AuCRB: An Efficient Mechanism to Provide Availability, Reliability and Authentication for Multihop Broadcasting in Wireless Networks

Erman Ayday
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250, US
Email: erman@ece.gatech.edu

Farshid Delgosha
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250, US
Email: farshid@ece.gatech.edu

Faramarz Fekri
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250, US
Email: fekri@ece.gatech.edu

***Abstract*—This paper proposes a reliable and secure broadcast protocol for ad hoc wireless networks. Since coding and security compete for the same resources, we jointly solve for reliability, availability and integrity for a broadcast scenario. Packets sent by the source node would travel in a hop-by-hop fashion to the other nodes. Hence, it is critical to reduce the number of transmissions and latency. We assume Byzantine attacks in which the adversary can drop (or modify) legitimate packets and inject its own packets via several insider nodes. We require that the source data is reached to all legitimate nodes in the presence of any number of colluding Byzantine attackers as long as the legitimate nodes are connected. We also require that each receiver node in the network to be equipped with a mechanism to verify the source node and the integrity of the received packets using limited cryptographic primitives. It is essential that every node receiving a malicious packet immediately filters it out and uses only the legitimate ones for forwarding to the next hop and decoding. Designing a broadcasting mechanism that satisfies all the above requirements is a very challenging problem. We develop an authentication scheme, using a reliable and energy-efficient broadcasting protocol called *Collaborative Rateless Broadcast* (CRBcast) and limited cryptographic primitives. On contrary to the previous schemes, our scheme is resilient with respect to Byzantine failures as well as routing and flooding attacks and protocol exploits. Moreover, we compared our scheme with the previously proposed broadcast authentication schemes and showed that our scheme outperforms them in terms of efficiency. This is a crucial improvement over the previous schemes that ensure availability by flooding, but with very large communication overhead and latency.**

I. Introduction

Efficient network-wide broadcasting is an important issue in wireless networks that attracted a lot of attention. Some important factors that influence the efficiency of a broadcasting scheme can be listed as following. Reliability, defined as the percentage of nodes in the network that are able to retrieve the data, energy-efficiency, complexity, scalability, and latency. Based on the application, some factors might be more important than others. For example, for updating the software in all the nodes in the network, reliability is very important, while latency might have less importance. Broadcasting streaming media is a case where latency is of paramount importance. Energy is usually an important issue especially for battery constrained networks. In this paper, we consider the case that a large amount of packets (of order 1000 or more) have to be broadcasted in a multihop wireless network while requiring robustness against Byzantine attacks, reliability, energy-efficiency and low latency.

We will consider Byzantine attackers (insider adversarial nodes with the same authority as any other legitimate node). Typical attacks the insider attackers may launch to interfere with the normal operation of a broadcast protocol are:

- *Data Drop:* An insider node drops a legitimate report on the forwarding path toward the sink.
- *Bogus Packet Injection* and *Packet Modification:* The adversary injects bogus packets or modifies the contents of legitimate reports.

Cryptographic services required to prevent these attacks are *data availability* and *data authenticity*, respectively. We note that, routing attacks via colluding multiple Byzantine nodes cannot be mounted on the proposed scheme. In our scheme, nodes broadcast the packets to their neighbors. Hence, we are not using any fixed routing. As a result, our scheme is not vulnerable to the routing attacks.

In this paper, we design a scheme, called *Authenticated Collaborative Rateless Broadcast* (AuCRB), that provides the aforementioned security services with moderate communication and computation overheads. We propose to combine rateless information delivery mechanism [1], [2] with probabilistic relaying to develop one-to-many multi-hop communication protocols for reliable and time-critical content delivery in ad hoc networks. Furthermore, we require that our broadcasting protocol meet the low power, low memory, and low processing requirements of devices while introducing a minimum latency.

The organization of the paper is as follows. In the rest of this section, we summarize the related work in broadcast authentication for wireless networks and present

the notation used throughout the paper. In Section II, we briefly review the mathematical and cryptographic primitives used in this paper and also give a brief description of CRBcast protocol. The detailed description of AuCRB is provided in Section III. We analyze the security of our scheme against possible attacks in Section IV. The performance analysis of the proposed scheme and comparison to related work are studied in Section V. Finally, the concluding remarks are provided in Section VI.

A. Related Work

Based on the main cryptographic primitives employed, we can classify previously proposed broadcast authentication schemes into three groups based on the main cryptographic primitive employed: (1) message authentication code (MAC), (2) signature amortization and (3) one-time signature.

Protocols in the first group are TESLA [3], its simplified version for resource limited networks μTESLA [4], and the enhancements of μTESLA such as [5]. These schemes provide broadcast authentication by using MACs and require time synchronization between the nodes and the sink. This requirement is an implementation hurdle for multi-hop broadcasting in densely deployed networks. Moreover, these schemes are vulnerable to flooding attack. Another shortcoming of μTESLA is the difficulty of establishing the initial trust between the nodes and the sink.

Schemes in the second group of broadcast authentication protocols employ signature amortization. One of the first protocols in this group is SAIDA [6]. This protocol is not robust against false packet injection and packet modification attacks. The designers of SAIDA have proposed using Reed Solomon codes to handle the packet modification attack. However, this kind of coding is too complex for the low-power processor of the nodes. A similar approach is proposed in [7], which is too complex for multihop broadcasting in wireless networks.

The one-time signature BiBa and an improvement of BiBa, called HORS, [8] are among the schemes in the third group. The major drawback of using one-time signature schemes in wireless networks is that the public key has to be frequently updated to maintain security. This requirement significantly adds to the communication overhead of the protocol. Moreover, broadcast authentication schemes based on one-time signatures are not suitable for designing node-to-network multi-hop broadcast protocols. Broadcast communications of any node has to be handled by the sink as an intermediary.

A recently proposed scheme [9] proposes a simple method to secure the deluge network programming. In Section V, we compared the efficiency of AuCRB with this scheme and showed that our scheme has considerable advantages over this recent approach.

To sum up, we claim that, neither of the previous schemes consider data availability in an efficient fashion. Although all these schemes are focused on efficient authentication, the communication efficiency and the data availability are largely ignored. Hence, in this work, we include both data availability and authenticity to design an efficient scheme. Moreover, latency has never been considered by the previous works when defining the data availability. However, we define data availability based on the latency, which is a crucial requirement.

B. Contributions of This Paper

One of the strict requirements is to provide availability of data for legitimate nodes as long as they are connected to the source. To achieve the above in the presence of attackers, one may resort to use flooding with some authentication mechanism. However, flooding is very inefficient. This paper exploits rateless coding to achieve the same with low communication overhead and reduced latency in the presence of attacks.

The main contributions of our scheme are summarized in the following.

1) AuCRB is designed based on a broadcast protocol using rateless coding. Hence, it benefits from the same low computation and communication overheads.
2) Nodes individually authenticate each received packet instead of waiting for several packets to perform authentication. Therefore, the receivers can immediately filter out bogus packets and save energy.
3) Rateless coding intrinsically provides data availability by the loss recovery of the coding mechanism.
4) Authentication information transmitted by the source can be used to detect malicious nodes in the network.
5) The scheme ensures availability of data with very low latency in the presence of the malicious nodes (as long as the network is connected). This is a substantial improvement upon similar scheme that ensures availability by flooding but with very large communication overhead and latency.

C. Network Model

We assume the nodes are uniformly and randomly deployed in the field. In addition, they form a connected stationary network. This key pre-distribution is required to prevent node impersonation attacks. We assume that the source node is trustworthy. We assume the existence of an underlying Medium Access Control (MAC) protocol for the channel access as in [11]. Although one of the advantages of our scheme is to provide reliability in the presence of lossy packets using its built in erasure coding mechanism, we considered lossless links between any two neighboring nodes for the simplicity of discussion and simulations. Even though it is not the main contribution of our paper, with the use of pairwise keys, AuCRB can prevent the node impersonation attack relevant to the proposed broadcasting scheme as discussed in IV. Hence, we assume that, nodes have pairwise keys with their neighbors by using a key pre-distribution scheme such as the one in [10].

D. Notations

In order to facilitate future references, frequently used notations are listed below with their meanings.

N	Total number of nodes in the network
p	Probability of forwarding in *phase I*
t	Number of data packets to be sent from the source
T	Number of encoded packets generated after rateless encoding
ℓ	Number of partitions in the second phase
P_i	The i-th encoded packet during *phase I*
Q_i	The i-th encoded packet during *phase II*
G_i	The i-th partition during *phase II*

II. Technical Background in Context

A. Bloom Filter

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries [12]. A Bloom filter for representing a set U of T elements is described by an array of m bits, initially all set to 0. It employs k independent hash functions $H_1, \ldots, H_k$ with range $\{1, \ldots, m\}$. For every element $x \in U$, the bits $H_1(x), \ldots, H_k(x)$ in the array are set to 1. A location can be set to 1 multiple times, but only the first change has an effect. To check if y belongs to U, we check whether all $H_1(y), \ldots, H_k(y)$ are set to 1. If not, y definitely does not belong to U. Otherwise, we assume $y \in U$ although this may be wrong with some probability. Hence, a Bloom filter may yield a *false positive* where it suggests that y is in U even though it is not.

The probability of false positive is an important parameter in a Bloom filter. After all elements of U are hashed into the filter, the probability that a specific bit is 0 is

$$\left(1 - \frac{1}{m}\right)^{kT} \approx e^{-kT/m}. \tag{1}$$

Hence, the probability of false positive is

$$\begin{aligned} \tilde{p} &= \left(1 - \left(1 - \frac{1}{m}\right)^{kT}\right)^k \\ &\approx \left(1 - e^{-kT/m}\right)^k. \end{aligned} \tag{2}$$

As an example, assume the number of elements in U is $T = 100$. If $k = 5$ and the desired probability of false positive is $\tilde{p} = 0.01$, then the length of the filter must be $m = 985$ bits. If we decrease the probability of false positive to $\tilde{p} = 0.001$ and increase the number of hash functions to $k = 10$, the size of the filter becomes $m = 1{,}438$ bits.

B. Brief Description of CRBcast

CRBcast [1] is a reliable and energy efficient node-to-network multi-hop broadcasting protocol for multihop wireless networks. It consists of two phases. *Phase I* is a probabilistic broadcast and *phase II* is based on rateless coding. The motivation for using CRBcast is that it is shown to be 72% more efficient than flooding in terms of energy efficiency [1].

In *phase I* of the CRBcast, the message to be broadcast is divided into t packets $w_1, \ldots, w_t$. Using rateless coding, these packets are encoded into T packets $P_1, \ldots, P_T$. The source broadcasts the encoded packets to the network. The desirable feature of rateless coding is that the reception of any $t\gamma$ (γ is 5% for $t = 1000$) encoded packets suffices to decode for the original message. Since assumption is that the communication links between nodes is lossless, we set $T = t\gamma$. An arbitrary node, upon receiving the encoded packets, stores them and forwards each one with probability p to its neighbors (two nodes are called neighbors if they are within the communication range of each other). As a result of such probabilistic routing, a few nodes become complete (i.e., received enough packets to decode for the original data) at the end of *phase I*. The complete nodes decode the received packets to obtain the original data.

In *phase II*, incomplete nodes try to receive the necessary number of packets from their one-hop complete neighbors for successful decoding. Complete nodes broadcast advertisement messages ADV to their neighbors. An incomplete node receiving ADV, respond with a request message REQ that includes the number of requested packets and the ID of the complete node. A complete node, after receiving all the REQ messages from its neighbors, generates encoded packets anew using the original data and sends them to its neighborhood. The number of transmitted packets is equal to the maximum number of requested packets to supply to the incomplete nodes with necessary number of packets. After several iterations of the above process, all the incomplete nodes in the network recover the source data.

III. AuCRB Description

The proposed AuCRB has three phases to decrease the number of transmissions and the total latency. In *phase 0*, the original packets are first encoded using a rateless code [2], [13]. Authentication information is then generated by the source from the rateless encoded packets. This information is then broadcast to the network via flooding. Then a protocol similar to CRBcast follows. In *phase I*, the encoded packets are broadcast using a simple and scalable probabilistic relaying scheme referred to as PBcast. In PBcast, a node rebroadcasts packets that has received for the first time to its neighbors with some probability $p < 1$. In *phase II*, the nodes which received sufficient number of packets to decode for the original data during *phase I* help their neighbors which still need packets to retrieve for the original data.

To make the CRBcast robust to adversarial attacks and suitable for designing an efficient authentication mechanism, we make some modifications that are explained in the following. In AuCRB, using two instances of rateless coding with different parameters, the source node generates two sets of encoded packets. Sets of encoded packets generated by *RatelessI* and *RatelessII*

are used in *phase I* and *phase II*, respectively. In the first instance, the linear coefficients of *RatelessI* are randomly driven from an optimized distribution [13]. The linear coefficients employed in *RatelessII* are generated using a pseudorandom function based on an optimized distribution that is known to all nodes. We assume that all nodes have access to the same pseudorandom function and employ the same seed[1] to generate random coefficients. Hence, using *RatelessII*, all nodes generate the same set of coefficients.

Compared to the CRBcast, we make a slight modification to *phase II*. Let $Q_1, \ldots, Q_T$ be the encoded packets generated by *RatelessII*. A complete node partitions these packets into groups $G_1, \ldots, G_\ell$ of almost equal sizes. This operation is demonstrated in Figure 1. As we will explain in Section III-A, the partitioning technique enhances the generation of authentication information. Complete nodes, instead of sending the encode packets, send groups $G_1, \ldots, G_\ell$ to the incomplete nodes.

Fig. 1. Partitioning packets in *Phase II* of AuCRB.

In this work, we also take the latency into account and define the data availability based on the latency. Hence, for our simulations, we consider a Medium Access Control (MAC) in the CRBcast protocol as in [11] to compute the latency of the network and to obtain more realistic results.

In the following subsections, we explain the three phases of AuCRB in detail. All three phases are simply illustrated in Figure 2.

A. *Phase* 0

Phase 0 consists of two steps: (1) generating the report and encoding the data packets at the source, and (2) generating authentication information. In the following, we provide details of two steps.

Upon obtaining information critical to the entire network, a source node generates the t packets $w_1, \ldots, w_t$, and then it constructs the encoded packets $P_1, \ldots, P_T$ via *RatelessI*, where $T = t\gamma$ and $\gamma > 1$.

The source generates authentication information partially using a Bloom filter. The Bloom filter takes the encoded packets $P_1, \ldots, P_T$ as inputs and employs k independent hash functions $H_1, \ldots, H_k$. The output of the Bloom filter, an array M of bit length m, forms a piece of the authentication information. Bloom filter process for the packet P_1 is illustrated in Figure 3. Another piece of the authentication information belongs to *phase II*. In *phase II*, the encoded packets are generated from the original data known to the source. Moreover, all complete nodes generate the same set of

[1]The seed is updated after every broadcast session.

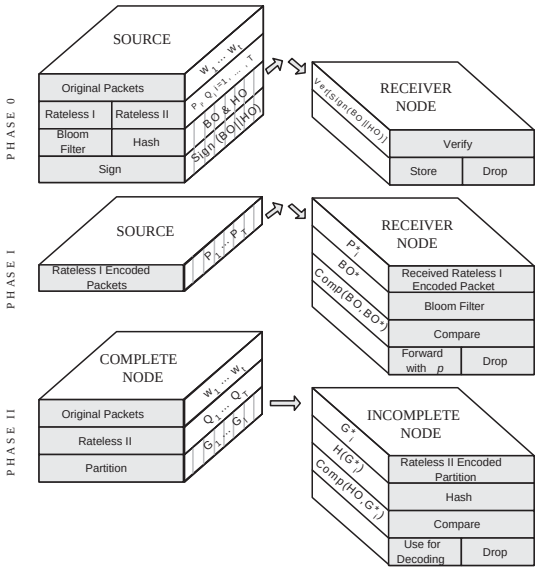

Fig. 2. Authenticated Collaborative Rateless Broadcast.

encoded packets using *RatelessII*. Therefore, the source generates authentication information for *phase II* as well. Let $Q_1, \ldots, Q_T$ be the encoded packets generated using *RatelessII*. These packets are partitioned into ℓ groups $G_1, \ldots, G_\ell$. Assuming that $j = T/\ell$ is an integer, the ℓ groups are related to the encoded packets as follows.

$$G_i = \left[Q_{1+(i-1)j}, \ldots, Q_{ij} \right], \qquad \forall i = 1, \ldots, \ell \quad (3)$$

Eventually, the source compiles the authentication information required for both phases as:

$$A = ID \| M \| H(G_1) \| \cdots \| H(G_\ell), \quad (4)$$

where $H(\cdot)$ is a cryptographically secure hash function, M is the output of the Bloom filter and ID is the ID of the source node. To prevent an adversary from modifying the authentication information, the source node signs A using an efficient signature scheme $Sign(\cdot)$ enhanced for use in resource constrained wireless networks [14]. Eventually, the source broadcasts the authentication information $(A, Sign(A), Ver)$ in multi-hop fashion to entire network. Here, *Ver* is the description of the signature verification algorithm. Other nodes in the network

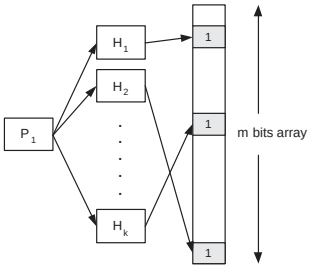

Fig. 3. Process of setting up the Bloom filter for the packet P_1.

broadcast this information to their neighbors. Every node receiving the authentication information, first verifies its integrity using $Sign(A)$. If it is verified, the node then broadcasts it with probability one to its neighbors.

For simplicity, we assume that every node has access to the algorithm for verifying the integrity of the authentication information. For this reason, it can be assumed that every node has the necessary information to execute *Ver* (every node had the public key of the other nodes in the network) or this information is provided by a trusted third party. However this approach is limiting the scalability of the network. Hence, we suggest using ID-based signature schemes [15]. In such schemes, the verification algorithm is obtained from the ID of the source node generating the signature.

B. Phase I

In this phase, *RatelessI* encoded packets generated for *phase I* during *phase 0* are broadcast to the network. When a node receives a packet encoded with *RatelessI*, it initially verifies the authenticity of the packet using the Bloom filter. Every packet authenticated by a node is forwarded with probability p to its neighbors. If a packet cannot pass the authentication test, it is dropped and the forwarding node's credential is decreased by its neighbors. In other words, a legitimate node keeps a list of its malicious neighbors based on their malicious behaviors. Hence, a legitimate node can take action against its malicious neighbors to increase the data availability of the network. This will be explained in Section IV.

C. Phase II

In *phase II*, complete nodes advertise their completeness to their neighbors by broadcasting ADV messages. Incomplete nodes respond by sending a request message REQ that includes the number of required packets. Complete nodes send packet groups $G_1, \ldots, G_\ell$ instead of the encoded packets $Q_1, \ldots, Q_T$. Since an incomplete node receives its requested packets from a specific complete node, it is not required to verify the authenticity of packets individually. Hence, an incomplete node authenticated the packets received from a complete neighbor as a group. If the authentication fails, all packets are discarded and the malicious node is detected.

Let c be the maximum number of packets requested from a complete node. This node broadcasts $G_1, \ldots, G_s$, where $s = \lceil (c/j) \rceil$. Using the authentication information A, incomplete nodes verify the authenticity of the blocks instead of individual packets. Block authentication failure implies that the complete node who is sending them is malicious. In this case, the incomplete node must wait until an ADV message from a legitimate neighboring complete node is arrived. It also adds the detected malicious node to its list.

IV. Threat Analysis

In this section, we analyze the security of our scheme in terms of data authenticity and node impersonation. It is worth noting that, in this work we are not considering the jamming attack, which is a threat for all existing broadcasting schemes. We assert that the availability is ensured by AuCRB in the presence of any number of insider attackers as long as the legitimate nodes form a connected graph with the source. This property is due to the *phase II* of the AuCRB. In other words, the complete legitimate nodes will help incomplete legitimate nodes to recover and decode via the execution of *phase II*.

In our scheme, every node can potentially be a broadcasting source. Providing the broadcast ability to every node allows an adversary to send its own data to the network by just compromising a single node. This is a common drawback of all such schemes that allow every node in the network to broadcast. A solution is requiring a message to be generated by the collaboration of several local nodes using a secret sharing scheme [16]. This not within the scope of our paper.

Assuming the legitimacy of the source node, the adversary cannot modify the report at its generation time. Moreover, adversary cannot deceive receivers by modifying the message since authentication information is provided and digitally signed by the source node. We note that, a receiver node does not accept any data packets before receiving the legitimate authentication information from the source. A bogus packet injected during *phase I* is filtered out with a high probability after one hop travel. The filtering strength of the *phase I* depends on the false positive probability of the Bloom filter. The network designer can arbitrarily decrease this probability to the expense of increasing communication overhead as explained in Section II-A. Similarly, in *phase II*, an incomplete node uses the signed authentication information to authenticate the group of packets. Hence, it is not possible for an adversary to inject malicious packets without being detected. Therefore, when an adversary either attempts to inject bogus packets to the network or drop legitimate packets in either of the two phases of the protocol, its major impact is increased latency. As long as the legitimate nodes form a connected graph, they finally receive sufficient number of packets to retrieve the original message via *phase II*. Thus the availability remains intact, however, with higher latency. This increase in latency due to adversary can be attributed to two factors. One is that the number of legitimate nodes who supply packets to the network is reduced (hence, the packets may travel longer hops). The other is increased waiting time to access the channel by the legitimate nodes. In other words, malicious nodes may compete via MAC to access the channel to inject packets or remain silent. There are several solutions to prevent the attack at MAC layer as in [17], which we do not discuss here.

Alternatively, to combat attacks at the MAC layer, we may use the fact that a malicious node sending bogus packets can be detected by its legitimate neighbors who keep a list of malicious nodes that they detected. Hence, when a legitimate node receives a bogus packet from one

of its neighbors, the receiver will no longer accept any packets from that specific node. In other words, when it hears an RTS (request to send) from the detected node, it will not respond with a CTS (clear to send) and as a result, another legitimate node can get the channel instead. We note that in our scheme every node is able to detect malicious nodes individually without using expensive and vulnerable voting systems. The only problem here is that, a malicious node may send bogus packets by impersonating another legitimate node. As a result of this attack, the impersonated legitimate node will be marked as malicious by its neighbors. To avoid this situation, the pairwise keys between nodes are used. Due to the initial key pre-distribution, all nodes in the network know their one-hop neighbors and hence, a node does not communicate with a node that is not its one-hop neighbor. Moreover, as a result of the Medium Access Control, one and two-hop neighbors of a node (that has access to the channel) know which node is occupying the channel. Thus, if a malicious node impersonates one of its one or two-hop neighbor nodes and sends RTS to the neighbors of the impersonated node, the impersonated node will learn about this due to either RTS (if it is a one-hop neighbor of the malicious node) or CTS (if it is a two-hop neighbor of the malicious node). As soon as a legitimate node realizes that it is being impersonated by a malicious node, it sends warning messages to its neighbors encrypted by its pairwise keys. Hence, the impersonation attack is easily detected with a cost of a few more transmissions.

This is illustrated in Figure 4. Here, node E is the legitimate node with neighbors $1, 2, 3, 4$ and 5. When the malicious node $\hat{E}$ impersonates E and sends RTS, E will also learn that a node occupying the channel using its ID. Hence E will send encrypted warning messages to its neighbors. As nodes 3 and 5 get these warnings, they remove E from their *detected adversary* list. We note that, since nodes 6 and 7 are not in one-hop neighborhood on E, they are not affected from this attack (they do not communicate with a node that is not their one-hop neighbor).

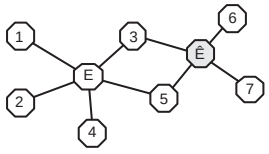

Fig. 4. Node impersonation attack.

Moreover, aside from preventing the detected malicious node from getting the channel, upon detection of a malicious neighbor in *phase I*, a legitimate node can automatically increase its forwarding probability, p, to compensate for the increased latency. However, we have not explored this in our analysis and simulations.

V. Performance Analysis and Comparison to Related Work

We particularly consider wireless sensor networks as an example of resource constrained multihop static networks. We compare our scheme with a recently proposed broadcast authentication scheme in [9]. The rational for choosing Secure Deluge in [9] for comparison is that all previous schemes on broadcast authentication either have vulnerabilities to certain attacks or they are impractical for implementation as we discussed in Section I-A. Even though Secure Deluge is also vulnerable to flooding attack, its implementation is easy and it does not require too much extra overhead. Besides, all broadcasting schemes, who guarantee availability, use the flooding technique to transmit the authentication information and the data packets. Hence, they have more or less similar overhead as in [9]. Throughout this section, we assume that the fraction of compromised nodes in the entire network is C and the network is connected unless otherwise stated.

A. Data Availability

As opposed to previous works, we define the data availability based on the latency. In other words, for 100% availability, all nodes in the network need to become complete in a definite time.

Using computer simulations, we have studied data availability in ensemble of 50 networks. In our simulations, we have assumed $N = 1000$ nodes, $T = 1000$ packets, the transmission range $r = 0.2$ units, size of the deployment field is 2×2[2] unit square, and the average degree of a node is 30 for connectivity. We consider both the energy consumption of the network (due to the number of packet transmissions only) and the latency versus the forwarding probability in *phase I*. Hence we created the energy consumption-latency metric as

$$metric = \frac{N_{tx}(i)}{min(N_{tx})} \times \frac{latency(i)}{min(latency)} \qquad (5)$$

where $N_{tx}(i)$ is the number of transmissions per node for $p = i$, $min(N_{tx})$ is the minimum number of transmissions per node among all p values, $latency(i)$ is the total time required for all nodes to become complete for $p = i$, and $min(latency)$ is the minimum latency value obtained from all p values. We assume that the transmission time for one packet is equivalent to one time-unit. Hence we obtained an optimal value for p as in Figure 5. In the figure, energy stands for $N_{tx}(i)/min(N_{tx})$ and latency stands for $latency(i)/min(latency)$. As expected, energy consumption increases and latency decreases with increasing p. Moreover, we note that, the decrease in latency is very low for $p > 0.3$. We simulated the same network for [9] and obtained the above metric as 6.0679. This is more than 4 times larger than our scheme with optimum p and more than 3 times larger than our worst case p. Hence, our scheme is more efficient than [9] in

[2]Length measurements are normalized to the unit of measurement.

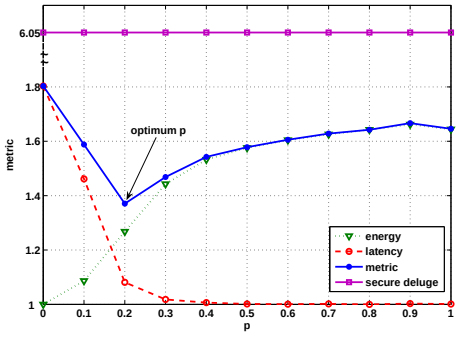

Fig. 5. Energy consumption-Latency metric versus p (without any adversary).

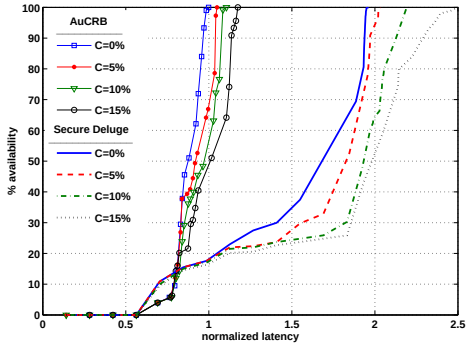

Fig. 6. Data availability versus latency for different number of malicious nodes.

terms of the energy-latency metric. Hence, we conclude that, confident that, our scheme outperforms all previous schemes that use flooding technique as in [9].

We also evaluated and compared the performance of both our scheme and [9] in adversarial environments where malicious nodes either drop or modify legitimate packets. In AuCRB, since the malicious nodes that modify the legitimate packets or inject bogus packets are detected, we note that the adversary gives most serious damage when it modifies or injects bogus packets during *phase II*. Because, when the adversary gets the channel in *phase I*, it can only send a single packet. If it sends a malicious packet, it will be immediately detected by its neighbors. Hence its legitimate neighbors can take actions against that malicious node as explained in Section IV. Hence, in our simulations, adversarial nodes only drop the packets during *phase I* and modify packets or inject bogus ones during *phase II*. When there are no malicious nodes in the network, we observed that all nodes become complete in about 7100 time units. Thus, we take this time as a reference and obtain the availability at 7100 time units for different number of malicious nodes in Figure 6. In this figure, we normalized all latency values by 7100 and compared AuCRB and Secure Deluge in terms of total latency with different number of adversarial nodes. Data availability at time 7100 decreases as the number of compromised nodes increases. The numerical results obtained for our scheme and [9] are summarized in Table I and Table II, respectively. The probability of availability at time 7100 and time needed to attain 100% availability are given for different number of malicious nodes. We conclude that the scheme in [9] requires twice as much time as our scheme to provide 100% availability. We should note that, we do not see the benefit of detecting malicious nodes in our latency simulations. As we mentioned above, in our simulations we are considering the the situation that the adversary gives the most serious damage (modifies packets or inject bogus ones during *phase II* only). Hence, even though the legitimate nodes detect their malicious neighbors during *phase II*, they are not using this information to enhance the availability in our simulations. The reason is that, we are simulating a single broadcast session such that adversary only plays an active role in the last phase (hence, the malicious nodes can only be detected in the last phase). The benefit of detecting malicious nodes on availability will become obvious when the legitimate nodes use their *detected adversary* list in the next broadcast sessions. Indeed, when we repeated the above simulation with the legitimate nodes have the malicious nodes (that are detected during the above simulation) in their *detected adversary* list since the beginning of the session, we observed that our scheme provides almost 100% availability at time 7100 for different number of malicious nodes used in the previous simulation.

TABLE I
AVAILABILITY VERSUS LATENCY FOR AUCRB.

C	% Availability @ 7100	100% availability @
0%	100	7100
5%	67	7450
10%	63	7850
15%	47	8340

TABLE II
AVAILABILITY VERSUS LATENCY FOR SECURE DELUGE.

C	% Availability @ 7100	100% availability @
0%	18	13900
5%	17	14430
10%	16	15560
15%	14	17710

B. Overhead Analysis

In this section, we study the computation and communication overheads of the AuCRB scheme and compare with [9]. Our study reveals that the proposed scheme when compared with all other schemes which use the

flooding technique is superior in terms of communication overhead and has a negligible disadvantage in computational overhead due to decoding. Thus, all schemes designed based on flooding are much more expensive than AuCRB in terms of the overall communication and computation overhead.

1) Computation Overhead: We analyzed the computation overhead of AuCRB for each phase separately.

Phase 0: The computation overhead introduced by the source at *phase 0* is mainly due to signature generation, Bloom filter construction, generation of authentication information for *phase II* and rateless encoding. We note that the computation cost due to rateless encoding and generation of the authentication information for *phase II* is negligible when compared to the other two. Since the Bloom filter takes T input packets and calculates k hash values for every packet, its computation cost is kT hash operations. Thus, assuming that time complexities of a single hash and a single signature calculation are τ_h and τ_s, respectively, the total computational cost at the source is

$$\mathcal{C}_s^0 = (kT)\tau_h + \tau_s. \tag{6}$$

At *phase 0* each receiver node verifies the signed authentication information formed by the source. Hence if the time complexity for the signature verification is τ_v, the computation cost for the receiver nodes is

$$\mathcal{C}_r^0 = \tau_v. \tag{7}$$

Phase I: At *phase I*, the only computational complexity is due to the Bloom filter verification. If we denote the average number of different packets received by a node at *phase I* as P, then the average computational cost at each node is

$$\mathcal{C}_r^1 = (kP)\tau_h. \tag{8}$$

Phase II: At this phase the main computations are the verification of the partitions, *RatelessII* encoding at the complete nodes and rateless decoding at the nodes that receive sufficient number of packets. However, we note that the cost due to the rateless decoding is dominant. We denote the time complexity to decode T packets as τ_d. Then, the average computational cost per node at phase II is

$$\mathcal{C}_r^2 = \tau_d. \tag{9}$$

Thus, the total average computational complexity per source,$\mathcal{C}_s$, and a receiver node,$\mathcal{C}_r$, for the three phases are obtained as

$$\mathcal{C}_s = (kT)\tau_h + \tau_s \tag{10}$$

$$\mathcal{C}_r = \tau_v + (kP)\tau_h + \tau_d \tag{11}$$

Here, we measure the computation overhead as the amount of energy consumed by those computations.

We numerically obtained the computational complexity and conducted our calculations for MICA2DOT with 4 MHz 8-bit processor using Chipcon CC100 antenna at both optimum forwarding probability $p = 0.2$ and $p = 1$ (the worst case in AuCRB in terms of energy consumption). We used 160-bit elliptic curve signature (ECC-160) and SHA1 for the hash. From [18], ECC-160 signature generation and verification consume 22.82mJ and 45.09mJ energy, respectively. From [19], we calculated the energy consumption for the hash operations. Moreover, using [20] and [21], we obtained energy consumed to decode T packets by counting the number of XOR operations, memory reads and memory writes.

As a result, we obtained the total average energy consumption (due to computations) for the source node as 1.4J. The average energy consumption (due to computation at each phase) is illustrated in Table III for every receiver node. The comparison of our scheme with the Secure Deluge in terms of computation overhead is shown in Figure 7. Since AuCRB uses rateless codes and provides immediate packet authentication, the computational overhead of our scheme is higher than the Secure Deluge scheme. However, as we will show in Section V-B2, this disadvantage in computation overhead has a negligible effect on the overall energy consumption when both the computation and communication overheads are considered.

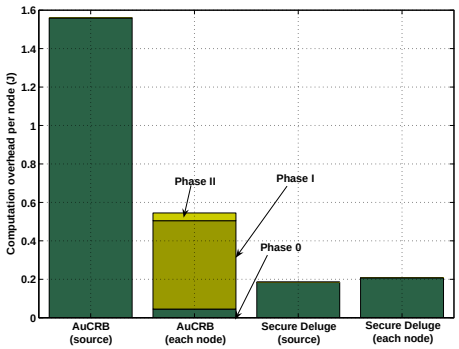

Fig. 7. Comparison of energy consumption due to computation overhead.

TABLE III
COMMUNICATION AND COMPUTATION OVERHEADS PER RECEIVER NODE IN AUCRB.

Overhead	*phase 0*	*phase I*	*phase II*	Total
Computation	45.9mJ	0.46J	40.4mJ	0.551J
Communication	120.160mJ	4.003J	950.4mJ	5.44J

2) Communication Overhead: Here, we study the energy consumption due to communication overhead of our scheme and compare it with the Secure Deluge in [9]. To transmit and receive one byte data using Chipcon CC100 antenna, 59.2μJ and 28.6μJ is consumed respectively. To be consistent with [9], we used packets with 64 byte payload in our simulations. We calculated the average number of packets sent and received by a node for both optimum forwarding probability $p = 0.2$ and $p = 1$ (the worst case scenario) and obtained the average energy consumption per node for different phases of

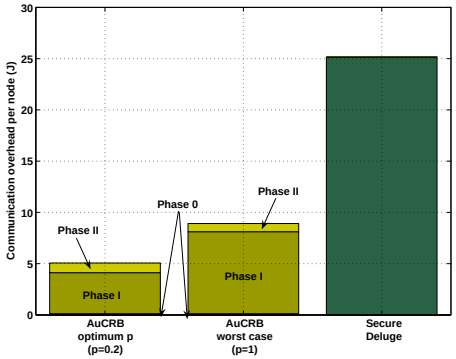

Fig. 8. Comparison of energy consumption due to communication overhead.

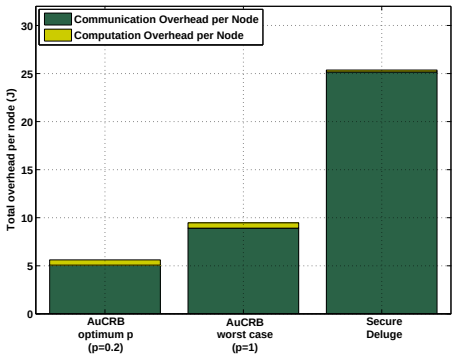

Fig. 9. Comparison of energy consumption due to total overhead..

the protocol. The results are summarized in Table III. Moreover, we compared the communication overhead of AuCRB for both $p = 0.2$ and $p = 1$ with the Secure Deluge scheme in Figure 8. Finally, considering both the computation and communication overheads, we calculated the total energy consumption per sensor node for AuCRB for $p = 0.2$, $p = 1$ as well as for [9]. The results are shown in Figure 9. Hence, we conclude that even in the worst case scenario (when $p = 1$) our scheme outperforms the Secure Deluge. We can make the similar conclusion against all the other schemes which use flooding to ensure availability in broadcasting.

VI. Conclusions

This paper was concerned with availability, reliability and authentication for broadcasting in ad hoc wireless networks, where adversary may compromise nodes, then drops or modifies packets, injects bogus packets or mounts routing attacks. We proposed a node-to-network multi-hop broadcasting scheme by simultaneously considering the above requirements. We showed superiority of our scheme to meet the requirements with reduced transmission and latency. Furthermore, our proposed scheme, due to its built in coding mechanism, can be employed when the packets are lost due to reasons other than the Byzantine attacks. Finally, our scheme is suitable to broadcast large number of packets to cope with the rateless coding overhead.

References

[1] N. Rahnavard and F. Fekri, "CRBcast: A collaborative rateless scheme for reliable and energy-efficient broadcasting in wireless sensor networks," in *Proc. Int. Symp. Inform. Process. Sensor Networks - IPSN'06*. TN: ACM Press, Apr. 2006, pp. 276–283.

[2] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[3] A. Perrig, R. Canettiz, J. D. Tygary, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *00*, May 2000, pp. 56–73.

[4] A. Perrig *et al.*, "SPINS: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, Nov. 2002.

[5] D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," in *Proc. Ann. Netw. Distribut. Syst. Secur. Symp. - ANDSSS'03*. Internet Soc., Feb. 2003.

[6] J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast stream authentication using erasure codes," vol. 6, no. 2, pp. 258–285, May 2003.

[7] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos, "Multicast authentication in fully adversarial networks," in *04*. CA: IEEE Comput. Soc., May 2004, pp. 241–255.

[8] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Proc. Australasian Conf. - ACISP'02*, ser. Lecture Notes in Computer Science, L. Batten and J. Seberry, Eds., vol. 2384. Berlin: Springer-Verlag, July 2002, pp. 144–153.

[9] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the deluge network programming system," in *Proc. Int. Symp. Inform. Process. Sensor Networks - IPSN'06*. TN: ACM Press, Apr. 2006, pp. 326–333.

[10] F. Delgosha and F. Fekri, "Threshold key-establishment in distributed sensor networks using a multivariate scheme," in *Proc. IEEE Conf. Comput. Commun. - INFOCOM'06*. Barcelona: IEEE, 2006, CD-ROM.

[11] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. - INFOCOM'02*, 2002.

[12] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *ACM Commun.*, vol. 13, no. 7, pp. 422–426, July 1970.

[13] M. Luby, "LT codes," in *Proc. IEEE Symp. Found. Comput. Science*. Vancouver: IEEE Comput. Soc., Nov. 2002, pp. 271–280.

[14] G. Gaubatz, E. O. Jens-Peter Kaps, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *05*. HI: IEEE Comput. Soc., Mar. 2005, pp. 146–150.

[15] W. Lee and W. Sriborrirux, "Optimizing authentication mechanisms using ID-based cryptography in ad hoc wireless mobile networks," in *Int. Conf. Inform. Netw. - ICOIN'04*, ser. Lecture Notes in Computer Science, H.-K. Kahng and S. Goto, Eds., vol. 3090. Berlin: Springer-Verlag, Feb. 2004, pp. 925–934.

[16] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbood of Applied Cryptography*. NY: CRC Press, 1997.

[17] A. Cardenas, S. Radosavac, and J. Baras, "Performance comparison of detection schemes for mac layer misbehavior," in *Proc. IEEE Conf. Comput. Commun. - INFOCOM'07*. Anchorage: IEEE, 2007, CD-ROM.

[18] A. S. Wander *et al.*, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. - PerCom'05*. CA: IEEE, Mar. 2005, pp. 324–328.

[19] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha, "Analyzing the energy consumption of security protocols," in *Proc. International Symposium on Low Power Electronics and Design*, 2003.

[20] "Report of MBMS FEC status in sa4," Technical Specification Group Services and System Aspects, Tech. Rep. $TSGS28(05)0246$, 2005.

[21] S. Park, J. W. Kim, K.-Y. Shin, and D. Kim, "A Nano operating system for wireless sensor networks," in *Proc. Advanced Communication Technology - ICACT'06*, 2006.