Secure Network-Coded Wireless Multicast for Delay-Sensitive Data

Tuan T. Tran[1], Hongxiang Li[1], Lingjia Liu[2], and Samee U. Khan[3]
[1]Department of Electrical and Computer Engineering,
J.B. Speed School of Engineering, University of Louisville, KY 40292.
[2]Department of Electrical Engineering and Computer Science,
The University of Kansas, Lawrence, KS 66045.
[3]Department of Electrical and Computer Engineering
North Dakota State University, Fargo, ND 58102.
Email: tttran14@louisville.edu; h.li@louisville.edu; lingjialiu@ittc.ku.edu; samee.khan@ndsu.edu

***Abstract*—Wireless multicast for delay-sensitive data is challenging because different receivers may experience different packet losses. Network coding offers significant advantages over the traditional Automatic Repeat-reQuest (ARQ) protocols in that it mitigates the need for retransmission and has the potential to approach the min-cut capacity. Network-coded multicast would be, however, vulnerable to false packet injection attacks, in which the adversary injects bogus packets to prevent receivers from correctly decoding the original data. Without a right defense in place, even a single bogus packet can completely change the decoding outcome. Existing solutions either incur high computation cost or cannot withstand high packet loss. In this paper, we propose a novel scheme to defend against false packet injection attacks on network-coded multicast for delay-sensitive data. Specifically, we propose an efficient authentication mechanism based on null space properties of coded packets, aiming to enable receivers to detect any bogus packets with high probability. We further design an adaptive scheduling algorithm based on Markov Decision Processes (MDP) to maximize the number of authenticated packets that can be received within a given time constraint. Both analytical and simulation results have been provided to demonstrate the efficacy and efficiency of our proposed scheme.**

I. Introduction

Multicasting delay sensitive data in wireless networks is of great interest in many applications. For example, in a cellular or WiFi network, multiple users may play a network-based game or subscribe to the realtime stock market information updated through the same base station or access point. In these applications, a common requirement is that the information must be delivered to as many receivers as possible within a certain time constraint.

In general, multicasting delay-sensitive data in a lossy environment is challenging. In particular, wireless channels are lossy in nature where packets may be lost during transmissions due to many reasons, such as RF interference and channel fading. At any given time point, different receivers may experience different packet losses. Traditional ARQ protocols are generally not suitable for multicasting in lossy networks which would incur excessive retransmissions and high latency, as shown in [9] for TCP in lossy networks.

Network coding is a promising communication paradigm, and it has been proved that random network coding can approach the min-cut capacity [7]. Simply put, under random network coding, the transmitter encodes a batch of original packets and transmits their random linear combinations to all the receivers. Any receiver can recover the original data packets as long as it has received a sufficient number of linear independent coded packets. By doing so, there is no need for retransmission and the number of receivers that successfully recover the original packets can be maximized.

Networking coding is, however, vulnerable to false packet injection attacks in a hostile environment. In particular, the adversary may inject bogus coded packets to prevent receivers from decoding the original data. Without any defense in place, this attack can be detrimental. In particular, because a set of coded packets are used to decode the original packets, e.g., using Gaussian elimination, even a single bogus coded packet can completely change the decoding outcome, rendering other genuine coded packets meaningless. A naive approach by letting the transmitter digitally sign every packet would not resolve this issue since it would incur significant computation overhead at the receiver side, due to expensive signature verification. In addition, the adversary may also send many bogus packets or replay intercepted packets to force receivers into wasteful packet processing so as to quickly deplete their limited energy and/or memory buffer.

To the best of our knowledge, little attention has been paid to the above problem. Notably, several schemes have been proposed to enable efficient stream authentication by amortizing one signature over multiple packets [5], [11], [13], [14], which, however, cannot withstand high packet losses. In addition, packet authentication in network coded system has recently been investigated in the context of pollution attack. Most of the existing solutions, e.g., [6], [2], [3], are based on expensive homomorphic hash computation, which are vulnerable to denial-of-service (DoS) attacks.

In this paper, we study secure wireless multicast and devise a novel scheme that enables efficient packet authentication at the receiver side, while at the same time maximizing the number of authenticated packets at each receiver within the given time constraints. Specifically, our proposed scheme has two main components. Inspired by null space defense originally proposed to defend against pollution attack in network coding, the first component lets the transmitter generate a set of null key packets to be transmitted besides coded packets. On receiving the null key packet, each receiver can detect any bogus packet in the buffer with high probability. We further design an efficient authentication scheme to significantly reduce the computation cost incurred by null key packet authentication. The second component aims at maximizing the authentication rate, i.e, the number of recovered data packets per unit time, at each receiver. Specifically, we design a novel efficient scheduling algorithm to adaptively transmit the null key packets based on the framework of Markov Decision Processes.

Our main contributions in this paper can be summarized as follows:

- We propose a novel scheme to enable efficiently authentication of transmitted data at the receiver side.
- We further propose an effective scheduling algorithm to maximize the authentication rate across all the receivers.
- The effectiveness of the proposed scheme is corroborated through both theoretical analysis and simulations.

The organization of the paper is as follows. We first provide in Section II the system model, attack model, and security objectives. We then present our proposed authentication scheme

in Section III. In Section IV, we formulate and solve the authentication problem by using the framework of Markov Decision Processes. Performance evaluation is presented in Section V. Finally, we conclude the paper in Section VI.

II. System, Attack Model, and Security Objectives

A. System Model

We consider a single-hop wireless network, where a transmitter wishes to securely multicast delay-sensitive data to a group of users. We assume that the system is time-slotted and each slot length corresponds to one packet transmission. Furthermore, we assume that a block of $m-1$ data packets arrives at the transmitter periodically, and each block is associated with a deadline of T time slots, $T \geq m$. That is, to be useful, a packet needs to be received and authenticated by the deadline. Additionally, we assume that the transmitter employs random network coding to generate coded packets. We assume that the links between the transmitter and receivers are lossy, i.e., each packet transmitted to receiver R_j is subject to an erasure with probability p_j. The erasure probabilities are independent across the receivers and independent and identically distributed $(i.i.d)$ across time slots. Furthermore, at each time slot any user that wants to transmit data contends the transmission channel by using the distributed coordination function (DCF), a standard MAC protocol in IEEE 802.11-based wireless networks [1].

B. Attack Model

We consider a computationally bounded adversary consisting of both *external* and *internal* attackers. External attackers do not belong to the target network, but they are capable of overhearing packet transmissions, injecting bogus packets, and replaying intercepted packets. Internal attackers are compromised yet undetected nodes which are fully controlled by the adversary. Following the conventional assumption, we further assume that the transmitter cannot be compromised and that non-compromised receivers are always the majority.

Among the many attacks the adversary can launch, this paper focuses on tackling the following two types of attacks on network coding based multicast.

- The adversary may inject bogus coded packets to pollute the receiver's buffer to prevent receiver from successfully decoding the original packets.
- The adversary may send many bogus packets or replay intercepted packets to force receivers into wasteful packet processing so as to quickly deplete their limited energy and/or memory buffer.

In addition, we assume that the transmitter has a public and private key pair K/K^{-1} and the public key K is publicly known.

C. Security Objectives

In view of the two aforementioned attacks, our objectives are two folds:

- Packet integrity: Any bogus packet should be detected with high probability.
- DoS attack resilience: Packet authentication should be efficient.

III. Packet Authentication

In this section, we present our proposed scheme that enables efficient packet authentication for multicasting delay-sensitive data.

First of all, our scheme is inspired by the null space defense originally proposed for defending against pollution attack. In particular, to enable efficient coded packet authentication, we let the transmitter generate a set of null key packets to be transmitted besides the coded packets. On receiving the null key packet, each receiver can authenticate all the coded packets

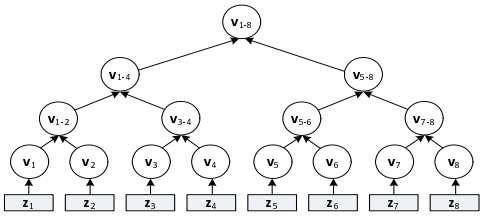


Fig. 1. An example of construction Merkle hash tree over $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, where $n = 8$.

it has received so far and detect bogus packets with high probability, if any. In addition, we also propose an efficient scheme whereby the receivers to authenticate null packets. In what follows, we detail the design of our scheme, including *data packet preprocessing*, *null key packet generation*, and *packet authentication at the receiver*.

A. Data Packet Preprocessing

As is standard, we assume that network coding is applied to a batch of $m-1$ incoming packets, denoted by $\{\mathcal{N}_i\}_{i=1}^{m-1}$. For the sake of security, each batch needs a signature packet, denoted as

$$\mathcal{S} = K^{-1}(h(\mathcal{N}_1 || \cdots || \mathcal{N}_{m-1})) \ , \tag{1}$$

where $K^{-1}(\cdot)$ denotes the transmitter's signature using its private key, $h(\cdot)$ is a good hash function, and $||$ denotes concatenation. For convenience, we subsequently treat the signature packet as a normal packet, i.e., $\mathcal{S} = \mathcal{N}_m$.

The transmitter interprets each packet $\mathcal{N}_i$ as an n-dimensional vector $(N_{i,1}, \cdots, N_{i,n})$ over a finite field $\mathbb{F}_\rho$ and appends a unit vector of length m to the vector $\mathcal{N}_i$ to create m augmented vector $\hat{\mathcal{N}}_i$ as

$$\hat{\mathcal{N}}_i = \langle \mathcal{N}_i, \underbrace{0, \cdots, 0, \overbrace{1}^{i}, 0, \cdots, 0}_{m} \rangle \ .$$

The jth coded packet is generated from the m original packets, in the form of

$$\begin{aligned} \mathcal{C}_j &= \sum_{i=1}^{m} \alpha_{j,i} \hat{\mathcal{N}}_i \\ &= \langle C_{j,1}, \cdots, C_{j,n}, \alpha_{j,1}, \cdots, \alpha_{j,m} \rangle \ , \end{aligned} \tag{2}$$

where $C_{j,i} = \sum_{i=1}^{m} \alpha_{j,i} N_{i,j}$.

B. Null Key Packet Generation

The adversary may inject bogus data packets to prevent receivers from successfully decoding the original packets. Recall that the transmitter signs the original packets, which can be verified after successful decoding. One challenge here is that if the receiver cannot differentiate genuine coded packets from bogus ones, it may be difficult to identify the sets of packets to correctly decode the original packets. For example, assume that the receiver has received m genuine packets and b bogus ones. To find the correct m packets for decoding, the receiver needs to examine possibly $\binom{m+b}{m}$ combinations, a number that grows exponentially in b.

We adopt the technique proposed in [8] to generate some null key packets whereby the receiver can authenticate coded data packet. Specifically, let $\mathbf{N}$ denote the $m \times (m+n)$ matrix whose ith row is $\hat{\mathcal{N}}_i$. These m augment vectors form a set of m independent vectors that span a subspace $span(\mathbf{N})$. We can see that any coded packet $\mathcal{C}_j$ belongs to $span(\mathbf{N})$.

The null space of the matrix $\mathbf{N}$, denoted as $null(\mathbf{N})$, is the set of all vectors $\mathbf{z}$ for which $\mathbf{N}\mathbf{z} = 0$. According to the rank-nullity theorem, we have

$$rank(\mathbf{N}) + nullity(\mathbf{N}) = m + n \ ,$$

where the nullity of the matrix $\mathbf{N}$ is the dimension of the null space of $\mathbf{N}$. Because $\{\hat{\mathcal{N}}_i\}_{i=1}^m$ are independent, we have $rank(\mathbf{N}) = m$ and $nullity(\mathbf{N}) = n$.

Assume that $null(\mathbf{N})$ is spanned by the basis vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, which can be computed using Gaussian elimination. We can see that $\{\mathbf{z}_i\}_{i=1}^n$ can be used to authenticate the coded packet, because $\mathcal{C}_j\mathbf{z}_i = 0$, for all j, i.

The adversary may also inject bogus null key packets. Without a proper authentication mechanism in place, genuine data packet may be misidentified as bogus ones. Moreover, null key packet authentication must be efficient, which may otherwise be exploited by the adversary to inject a large number of bogus ones to force receiver to perform expensive signature verification.

We leverage a combination of Merkle hash tree [10] and digital signature to realize efficient authentication for null key packets. Specifically, let $n = 2^d$ for some integer d. To authenticate $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, the transmitter builds a Merkle hash tree of depth d on top of $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, as illustrated in Fig. 1. In particular, the transmitter computes $\mathbf{v}_i = h(\mathbf{z}_i)$, for all $i \in [1, n]$, and builds a binary tree by computing each internal node as the hash of its adjacent children nodes. For example, $\mathbf{v}_{3-4} = h(\mathbf{v}_3||\mathbf{v}_4)$ and $\mathbf{v}_{1-4} = h(\mathbf{v}_{1-2}||\mathbf{v}_{3-4})$ in Fig. 1.

The transmitter signs the root of the Merkle hash tree, e.g., $\mathbf{v}_{1-8}$ using its private key K^{-1}. Given the Merkle hash tree, the transmitter constructs one null key packet for each $\mathbf{z}_i$, which consists of $\mathbf{z}_i$ itself and its authentication information, i.e., all the siblings of the nodes in the path from $\mathbf{v}_i$ to the root of the Merkle hash tree. For example, we have

$$\mathcal{K}_2 := \langle \mathbf{z}_2, \mathbf{v}_1, \mathbf{v}_{3-4}, \mathbf{v}_{5-8}, \mathbf{v}_{1-8}, K^{-1}(h(\mathbf{v}_{1-8}))\rangle$$

C. Packet Authentication at the Receiver

We defer the scheduling at the transmitter and introduce the following operation at the receiver in this subsection.

Upon receiving a data packet $\mathcal{C}_j$, the receiver marks it as an unverified packet and store in its buffer. When a new null key packet arrived, the receiver takes the following steps:

1) Step 1: check if a valid root of the Merkle hash tree of the current batch, e.g., $\mathbf{v}_{1-8}$, has been received before. If not, verify the signature $K^{-1}(h(\mathbf{v}_{1-8}))$ using the transmitter's public key K. Otherwise, check if it matches the one in new null key packet. The null packet is dropped if either $\mathbf{v}_{1-8}$ does not match or the signature is invalid .
2) Step 2: verify the authenticity of the received null key $\mathbf{z}_i$ using the authentication information along the path to the root of the Merkle hash tree. For example, for null key packet $\mathcal{K}_1$, the receiver checks if
$$\mathbf{v}_{1-8} = h(h(h(h(\mathbf{z}_1)||\mathbf{v}_2)||\mathbf{v}_{3-4})||\mathbf{v}_{5-8}) \ .$$
3) Step 3: using the received null key to check all the received data packets to detect bogus packets, if any.
4) Step 4: repeat Step 1 to Step 3 until it reached the deadline. If the number of coded packets passed the authentication is larger or equal to m,[1] decode the original data packets $\{\mathcal{N}_i\}_{i=1}^m$. If the number of authenticated coded packets is less than m, say m', then randomly choose a $m - m'$ packets from unverified packets to do the decoding.
5) Step 5: verify if $\mathcal{N}_m$ is the valid signature of $\mathcal{N}_1||\cdots||\mathcal{N}_{m-1}$ (cf. Eq. (1)). If so, the batch is considered authenticated, and dropped otherwise.

Note that, the recovery of the transmitted data succeeded only if sufficient coded packets have been received and the receiver chooses a correct set of the coded packets to decode. pass the authentication of a newly received null key.

[1]Assume a large finite field is used so that all coded packets are independent.

IV. MDP-based Scheduling

Consider a random-access based system with a deadline constraint on each data packet. We will design a transmission policy to maximize the authentication rate at the receivers. In what follows, we show that given a batch size m and its associated deadline T, without carefully scheduling transmissions it may result in poor performance. Moreover, we show that the time instances, at which the null keys are released, play an important role in determining the expected authentication rate. Finding an optimization scheduling boils down to how the transmitter chooses which packet to transmit at each time slot. We formulate and solve the optimization problem by using the tools of the Markov Decision Proccess. We start by presenting an example to motivate the adaptive scheduling in our proposed scheme.

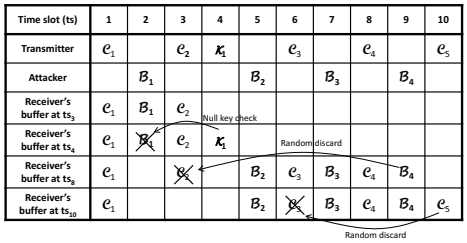
A. Time-delay Null Key Release

In order to filter out bogus packets, we use the null keys as discussed above. We should caution that there is a high chance that the attackers can receive all the transmitted null keys; therefore, they can generate bogus packets that pass all the null key checks. To tackle this issue, we propose a time-delay null key release mechanism to limit this effect, with the basic idea being to release null keys at different time instances after some transmissions of the data packets. To get a more concrete sense, we present the following example for illustration.

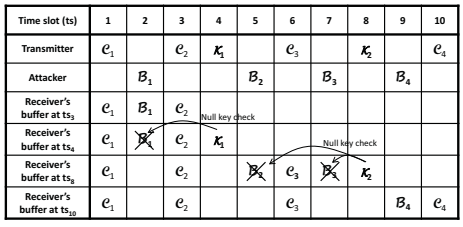
Example 1: We consider a transmission scenario consisting of one transmitter, one receiver, and one attacker. The transmitter wishes to deliver $m = 4$ delay-sensitive packets associated with a deadline (transmission window) $T = 10$ time slots to the receiver. Assume that the receiver has a buffer $|Q| = 6$ packets. If the buffer is full and a new packet arrives, one of the received packets will be discarded uniformly at random. In this example, for the sake of exposition, we assume that there is no loss in transmissions. At the beginning of each time slot, both the transmitter and attacker contend the medium for transmissions. Consider a realization in which the transmitter obtains the channel for transmissions at time slots {1, 3, 4, 6, 8, 10}, and the attacker obtains the channel for transmissions at time slots {2, 5, 7, 9} as shown in Fig. 2. We let $\mathcal{C}_i$, $\mathcal{K}_i$, and $\mathcal{B}_i$ denote the coded packets, null keys, and bogus packets, respectively. We assume that all packets have the same size and each packet is fitted into one time slot.

In the first scheme shown in Fig. 2(a), after time slot ts_4 the receiver obtains four packets $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{B}_3$, and $\mathcal{K}_1$. By using null key $\mathcal{K}_1$, the receiver is able to detect and discard the bogus packet $\mathcal{B}_1$. The receiver then also discards null key $\mathcal{K}_1$ since, with high probability, the attacker might receive $\mathcal{K}_1$ and generate bogus packets that satisfy the checking conditions of $\mathcal{K}_1$.

At time slot ts_9, the attacker transmits another bogus packet $\mathcal{B}_4$. Since the receiver's buffer is full, one of the received packets will be discarded uniformly at random. Because there are more coded packets in the buffer, with high chance one of them will be discarded, assumed $\mathcal{C}_2$. Hence, after nine time slots, the receiver has six packets $\{\mathcal{C}_1, \mathcal{B}_2, \mathcal{C}_3, \mathcal{B}_3, \mathcal{C}_4, \mathcal{B}_4\}$. Assume that at the last time slot ts_{10}, the transmitter transmits another coded packet $\mathcal{C}_5$. Upon received it, the receiver also chooses one of the received packet uniformly at random for discard. It may happen that another coded packet will be discarded, assumed $\mathcal{C}_3$. In the end, the receiver obtains only three coded packets and it cannot recover the transmitted data.



(a)



(b)

Fig. 2. Examples of different transmission schedulings when $m = 4$, $|Q| = 6$, and $T = 10$.

We now show a better transmission strategy in Fig. 2(b). As shown, all the transmissions are scheduled the same as before except at time slot ts_8. Particularly, instead of transmitting coded packet $\mathcal{C}_4$, the transmitter releases another null key $\mathcal{K}_2$. Upon receiving $\mathcal{K}_2$, the receiver uses it to detect and eliminate bogus packets $\mathcal{B}_2$ and $\mathcal{B}_3$. Using this transmission strategy, after ten time slots, the receiver obtains four coded and one bogus packets; thus, with high probability it is able to recover the transmitted data.

Remark 1: In some scenarios, the buffer size may be larger than the transmission window T, hence, there is no received packets dropped. However, the role of adaptive transmitting null keys still plays an important role in helping the receivers to recover the transmitted data. On one hand, if the transmitter sends too many coded packets, more receivers will be able to obtain enough data for decoding. However, existing too many bogus packets in the receivers' buffer makes the decoding expensive. On the other hand, if the transmitter sends more null key packets, some of the receivers may not be able to collect enough data for recovering the transmitted data. That said, there is a nature trade-off in balancing the number of coded and null key packets in transmission. A good transmission scheduling is the one that reduces the cost in recovering the transmitted data while maximizing the average authentication rate across all the receivers.

B. MDP-based Scheduling

Next, we show how the transmitter adaptively schedules transmissions (data packets and null keys) based on perfect feedbacks from the receivers to minimize the decoding cost and maximize the average authentication rate. We assume that at the beginning of each time slot the transmitter receives one-bit feedback from the receivers to indicate whether the previous transmitted packet has been received successfully. The network dynamics is modeled by using the framework of MDP in which the transmitter acts as a decision maker to decide which action to take from an action set at every time slot. We specify the network dynamics by a six-tuple $(\mathbf{S}, \mathbf{A}, T, \mathbf{P}, \mathbf{r}, \gamma)$.

1) State space $\mathbf{S}$: A state s is defined by a matrix

$$s = \begin{bmatrix} NC_1 & NB_1 \\ NC_2 & NB_2 \\ \cdots & \cdots \\ NC_M & NB_M \end{bmatrix}, \quad (3)$$

where the first and second columns of the ith row $[NC_i, NB_i]$ respectively represent the number of coded and bogus packets received by receiver R_i.

2) The action set $\mathbf{A}$ consists of i) sending a coded packet, ii) sending a null key packet, and iii) sending nothing (at time slots the transmitter fails to obtain the transmission channel).
3) The transition distribution $\mathbf{P}(s_{t+1}|s_t, a_t)$ is computed based on the network current state, s_t, action taken, a_t, and packet erasure probability of the channel to each receiver. For example, in the case of one receiver with coded and bogus packet erasure probabilities respectively are p and e, and assume that the transmitter obtains the channel, $\mathbf{P}(s_{t+1} = [00]|s_t = [00], a_t =$ "sending a coded packet") $= p$ and $\mathbf{P}(s_{t+1} = [10]|s_t = [00], a_t =$ "sending a coded packet" $) = 1 - p$.
4) The immediate reward matrix $\mathbf{r}(s, a)$ is computed based on the future-dependent reward function, which is defined by

$$\mathbf{r}(s'|s, a) = \begin{cases} f(.) & \text{if } s' \text{ is a data-recoverable state} \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

where $f(.)$ is a non-negative function, and it is designed to reflect whether the receivers are able to recover the transmitted data, and how easy the processing is carried out. This is similar to a delay reward assignment, i.e., the intermediate states, in which the receiver does not have enough data to recover the transmitted data, should have a zero immediate reward.
5) T is the number of stages or time constraint associated with the current data batch; $\gamma \in [0, 1)$ is discount factor. When γ is close to zero, the transmitter tends to consider only immediate reward, and γ is close to one, the transmitter prefers future reward with higher weight.

Once the parameters are specified, we use the backward induction algorithm [12] to find an optimal policy. The main idea of this algorithm is that it computes the immediate and expected rewards backwardly from the final stage to the initial stage. The optimal policy is the one that results in the maximum expected reward. However, the MDP-based scheduling scheme is subject to the curse of dimensionality when dealing a large space problem as its time complexity is $\mathcal{O}(\mathbf{T}|\mathbf{S}|^2|\mathbf{A}|)$. To tackle this problem, we use the simulation-based method [4] to approximate the optimal policy. In particular, our algorithm combines backward induction with simulation-based methods to reduce the time complexity to $\mathcal{O}(\mathbf{T}\Delta|\mathbf{S}||\mathbf{A}|)$ with $\Delta \ll \mathbf{S}$. The algorithm is summarized in Algorithm 1.

Algorithm 1 : Simulation-based Backward Induction Algorithm (SBIA)

Input: $\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{P}, \mathbf{r}, \gamma, \Delta$.
Output: $\pi^* = \{d(s_1), d(s_2), \ldots, d(s_T)\}$
Initialize: $t = T$, set $V^*(s_T) = 0$ for all $s_T \in \mathbf{S}$
for $t := T - 1$ to 1 **do**
  for each state $s_t \in \mathbf{S}$ **do**
    try all actions $a_t \in \mathbf{A}_{s_t}$ for Δ iterations, and compute
    $\widehat{V}^*(s_t, a_t) = \frac{1}{\Delta} \sum_{\Delta} [\mathbf{r}(s_{t+1}|s_t, a_t) + \gamma V^*(s_{t+1})]$
    $V^*(s_t) = \max_{a_t \in \mathbf{A}_{s_t}} \left\{ \widehat{V}^*(s_t, a_t) \right\}$
    $d(s_t) = \arg\max_{a_t \in \mathbf{A}_{s_t}} \left\{ \widehat{V}^*(s_t, a_t) \right\}$
  end for
end for

Remark 2: So far, we have solved the problem with perfect feedback from the receivers. However, in practice feedback

messages are subject to losses and errors; thus, the transmitter observes only partial of the states. Fortunately, the problem can be formulated and solved as a Partially Observable Markov Decision Process (POMDP). The detail is described in the extended versionm of the paper.

V. Performance Evaluation and Discussions

A. Security Analysis

In what follows, we outline the security analysis of the proposed authentication scheme.

1) The integrity of null key packets: The transmitter signs the root of the Merkle hash tree using its private key K^{-1}, which is included in every null key packet. All the receivers know the public key of the transmitter, and thus can verify the signature, which means that all the receivers can authenticate the root of the Merkle hash tree, whereby to authenticate all the null keys. Therefore, any forged null key packet can be easily detected.

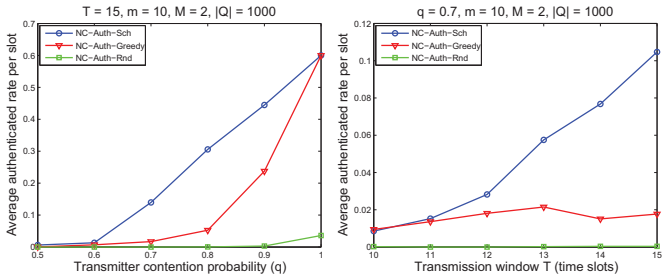
2) The integrity of coded packets: Each coded packet must pass the authentication using all received null keys. Without loss of generality, assume that the transmitter have released $g < n$ null keys, say $\{\mathbf{z}_i\}_{i=1}^{g}$. In the worst case, assume they have also been received by the adversary. The adversary can generate a bogus packet $\mathcal{B}$ such that $\mathcal{B}\mathbf{z}_i = 0$ for all $i \in [1, g]$. Because the adversary cannot predict $\mathbf{z}_{i+1}$, the probability that $\mathcal{B}$ is also orthogonal to the next null key $\mathbf{z}_{i+1}$ is $1/\rho$, where ρ is the underlying field size. In other words, any bogus coded packet can be detected with probability at least $(1 - 1/\rho)$ before being used to decode the original packets.

B. Data Recovery Evaluation

In this subsection, we illustrate the performance gain of the proposed MDP-based scheduling via simulations. For the sake of comparison, we introduce two more schemes: NC-Auth-Greedy and NC-Auth-Rnd. Intuitively, these two schemes are of interest because they have low time complexity and might result in a good performance. In NC-Auth-Greedy scheme, the transmitter only transmits null key packets if existing a buffer overflow. Whereas, in NC-Auth-Rnd scheme, whenever the transmitter has an opportunity for transmission, it will pick a packet uniformly at random among coded and null key packets. In our simulation, we consider the scenario where the buffer size $|Q|$ is larger than the transmission window T. For all simulations, we set $\Delta = 5$, discount factor $\gamma = 0.8$, and packet loss probabilities $p_i = e_i = 0.1$ for $\forall i \in \{1, \ldots, M\}$.

First, we investigate the impact of channel contention probability on the average authentication rate of different schemes in Fig. 3(a). As expected, the average authentication rate increases with the transmitter contention probability q. In particular, the proposed scheme using approximation algorithm SBIA outperforms the others with a significant gap, specially, when the transmitter has more chances to transmit. In addition, if there is no attacker, i.e., $q = 1$, the two schemes NC-Auth-Sch and NC-Auth-Greedy would result in the same performance. This is because, when there is no attacker, NC-Auth-Sch scheme will never transmit null key packets; thus, its transmission is equivalent with that of NC-Auth-Greedy.

Next, Fig. 3(b) depicts the average authentication rate with respect to the transmission window T. First, we note that when the number of time slots available for transmissions is equal to the number of coded packets, the optimal policy is the greedy. This is because there is no time slot for transmitting null key packets, thus, the transmitter sends coded packets at every time slot. Next, as shown in Fig. 3(b), the average authentication rate of NC-Auth-Sch scheme increases significantly with T, especially, when $T > m$. The intuition is that NC-Auth-Sch scheme chooses each packet for transmission to maximize the average authentication rate over the given time horizon T, whereas NC-Auth-Greedy scheme optimizes the transmission for only one time step. As expected, NC-Auth-Rnd scheme has the worst performance because randomly choosing a packet for transmission may waste many transmission opportunities.



(a) (b)

Fig. 3. Average authentication rate per time slot (a) vs. transmitter contention probability q and (b) vs. transmission window T.

VI. Conclusion

In this paper, we have studied the impact of the DoS attacks in a one-hop wireless network, where the transmitter wishes to multicast delay-sensitive data to multiple receivers over lossy channels. In particular, to defend against the false packet injection attack, we propose an efficient authentication mechanism with small overhead and light verification computation, which is resilient to data losses. One of the key ideas in the proposed method is to exploit the null space properties of network coding combined with adaptive scheduling. To find an optimal transmission strategy, we cast the problem in the framework of Markov Decision Processes. The simulation-based backward induction method is then used to approximate an optimal solution. The performance evaluation has been provided to demonstrate the efficacy and efficiency of our proposed scheme.

Acknowledgment

The authors would like to thank Dr. Junshan Zhang and Dr. Yanchao Zhang at Arizona State University for some technical discussions. This work was partially supported by National Science Foundation (#1032567).

References

[1] IEEE Standard for Wireless LAN Medium Acess Control (MAC) and Physical Layer (PHY) Specifications. Nov. 1997. P802.11.

[2] Shweta Agrawal and Dan Boneh. Homomorphic macs: Mac-based integrity for network coding. In *Applied Cryptography and Network Security*, pages 292–305, 2009.

[3] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In *PKC'09*, volume 5443, pages 68–87, 2009.

[4] Hyeong Soo Chang, Michael C. Fu, Jiaqiao Hu, and Steven I. Marcus. *Simulation-based Algorithms for Markov Decision Processes (Communications and Control Engineering)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[5] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, 1997.

[6] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *IEEE INFOCOM'06*, pages 1–13, Bacelona, Spain, April 2006.

[7] T. Ho, M. Medard, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[8] Elias Kehdi and Baochun Li. Null keys: Limiting malicious attacks via null space properties of network coding. In *IEEE INFOCOM*, 2009.

[9] A. Kumar. Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Transactions on Networking*, 6(4):485 –498, August 1998.

[10] R. Merkle. Protocols for public key cryptosystems. In *IEEE S&P'80*, pages 122–134, Oakland, CA,USA, Apr. 1980.

[11] Jung Min, Park Edwin, K. P. Chong, and Howard Jay Siegel. Efficient multicast packet authentication using signature amortization. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 2002.

[12] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics, March 2005.

[13] Pankaj Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the 6th ACM conference on Computer and communications security*, 1999.

[14] Chung Kei Wong, Wong Simon, and Simon S. Lam. Digital signatures for flows and multicasts. In *IEEE/ACM Transactions on Networking*, 1998.