySign_{src}, HT, ID_{fwder}$$

, wherein

$$PubKeySign_{src} = Sign^{src}_{PubKey}(ID_{src}, Seqno, Msg)$$

ID_{src} and ID_{fwder} are IDs of the source and the last forwarder, respectively. When the source signs Msg, ID_{fwder} is set to ID_{src} since source is the last forwarder. $PubKeySign_{src}$ is the public key signature signed by the source and is never changed during forwarding process. HT is the number of **h**ops **t**raversed since the last authenticating node. It is reset to 0 at every authenticating node. Sources always set HT to 0. Each forwarder, who forwards the unverified copies, increases HT by 1. Here, we assume that receivers know the public key of the source; otherwise, sources' certificates should be broadcast as well.

Next, we will describe *packet verification and forwarding* algorithm at receiver sides based on flooding.

We reduce average end-to-end delay at the cost of imperfect containment. Therefore, it is crucial to carefully control unverified forwarding. We have the following requirements:

- The decision is independently made to avoid message negotiation;
- The decision is probabilistically made to achieve load balancing and to prevent a single point of failure;
- The number of hops that an unverified message is allowed to travel is controlled so that on one hand, broadcast packets spread out in space fast and on the other hand, false injected packets are contained near their originators;
- The number of "forward-first" nodes is kept small to reduce the cost of communication and public key computation wasted on false packets.

Public key signature verification is an expensive operation and usually takes time in seconds or tens of milliseconds in resource-constrained wireless mobile devices. Hence, a *verification queue* is placed in DREAM to buffer the packets waiting for signature verification. We assume for simplicity that at any moment, only one signature verification is performed so that spare CPU resources are reserved for other tasks. A *verification demon* process, whenever free, continuously monitors the verification queue. Once a packet is found at the head of the queue, the verification process removes the packet from queue, verifies its public key signature and relays the authentic one to applications. In addition, if the authentic packet has not been forwarded yet, it rebroadcasts the packet not under suppression. When the demon process completes processing a packet, it becomes free.

Upon receiving a broadcast message m, a node v probabilistically determines whether to forward m first or authenticate m first according to Algorithm 1. $Rand$ is a random number generated uniformly from [0, 1]. b, c and K are system parameters. b and c are the expected numbers of neighbors in the one-hop neighborhood of the source and the last forwarder $m.ID_{fwder}$ (other than the source) respectively, who forwards m first. K is the maximum number of hops that m is allowed to travel without verification.

Algorithm 1: Verification and Forwarding Algorithm

```
   input : An overheard broadcast message m
1  if (overheard messages with the same
   (m.ID_src, m.seqno) before) then return;
2  if (m.HT == 0 & m.ID_fwder is unknown neighbor)
   then return;
3  if (I am one-hop neighbor of m.ID_src) then
4  |  prob = b/ | Nbr(m.ID_fwder) |;
5  else
6  |  prob = 2 * c/ | Nbr(m.ID_fwder) |;
7  end
8  if (Rand > prob or m.HT == K) then
9  |  // authenticate m first;
10 |  m.HT = 0;
11 |  place m into verification queue;
12 else
13 |  // forward m first;
14 |  m.HT++;
15 |  rebroadcast m;
16 |  place m into verification queue
17 end
```

Algorithm 1 incorporates 2 steps:

(i) Suppression and Filtering (*Line 1-2*): If v has received at least one message with the same sequence number from the source before, message m is dropped. Thereby, a node forwards each broadcast message at most once. If v is one-hop away from the last authenticator, which is an unknown neighbor, message m is ignored due to lack of trust.

(ii) Probabilistic Pruning (*Line 3-17*): v decides whether to forward m first or authenticate m first probabilistically. $|Nbr(m.ID_{fwder})|$ is the neighborhood size of last forwarder $m.ID_{fwder}$. This value is available via *Neighbor Management module*. b is usually set to an appropriate value so that unverified broadcast messages can cover most directions (4 for instance). c is selected from [1, 2] to make the event that all the one-hop neighbors are pruned unlikely. Both b and c cannot be too large, because we need to control the number of unverified packets. The constant 2 before c in line 6 accounts for the fact that on average, half of the neighbors of the last forwarder have already heard m. Considering a case that m is forwarded from node A to v, via B, averagely half neighbors of B have already heard m broadcast by A, thus suppressing m according to line 1.

If m has already traversed K hops without verification, it is better to authenticate m first to confine its permeation in case of a false packet. If v decides to authenticate m first,

it resets HT field and places m in the verification queue. If v decides to forward m first, it increases HT field by 1, forwards m and then places m in the verification queue. The verification process is responsible for rebroadcast after signature verification. If $m.HT > 0$, verification process knows that m has already been forwarded once.

Since we use $|Nbr(m.ID_{fwder})|$ to make probabilistic forwarding decision (in Line 4 and 6), network topology should be relative stable so that during the Hello period, the neighbor information is accurate. Otherwise, inaccurate and outdated neighborhood sizes may result in more or less number of nodes forwarding the packet *first* from the last forwarder.

B. Risk Management

Nodes working as in previous section are said to be in *Normal Mode*. Via signature verification, they are able to detect emergence of false packet attacks if the number of received packets with invalid signature in a time interval exceeds a predetermined threshold. They switch to hop-by-hop authentication scheme and totally disable the "forward first" policy to contain false packet injection in a defensive way. They are then said to be in *Alert Mode*. Conversely, when the false packet attack lessens, nodes switch back to Normal Mode to trade for improved end-to-end delay. The transition between two modes is shown in Figure 2.

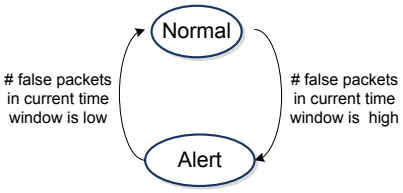

Fig. 2. Adaptation to Contextual Threat

Each node continuously monitors the number of detected false packets every $riskWindow$ interval. The node switches to Alert Mode if this number reaches α and switches back to Normal Mode if this number drops to β. α and $riskWindow$ are related to the tolerable percentage of CPU resource spent in evaluating false packets before switching to Alert Mode. Suppose the processing time to validate one public key signature is T_{val} seconds, a node can tolerate $\frac{T_{val}*\alpha}{riskWindow}$ CPU resource wasted on validating false packets. For mission critical networks, this percentage can be set to a small value. β is there to avoid frequent transitions and instability.

Switching to Alert Mode adversely increases end-to-end delay. However, we isolate the rest of network from infection and their computation and communication resources are protected. It is expected that when converging, only the nodes around the attackers enter Alert Mode. Other nodes can still receive the packets through alternative forwarding paths quickly.

IV. Evaluation

In this section, we evaluate the performance of DREAM in ns2 network simulator by comparing it with hop-by-hop authentication scheme and Dynamic Window (DW) scheme proposed in [3]. In Dynamic Window scheme, the forwarding decision is made by comparing the size of a locally maintained dynamic window on sensor nodes and the number of hops the incoming message traversing after its last authentication: if window size is larger, they use forwarding-first; otherwise, they use authentication-first. Additive Increase Multiplicative Decrease (AIMD) technique is used to dynamically manage the window: if an authentic message is received, window size increases; otherwise, window size decreases.

The criteria of our evaluation represents two aspects: penalty on authentic messages and containment capability of false messages. We have the following metric for legitimate packets:

- The average end-to-end delay: End-to-end (authentication) delay of packet m from broadcast source src at node v is defined as the interval between the moment src broadcasts m into wireless networks and the moment v finishes verification of m.

We have the following metrics for false packets:

- The number of nodes forwarding the false packets.
- The number of nodes receiving the false packets.
- The number of public key signature validation.

Public-key signature validating time is set to 0.5 second. At any moment, a device performs a single public key validation and all the other validation requests, up to 50, are queued. Public key signature has size 40 bytes. Each node is equipped with an omni-directional antenna operating on a single channel. The channel model is two-ray path-loss propagation model. The broadcast traffic is sent via CBR (constant bit rate) traffic with packet size 64 bytes in UDP. 802.11 DCF Medium Access Control protocol is used with default configuration. Transmission range is 250m.

We investigate two network topologies, i.e. grid topology and random topology.

A. Grid Topology

The first scenario we study is grid topology and each data point is averaged over 3 runs of simulations. Since delay penalty is critical in large scale networks with long diameter, we place 400 nodes in a grid. Each row (column) contains 20 nodes equally spaced with distance 240 meters apart. There is a single broadcast source located at the left top corner. It continuously sends CBR traffic at the rate of a packet every 2 seconds. The average network radius from the broadcast source is 10 hops based on the setting.

First, we study end-to-end authentication delay for legitimate traffic in absence of attackers. We vary K from 3 to 5 and c from 1.0 to 2.0. b is set equal to c. The length of $riskWindow$ is 6 seconds. $\alpha = 6$ and $\beta = 1$. For Dynamic Window Scheme, the initial window size is set to 64. Additive increase and multiplicative values are 1 and 2 separately, default as in [3].

As shown in Figure 6, the average end-to-end delay in hop-by-hop authentication is the worst, above 4.6 seconds because the average length of paths from the broadcast source is 10 hops and the public key validation delay is 0.5 second. Clearly, the average end-to-end delay in Dynamic Window scheme is optimal at about half a second. Because there are no false packets detected, the window size is at least 64, which is always greater than the number of hops traversed since last authenticator. The performance of DREAM is shown by the middle 4 lines. As c increases, the end-to-end delay decreases since more nodes are inclined to forward packets first, thus

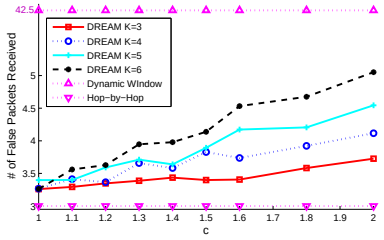

Fig. 3. Normalized # of False Packets Received

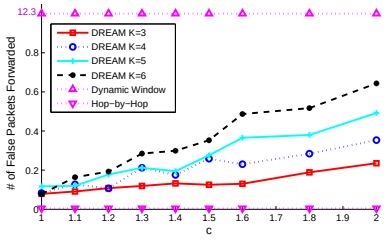

Fig. 4. Normalized # of False Packets Forwarded

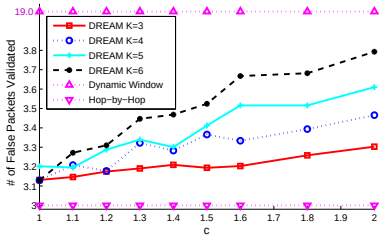

Fig. 5. Normalized # of False Packets Validated

virtually reducing the end-to-end path length. However, as c increases above 1.5, the degree of improvement levels off. When K increases, end-to-end delay decreases because distance between two successive authenticators is lengthened and the entire network is quickly covered.

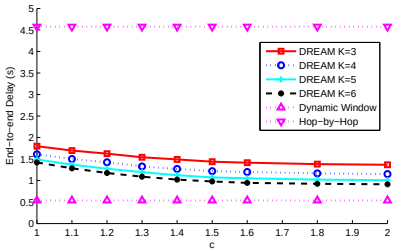

Fig. 6. End-to-End Delay

Next, we study the containment capability of DREAM and its response to malicious injection from Figure 3 to 5. Clearly, if attackers always flood less than α false packets in $riskWindow$ interval, DREAM never switches to Alert Mode. However, this is not the best interest for attackers. Therefore, we focus on cases of high-rate false injection. Again, we use the same configuration as before except that an attacker is flooding simultaneously near the source at the left top corner of grid. The attacker floods packets at the rate of 1 packet per second. During the false injection attack, it is out of question that all the neighboring nodes have to receive the false packets and validate them. But our scheme can effectively confine the false packets within a small number of nodes. The number of false packets is normalized by the total number of false packets sent by the attacker. The normalized number of received false pakets is below 45 for both DREAM and Dynamic Window scheme since they both switch to defensive mode. However, before the window size in DW is adjusted down, many false packets are broadcast to the whole network. Furthermore, the mixing traffics of good and false packets interdict the slowdown of dynamic window. For every packet sent by attacker, DW has 12 nodes to forward, 43 nodes to receive and 19 nodes to verify the message. On the contrary, the containment performance of DREAM is close to hop-by-hop authentication. The infection of false injection increases as c or K increases. However, waste on both transmission and computational resources by the false packets are under control.

The end-to-end authentication delay in presence of the attacker is shown in Figure 7. Delay for both DREAM and Dynamic Window scheme increases, compared with the case in absence of attackers.

From the above measurements, we see clear tradeoff between end-to-end authentication delay and containment capability, i.e. public key signature computational overhead and forwarding overhead for false packets.

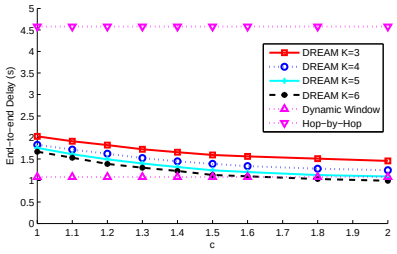

Fig. 7. End-to-End Delay

B. Random Topology

The second scenario we study is random topology with 400 nodes randomly placed in a 4000m*4000m network. We make sure that the resulting topology is almost connected. There are 1 broadcast source randomly selected to send CBR traffic at rate of 2 packets per 3 seconds and two attackers randomly selected to send CBR traffic at rate of 1 packets per 2 seconds each. Nodes switch to Alert Mode whenever the number of false packets in 6-second interval reaches 3 and falls back to Normal Mode when this number drops to 1. Since the processing time for one public key signature validation is 0.5 second, a node can tolerate at most $3 * 0.5/6 = 25\%$ CPU spend in evaluating false signatures.

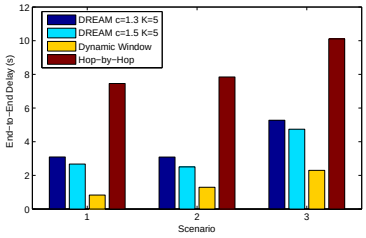

Fig. 8. End-to-End Delay

The end-to-end delay for two example configurations of DREAM is compared with the delay for Dynamic Window and hop-by-hop authentication schemes in Figure 8. X-axis shows 3 random scenarios with different location deployments. As usual, hop-by-hop authentication has the worst end-to-end delay for authentic traffic. DREAM with c set to 1.3 halves the end-to-end delay. Further improvement is present with c set to 1.5. Even though Dynamic Window Scheme has the best performance in terms of end-to-end delay, it is not resilient to false packet injection mixed with the legitimate traffics. As shown in Table I, for the total 330 false packets sent by attackers, Dynamic Window scheme forwards more than 5000 false packets in order to achieve the desired delay. But DREAM only forwards tens of false packets before nodes

around attackers switch to Alert Mode. Due to space limit, we do not place the results for the number of false packets received and validated here. Those two results are similar as the number of false packets being forwarded.

TABLE I
NORMALIZED # OF FALSE PACKETS FORWARDED

	Scenario I	Scenario II	Scenario III
DREAM c=1.3	28	12	12
DREAM c=1.5	30	25	32
Dynamic Window	6992	5768	14207
Hop-by-Hop	0	0	0

We show sensitivity to b in Figure 9, varying c and fixing K at 5 in absence of attackers. The same configuration as in Figure 8 is used. Increasing b decreases end-to-end delay, but without dramatic improvement. Possible reasons are lack of enough number of neighbors around broadcast sources or the set of non-authenticating forwarders are not spread out enough in all directions.

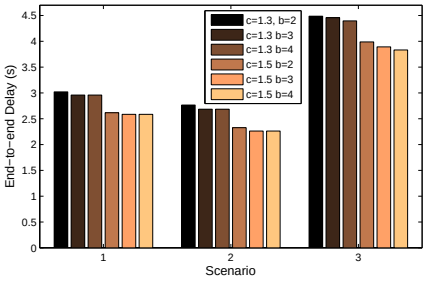

Fig. 9. End-to-End Delay

V. RELATED WORK

Wang, Du and Ning proposed a dynamic window scheme to contain bogus data by public key cryptography (PKC) authentication [3], where sensor nodes determine whether to verify a message first or forward the message first by estimating their distance from the malicious attackers and how many hops the incoming message has passed without authentication. As we show in the evaluation section, despite the fact that Dynamic Window scheme displays good potential to improve end-to-end delay of authentic traffic, it is not robust against "smart" attackers who maliciously manipulate AIMD scheme. Additional, during the decreasing process of AIMD, a large number of false packets are broadcast to the whole network. Drissi and Gu considered a large sensor network with a few predetermined trusted and better secured nodes which divides the whole network into subnets [5][4] . Trusted nodes use TESLA to broadcast messages to its subnet and each ordinary node, upon receiving broadcast messages from its subnet, rebroadcasts them before validation. Trusted nodes exchange broadcast messages among themselves. A source other than the trusted nodes sends messages to a nearby trusted node. However, the predetermined set of "trusted" nodes may attract attacks and too much authentication and coordination overhead consumes their battery and the battery of nodes around them quickly. Besides, how messages are sent from source to trusted nodes, from trusted nodes to trusted nodes and from trusted nodes to subnets may introduce additional vulnerability, such as DoS attacks and jamming. [8] proposed an efficient, one-time signature-based broadcast authentication scheme that reduces storage usage and includes a re-keying mechanism and [9] introduced a novel cryptographic paradigm of broadcast authentication with "preferred" verifiers. But, none of them offers containment capability against false injection attacks.

VI. CONCLUSION

In this paper, we propose a novel broadcast authentication scheme, which confines false packet injection mostly inside one-hop neighborhood around attackers with improved end-to-end delay for legitimate traffic. Via frequently using "authenticate-first" policy, DREAM contains most of faked packets in one-hop neighborhood of attackers. Probabilistic "forward-first" decision dramatically improves the end-to-end delay for nodes far away from the broadcast source. With dynamic adaptation to changing threat level, our scheme is flexible to trade off delay with containment capability. Extensive simulation verifies end-to-end delay and containment performance in DREAM.

In the future, we plan to integrate DREAM with other broadcast protocols and test its performance. DREAM relies on the probabilistic decision to push unverified packets remotely. But the unverified forwarding path may not be optimal for quick coverage. We will apply GPSR routing protocol [10] to deterministically forward the unverified packets further along the best direction. We will also investigate counter-measurement for attacks on broadcast suppression, wherein attackers pretend to be the broadcast source and send packets with higher or equal sequence numbers before the true source.

ACKNOWLEDGEMENT

The research in this paper is supported by Motorola grant 1-557641-239016-191100.

REFERENCES

[1] A. Perrig, R. Canetti, D. Tygar, and D. Song, "The tesla broadcast authentication protocol," 2002. [Online]. Available: citeseer.ist.psu.edu/perrig02tesla.html

[2] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical broadcast authentication in sensor networks," in *MOBIQUITOUS '05: Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2005.

[3] R. Wang, W. Du, and P. Ning, "Containing denial-of-service attacks in broadcast authentication in sensor networks," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, 2007, pp. 71–79.

[4] Q. Gu and J. Drissi, "Dominating set based overhead reduction for broadcast authentication in large sensor networks," in *ICNS '07: Proceedings of the Third International Conference on Networking and Services*. Washington, DC, USA: IEEE Computer Society, 2007, p. 81.

[5] J. Drissi and Q. Gu, "Localized broadcast authentication in large sensor networks," in *ICNS '06: Proceedings of the International conference on Networking and Services*, 2006, p. 25.

[6] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2003)*.

[7] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *INFOCOM (2)*, 2000, pp. 585–594.

[8] S.-M. Chang, S. Shieh, W. W. Lin, and C.-M. Hsieh, "An efficient broadcast authentication scheme in wireless sensor networks," in *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 2006, pp. 311–320.

[9] M. Ramkumar, "Broadcast authentication with hashed random preloaded subsets," in *ePrint Archive, 2005.*

[10] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*.