Efficient Key Exchange for Wireless Multi-hop Networks

Iris Safaka, László Czap, Katerina Argyraki
EPFL, Switzerland

Christina Fragouli
EPFL & UCLA

***Abstract*—We present a protocol that enables to create pairwise keys between nodes in a wireless network, so that the keys are secure from an adversary, Eve, with unbounded computational and memory capabilities, but with limited network presence. Our proposed protocol works over arbitrary multi-hop networks, unlike previously proposed protocols that only work when nodes are restricted to be within the same broadcast domain. Multi-hop networks enable to support a wider range of applications; we are inspired in this paper by a traffic-anonymization application. Multi-hop networks are also more interesting technically, as they offer a much richer palette of opportunities for secrecy, that include the inherently noisy nature of wireless, the existence of multiple paths between the network nodes, and interference from concurrent transmissions. We propose a protocol that leverages these opportunities, and show that it can create keys at a rate of tens of Kbps.**

I. Introduction

Current cryptographic key-exchange algorithms are designed around computational hardness assumptions: security breach cannot be achieved in useful time, since current systems do not have sufficient computational power. Recent work has started exploring secret key exchange, where the security of keys does not rely on the bounded computational capabilities of an eavesdropper, Eve, but instead, on her limited network presence: on the fact that Eve will not overhear all transmitted packets in the network due to channel variability and noise. The feasibility of such protocols for creating keys between a single pair of nodes over the wireless channel was demonstrated in [1], [2] and multiple pairs of wireless nodes in [3]. However, these protocols can only be used when all participating nodes are within the same broadcast domain, i.e., in single-hop networks.

In this paper, we look at arbitrary, multi-hop networks and design algorithms to create pairwise keys between all pairs of nodes. Like the protocols in [1], [3], our protocol offers security againts a computationally unbounded adversary, yet without being constrained to a single hop network. This is an important difference from a practical point of view: even when networks have a small number of nodes, as connectivity is impaired from distance, interference and other impediments (e.g. metal obstructions), it is challenging to consistently maintain a single-hop connected network. This is even more so the case when the network size increases to tens of nodes, as can be the case in social applications. We are inspired in this paper by the following use-case scenario, where we need to create keys to ensure privacy: in a street protest, participants use local communication (e.g. WiFi) to cooperate and hide the identity of someone who needs to use cellular Internet connectivity to send reports to the media (and thus may be a target for the authorities eavesdropping the local communication).

Key generation over arbitrary networks is also interesting from a technical point of view. Multi-hop networks provide two new opportunities for secrecy we can leverage: interference and multi-path propagation. Interference between concurrent transmissions (such as caused by the hidden terminal problem) may interfere with Eve's packet reception but not with the reception of other legitimate nodes; distinct packet propagation through multiple paths can ensure that Eve, located in an unknown but fixed position in our network, does not have access to all of them, and again misses packets that legitimate nodes receive. To realize these benefits we need a customized packet dissemination scheme that balances two conflicting goals: spreading the packets as efficiently and as widely as possible among the legitimate nodes, while ensuring that a significant fraction of packets will not be overheard by Eve, who could be located in any place within the network.

We propose in this paper a key-generation protocol that employs such a custom-designed packet dissemination scheme to maximizes Eve's uncertainty, together with linear coding techniques, to generate secure keys. Our protocol is completely decentralized and does not differentiate between nodes; and it simultaneously creates pairwise keys between all node pairs in the network. Through numerical evaluations, we show that our protocol can generate pairwise keys at rates of tens of Kbps on average.

We note that we do not advocate our protocols to substitute existing cryptographic techniques; however, as increasingly the bulk of our data is carried through wireless, and becomes vulnerable to new computational attacks, we believe scientists should start thinking about new techniques and schemes that can complement the existing practices.

II. Setup and Background

A. *System model*

We consider a network of n wireless nodes that form an ad-hoc network. From the nature of wireless, each node's transmission can potentially be received by its neighbors, i.e., all nodes within its transmission radius. Each transmission might get corrupted by noise, channel fading and potentially interference. When we describe our protocol, we consider as *unicast* the transmissions intended to a specific neighbor, and as *broadcast* the transmissions intended to any neighbor. We capture the network structure using two parameters:

• *Density* expresses the expected number of nodes per unit network area; it affects the expected number of neighbors that a node has.

• *Number of hops* describes the maximum distance (in hops) between any two network nodes. More formally, in a k-hop network, for any two nodes (i, j) in the network there exists a k-hop path $i, r_1, \ldots, r_{k-1}, j$ with $k-1$ intermediate nodes such that every node is the neighbor of its preceding node along the path; moreover, there exists at least one pair of nodes for which there is no path with $k-1$ hops.

We make the following simplifying assumptions: the network has a static topology; all nodes have a unique identifier that is revealed to all other nodes in the network; we have a connected network, where there exists at least one path between any two nodes; broadcast transmissions are transmitted only once, while unicast transmissions may be re-transmitted for increased reliability, e.g. as in IEEE 802.11. We make these assumptions to create an easy to attack environment for Eve; our algorithms can extend to mobile and more uncertain environments, yet it becomes challenging to analyze how much (higher) security against Eve we would achieve.

B. Adversary model

We aim for protection against two types of adversaries. First, a passive adversary, who eavesdrops but does not reveal her presence with any form of communication, and can be located anywhere inside the network, at an unknown location. Second, an honest-but-curious node, who legitimately participates in the network and honestly follows the protocols, but tries to breach security using the information at her disposal. In the following we will call the adversary Eve, without specifying (unless needed) if she is a passive eavesdropper or an honest-but-curious node.

We say that Eve experiences an *erasure* or missed a packet (she learns nothing about its content), if the packet is not received by her physical layer (i.e., the reception SNR is not sufficiently high for her physical layer to lock on the physical signal). Otherwise, if a packet is received at the physical layer, we assume that Eve correctly receives it, although the packet may be corrupted and not propagated to the MAC layer.

We also assume that Eve has access to the same physical layer (radio technology, number of antennas etc.) as the legitimate nodes. However, we assume that Eve may have infinite memory as well as unbounded computational capabilities at her disposal; this would follow the model of an adversary that does not want to reveal her identity by using specialized equipment, yet has offline access to unbounded resources to breach security. Moreover, we assume that Eve has perfect knowledge of the protocols, of the network topology, of the node identities, and of whether a transmitted *random* packet (we later give the definition of such packet) was successfully received by a receiver node or not (however, she does not know the content of the packet she has not herself successfully received). Eve, using her knowledge, she can optimally position herself inside the network, and keep her position secret. However, she has limited network presence; in this paper we assume she is situated in a single position.[1]

C. Prior results on key-generation

The idea of linearly combining shared packets for constructing information-theoretically secure keys, has been presented in [1] and applied in a 1-hop setup in [3]. In an initial step, legitimate nodes produce and transmit *random* packets, i.e., packets whose payload consists of randomly produced bits; next, they publicly announce to each other which packets they correctly received. In a second step, the nodes linearly combine their common packets to create keys.

We illustrate the key construction step with an example. Assume that two nodes, Alice and Bob, share the random packets X_1, X_2 and X_3, and suppose they also know that there is at least one packet out of these three that Eve does not have. Then they can both compute $K = X_1 \oplus X_2 \oplus X_3$, which serves as a pairwise key, certainly secure from Eve. Note that Alice and Bob do not need to know *which* packets Eve has, they only need to know *how many* she has. In general, using Maximum Distance Separable (MDS) codes [5], we can securely create as many linear combinations as the number of packets Eve does not have [1].

The key construction step requires that Alice and Bob know how many packets Eve misses out of their commonly shared; in other words, they need to know Eve's channel quality. In a real-world wireless network, where channel conditions are highly unpredictable, Alice and Bob could not possibly have precise knowledge on Eve's receptions, unless an oracle would directly reveal this information to them. Instead, they could attempt to conservatively estimate Eve's knowledge based on side observations; for example based on legitimate nodes' packet receptions, as proposed in [3]. This heuristic approach essentially attempts to model the communication medium by exploiting the knowledge, gained during the public announcements step, on the honest nodes' channel quality.

More formally, any pair of nodes can construct a key, as described next. Let I_i denote the set of indexes of the packets that node i has received in the packet dissemination step. A pair (i, j) computes:

$$c_{ij} = |I_i \cap I_j| \tag{1}$$

$$w_{ij} = \max_{\ell \in \{1\ldots m\} \setminus \{i,j\}} |I_i \cap I_j \cap I_\ell|, \tag{2}$$

where c_{ij} is the number of packets they share and w_{ij} is the maximum number of packets that some other node also has. They also compute an estimation ε_{ij} of the number of packets Eve has overheard out of the c_{ij} common packets. The size of the pairwise key in number of packets is:

$$k_{ij} = c_{ij} - \max\{w_{ij}, \varepsilon_{ij}\}. \tag{3}$$

[1]This assumption can be relaxed to the case where Eve is in multiple positions, as discussed in [3], [4]; yet we need that Eve is not omnipresent, and does not successfully receive all transmissions.

Let G_{ij} be the set of c_{ij} common packets. The pairwise key K_{ij} is then computed as $K_{ij} = G_{ij}H$, where H is a $c_{ij} \times k_{ij}$ MDS matrix that both parties know[2].

D. Goals and performance metrics

The goal of our key exchange protocol is to set up pairwise keys between all $\binom{n}{2}$ pairs of nodes in an arbitrary k-hop network, such that the keys are secure from Eve. We use three metrics to evaluate our protocol [3]:

• *Efficiency* expresses the amount of traffic produced by the protocol in order to generate a key K_{ij}, of length $|K_{ij}|$ bits, between nodes i and j. The achieved efficiency is defined as:

$$\rho_{ij} = \frac{|K_{ij}|}{\text{total transmitted bits}},$$

• *Key rate* measures how many secret bits per second are created between a pair of nodes; the key rate is a function of the efficiency and the transmission rate.

• *Reliability* measures the security of our produced keys. If K denotes a generated key, and E denotes all observations that Eve has after the key-exchange protocol run, the reliability of K is defined as:

$$R_K = \frac{H(K|E)}{H(K)},$$

where $H(K)$ is the entropy of K, and $H(K|E)$ is Eve's uncertainty (conditional entropy given her observations) about K. R_K captures the quality of K, i.e., the amount of information that Eve learns about it by observing the protocol. $R_K = 1$ implies information-theoretical secrecy: $I(K; E) = 0$; in other words by observing the protocol Eve does not learn anything about K.

III. Key exchange protocol

Similarly to [2], [3], our pairwise key exchange consists of two steps: (1) disseminate random packets to create shared randomness (2) compute a pairwise key between any pair of nodes on request. Before giving the protocol description, we illustrate the core design concepts of the packet dissemination step.

A. Leveraging more than channel noise

The goal of the random packet dissemination is to ensure that each pair of nodes share a number of common packets the content of which is not known to Eve. The 1-hop network key-exchange protocols in [2], [3] simply have one or more sources generate random packets and broadcast them. For multi-hop networks, we need a more sophisticated dissemination protocol, that balances two goals: on one hand maximizing the number of random packets between every pair of nodes, and on other other, minimizing the number of packets that Eve overhears. For instance, having a source generate random packets and flooding the network with them does not work well, because Eve ends up overhearing most of these packets, and thus they cannot be exploited for secrecy. We need a protocol that efficiently "creates erasures"; we achieve this by exploiting the following three opportunities wireless networks offer:

1) *Channel noise and fading.* Ideally, we would like the broadcast transmissions to be subject to independent erasures across the receivers so that Eve does not receive exactly the same packets as her close neighbors. To achieve this, in the dissemination protocol we have every node in the network act as a source, to uncorrelate as much as possible the quality of reception from a node's location. Additionally, each source broadcasts each random packet it generates exactly once (without retransmissions). Note that we can do this because we do not care *which* random packets nodes share, only *how many*.

2) *Interference from simultaneous node transmissions.* Such interference for example occurs in the IEEE 802.11 protocol due to the hidden terminal problem. For us this is not a problem but a blessing in disguise: we would like our dissemination protocol to *incur* such interference (yet still not employ a large number of unnecessary re-transmissions).

3) *Multiple paths.* If there are two paths between Alice and Bob in the network, and Eve overhears only one of them, then if Alice sends X_1 on one path and X_2 on the other, Eve will receive only one of the two packets. In general, if Alice and Bob are connected with ν paths, while Eve can overhear at most $z < \nu$ of these (any z), it is optimal for the key generation rate if Alice sends to Bob a *different* packet through each path: Alice and Bob will share ν packets and Eve will learn only z of them [6]. To achieve this, we need a dissemination protocol that sends *each packet through a single path.*

B. Packet dissemination protocol

Parameters and notation: Every node in the network can generate *random* packets and forward these to other nodes. A random packet has a payload of L symbols over a finite field $\mathbb{F}_q$ and thus has a size of $L \log q$ bits. The payload of a random packet is drawn from the uniform distribution. Each packet has a *unique identifier*, that consists of the generator's id together with a sequence number, and a field ttl describing the maximum number of times this packet can be transmitted in the network. Whenever a node performs a transmission of a packet (either generated locally or received by another node) is referred to as the *sender* of this packet. Each node broadcasts at rate $\frac{1}{\lambda}$, where λ is the number of its neighbors. All nodes record all overheard packets.

The packet dissemination protocol works as follows:

1) Every node generates and broadcasts N random packets; it waits a random time between transmissions so that on average it transmits at rate $\frac{1}{\lambda}$. In each packet the *unique identifier* and the ttl are appended.
2) Upon reception of a random packet p, the receiver checks if this is first time it received this packet; if yes, the receiver unicasts an acknowledgment to the sender, otherwise it does not acknowledge.

[2]Note that finding such a matrix H does not require further communication: it can be pre-shared or nodes can use the same deterministic algorithm to compute it.

3) The sender of a packet p selects a forwarder: Let $\mathcal{R}_p$ denote the set of nodes that acknowledged p. The sender chooses a node uniformly at random from $\mathcal{R}_p$, and unicasts a control message to inform the node it is the selected forwarder. If $\mathcal{R}_p = \emptyset$, then p is not forwarded anymore.
4) The selected forwarder of a packet p (the next sender of p), reduces the ttl field by one and broadcasts it.

Steps 2 to 4 are repeated till the ttl field of all the packets in the network expires. Note that when broadcasting a packet p the sender sets a timer T_p, which defines a time window for acknowledging. Once T_p has expired, step 3 takes place.

Discussion: The ttl determines how far a packet will propagate; thus it enables to control the trade-off between creating a large number of common packets between nodes, while keeping Eve's chances of overhearing low. Each node acts as source, so that we generate uniform traffic across the network, and make packet receptions spatially uncorrelated. We transmit at random intervals to incur collisions, and at rate $\frac{1}{\lambda}$ so that, as the density of the network increases, we do not cause congestion. By selecting a single forwarder we avoid flooding and exponential replication of packets; instead, each packet follows a single random walk through the network, so that we exploit multi-path erasures.

C. Computation of pairwise keys

After the packet dissemination step, each legitimate node announces to every other node *which* packets it has received. We refer to this operation as the *feedback* step: Each node i constructs a $1 \times nN$ vector v_i, with a 1 in the mj^{th} position if the node has received packet j from node m, and a 0 otherwise, and *broadcasts* v_i into the network, using special packets indicated as *feedback* packets.

Once the feedback step is completed, on request, any pair of nodes can compute a pairwise key following the procedure described in Section II-C.

Estimating Eve's knowledge: An important step in our protocol is in estimating ε_{ij}, i.e., what fraction of the packets, that a pair of nodes (i, j) shares, was overheard by Eve. Our intuition is that in a multi-hop network, this depends on how "far from each other" the pair of nodes are: nodes that are further apart may collect less common packets; yet among the packets they collect, Eve is likely to have overheard a smaller amount, since she would not intercept the transmissions in all paths that connect them. In addition, Eve will aim to position herself inside the network so as to maximize her probability of eavesdropping as many paths as possible, i.e., a position through which the majority of the available paths pass by.

We estimate ε_{ij} conservatively, exploiting our knowledge of the packets that honest nodes have (or not) received. We form a set $\mathcal{L}$ of the ℓ nodes that have the largest number of neighbors. Let d_{ij} be the distance between nodes i and j in hops and let $\mathcal{P}(d_{ij})$ denote the set of all pairs of nodes in the network with the same distance d_{ij}. Then:

$$\varepsilon_{ij} = \underset{\mathcal{P}(d_{ij})}{\text{avg}} \max_{\ell \in \mathcal{L}} |I_i \cap I_j \cap I_\ell|, \tag{4}$$

where $\text{avg}_{\mathcal{P}(d_{ij})}$ denotes average taken over the set $\mathcal{P}(d_{ij})$. In the above formula: we take the maximum to be conservative; we select the ℓ nodes with most neighbors to be conservative (note that the larger the $\mathcal{L}$ the more conservative we are); we also calculate the average taken over all pairs with distance d_{ij}, because a similar behavior is expected from pairs at the same distance. Note that this estimation can be performed as needed, after the feedback step.

D. Analysis: security of keys

We carry out the analysis for one specific pair of nodes. We refer to Equations 1, 2, 3 and 4; to simplify notation we omit the indexes $_{ij}$ and use k, c, w and ε.

Let e denote the number of packets that Eve has received from the c packets that the selected pair has in common. Lemma 2 of [1] ensures that if $k \leq c - e$, then the resulting key K is perfectly secure, i.e., $R_K = 1$. It means that if nodes knew e, then $\varepsilon = e$ and they could always create a perfectly secret key. However, a passive eavesdropper does not give any feedback, e can only be estimated. Despite of this, we have that $k + w \leq c$, which (by the same lemma) implies that keys are perfectly secure against an honest-but-curious node.

What happens if the estimate of e is not accurate? If $w = \max\{w, e\}$, then for any estimate $k \leq c - e$ holds, meaning that despite of a wrong estimate $R_K = 1$. In the case $e = \max\{w, e\}$, and e is overestimated ($\varepsilon > e$), then again $k < c - \max\{w, e\}$ and again $R_K = 1$. On the other hand, if e is underestimated and $\max\{w, \varepsilon\} < e$, then Eve's uncertainty about the key drops. Expressed in reliability:

$$R_K = \min\left\{1, \frac{c - \max\{w, e\}}{k}\right\}. \tag{5}$$

Using reliability we can express the probability that Eve guesses the key. K has a uniform distribution, hence the probability of correctly guessing a key of size k packets equals $2^{-kR_K L \log q}$. This uncertainty is satisfactory in practice even if $R_K < 1$. E.g. if $R_K = 0.7$ a secret key of length 128 bits gives a security level of $\sim$ 90 bits.

In the evaluation part we show results for the *ideal* efficiency (rate) and for the *effective* efficiency (rate), that is, the efficiency achieved between a pair of nodes (i, j), under perfect knowledge and under estimation of Eve's packets respectively. We also show results for the reliability achieved, which essentially captures the effectiveness of our estimation technique.

E. Communication overhead of the feedback

We assume in this paper that we use an efficient all-to-all broadcast dissemination scheme for the feedback step; indeed, many such schemes have been explored in the literature [7], [8]. In Section IV, we evaluate the key rate achieved by our protocol taking into account only the overhead of the packet dissemination step; thus we do not take into account the overhead of the feedback step that would depend on the particular all-to-all scheme employed. To approximately estimate how much this overhead could reduce our key generation rate, we next perform a back of the envelope calculation.

MAC Layer	Slot Time	$20\mu s$
	W_{min}	31 slots
	W_{max}	1023 slots
	SIFS	$10\mu s$
	DIFS	50 μs
	PHY header	192 bits
	MAC header	272 bits
	DATA frame header	464 bits
	ACK frame	304 bits
PHY Layer	Frequency	2.4 GHz
	Basic Rate	1 Mbps
	Data Rate	36 Mbps
	Tx Power	15 dBm
	Sensitivity Threshold	-81 dBm
	Reception Threshold	-71 dBm
	Reception Model	SNR
	SNR Threshold	15 dB
Channel Model	Propagation Model	TwoRay
	Fading Model	Rayleigh
	Interference Model	AdditiveNoise

Table I
CONFIGURATION OF SIMULATION SETUP

For the dissemination step there are $T_d \simeq n/\lambda \times (NL \log q \times ttl)$ bits transmitted in total[3], with λ here denoting the average number of neighbors. For the feedback step we have $T_f \simeq n[\delta(n-1)+1] \times nN$ bits, where $0 \leq \delta \leq 1$, denoting a forwarding factor for each node, that depends on the broadcast protocol used. Thus, our key rate would be approximately reduced by a constant factor of $1+\mu$, where μ is defined as follows:

$$\mu = \frac{T_f}{T_d} \simeq \frac{\lambda \times [\delta(n-1)+1] \times n}{L \log q \times ttl} \tag{6}$$

Example: Assume a k-hop network with $k=3$ and $n=90$ in which we disseminate random packets of size 1KB and $ttl = 3$, during the dissemination step. In addition, assume we use a network coding technique as described in [8] for the feedback step, for which $\delta = 2/\lambda$ yields an almost 100% packet delivery ratio. In that case, $\mu \simeq 0.67$ meaning that the achieved rate should be divided by a factor of $1+\mu = 1.67$. For the same network and for $n = 135$, the rate should be divided by a factor of $1+1.51 = 2.51$.

IV. EVALUATION

We resorted for our evaluation to a simulator, as we were interested in evaluating the performance of our protocols in large networks (up to 500 nodes) which was not feasible in a test-bed. However, as validation of our simulation environment, we simulated the key generation using parameters that correspond to the 1-hop test-bed and the network conditions in [3], and we found very comparable values.

A. Simulation environment

We use the Java-based, discrete event-driven simulator JiST [9], along with the SWANS library [10], that builds on top of JiST and provides all the elements needed to simulate ad-hoc wireless networks. We also used the extensions and bug-fixes proposed in [11]. In Table I we summarize the configuration parameters of the simulation setup. We use an IEEE 802.11b/g compliant MAC configuration and an SNR frame reception model with an SNR threshold value appropriate for high data rates [12]. The RTS/CTS functionality is by default disabled.

[3]We do not account for re-transmissions, since we assume a MAC layer where re-transmissions are by default disabled in broadcast mode, as in IEEE 802.11.

The signal interference model used in the JiST/SWANS simulator is equivalent to the physical model of successful receptions as defined by Gupta *et al.* in [13]. This feature enables to simulate the hidden-terminal effect and exploit collisions and frame erasures for secrecy.

We simulate a wireless ad-hoc network as a set of n nodes uniformly at random placed on a square area of dimension x meters. All nodes have the same communication capabilities that yield a transmission range of r meters. Under the configuration parameters described above $r \approx 200m$. Therefore, for a k-hop network, as defined in Section II, we set $x = k * \frac{r}{\sqrt{2}}$. We define the *unit area* as an 1-hop area. We consider networks with fixed network density per unit area, that is, for a k-hop area and a given density d we have in total $n = k^2 * d$ nodes.

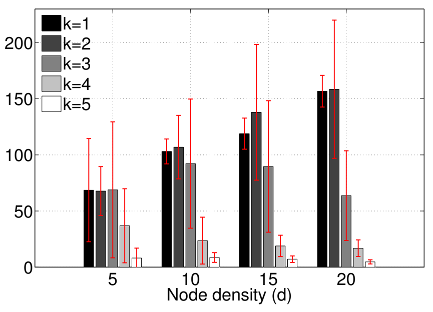
In our protocol, we set $ttl = k$, the maximum distance in the network, and the packet payload to 1KB, so that the resulting MAC frame (including the necessary headers of our protocol and of other layers) does not get fragmented. We also position Eve in each configuration to be in the network center, where we verified that she would have the highest probability to overhear the largest amount of packets. We also note that in the key generation rate calculation we do not take into account the overhead of the feedback step (see Section III-E).

B. Performance evaluation results

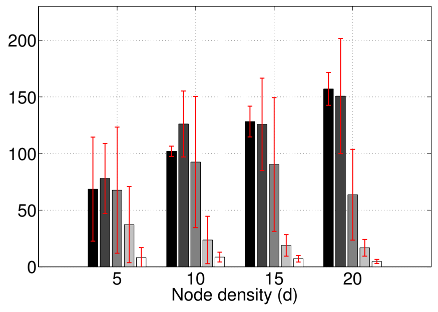
Ideal key generation rate: We begin by examining the ideal key generation rate, that is, the rate achieved by an ideal world alternative protocol. This protocol operates as our key exchange protocol, with the only difference that the nodes are assisted by an oracle to determine precisely the packets that Eve has missed during the packet dissemination step, instead of relying on their own estimations. Therefore, the computed keys are by definition the larger possible and definitively secure. Fig. 1(a) shows the average key generation rate achieved by the oracle alternative, over k-hop networks, with $k = 1 \ldots 5$, as a function of the network density.

First, we observed that in all cases we simulated, we could generate non-zero key rates across (almost) all pairs in the network, with average values as high as tens of Kbps. Notably, we observed that in all our simulations, only 24 pairs of nodes in total experienced zero rate (in particular configurations of 500 nodes, where in each configuration there exist 124750 possible pairs). This is an encouraging result: even in large multi-hop networks, where nodes are up to five hops apart from each other, nodes are still able to share a significant portion of packets and create a non-negligible amount of secure keys.

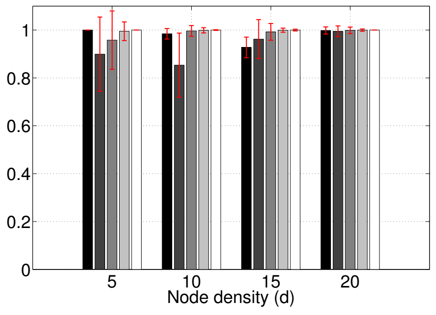
Second, for the 1-hop network, we observe that as the density increases, the key generation rate also increases; this is because we have more nodes acting as sources, thus making packet reception almost independent from a node's location, and hindering Eve from collecting the same packets



(a) Ideal key generation rate (Kbps)



(b) Effective key generation rate (Kbps)



(c) Reliability

Figure 1. Performance evaluation results for the key-generation protocol

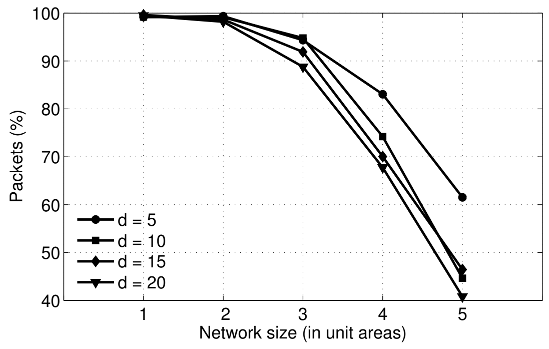


Figure 2. Eve's knowledge on shared packets

as her close neighbors. Uncorrelated packet reception is further assisted by interference and multiple paths existence, as demonstrated by the achieved key rates for a 2-hop network; the more nodes we have the more probable is that two nodes are connected through more than one paths, out of which Eve does not observe at least one.

Finally, for every node density, we observe that as the size of the network increases, namely for $k \geq 3$, the key generation rate significantly drops. This is the aggregated result of two conflicting effects: (1) to create shared randomness over a k-hop network, each packet needs to be transmitted at least k times, which correspondingly reduces the key rate; moreover the amount of common packets that a pair of nodes collects during the packet dissemination phase is smaller, because a smaller percentage of the generated packets reaches both nodes, which in turn reduces the key rate; (2) due to the existence of interference and multiple paths between two nodes in larger networks, Eve observes a smaller fraction of the common random packets that both nodes collect, which boosts the key rate. We verified these effects in our simulations; we show here in Fig. 2 the second effect: we examine what percentage of packets shared between two nodes Eve has also observed (on average), and we find that this percentage decreases with the network size.

Effective key generation rate and Reliability: Fig. 1(b) demonstrates the average key generation rate achieved by our protocol. In contrast to the oracle-assisted protocol, in our protocol the nodes need to estimate how many packets Eve misses, using the technique in Section III-C. We first observe that our protocol can closely follow the oracle-assisted protocol's performance, i.e., our estimator yields rather accurate estimations on Eve's knowledge.

What happens if our protocol either overestimates or underestimates Eve's information? In the former case, the keys generated are shorter than feasibly possible but certainly secure from Eve; the effective key rate is smaller than the corresponding ideal one. In the later case, the keys are larger but may not be ultimately secure from Eve. The security of our generated keys are essentially measure by the level of reliability, as illustrated in Fig. 1(c). We observe that in all cases the reliability was above 0.8, implying a high level of security: reliability 0.8 means that the probability of Eve guessing a key of 1KB is $2^{-0.8\times8000} = 2^{-6400} \simeq 0$ (recall that we use random packets of 1KB and the nodes linearly combine packets of that size to create keys).

V. Related work

Existing information theoretical results characterize the largest achievable secret key rate under a variety of idealized channel models [14]–[17]. The most common setting considers pairwise secret key generation over a single channel with a single sender and one or more receivers. Some results are available for a network setting, most notably secure network coding for an error-free wired network [6]. The secrecy capacity of wireless erasure networks is investigated in [18], but no complete characterization is provided. The closest theoretical result to this work is [1], that considers packet erasure channels with public acknowledgments and uses the key computation algorithm we also employ. Several practical protocols were recently also proposed that build on the symmetry and the randomness extracted from the wireless channel to set up information theoretically secure pairwise keys [19]–[22]. These techniques require node proximity, and thus do not naturally translate to multi-hop networks/multiple keys creation; moreover, they lead to modest key rates. iJam utilizes artificial interference to increase Eve's uncertainty [23]. We also use interference implicitly, as simultaneous transmissions increase the number of collisions and thus the amount of missed packets, but we also leverage noise and path diversity. The setting investigated in [2], [3] can be seen as a special case of a 1-hop network in our terminology. However, as noted earlier, the extension for a multi-hop network requires new techniques and also brings new secrecy opportunities. To our best knowledge the current work is the first to develop protocols for secret key exchange in a multi-hop network that simultaneously exploits channel and network properties.

VI. Conclusions

We proposed a key exchange protocol that enables to simultaneously generate pairwise keys between all pairs of nodes in a large multi-hop network, that are secure from a computationally unbounded, yet with limited network presence, adversary. Our evaluation results indicate the feasibility of high-rate pairwise key generation.

References

[1] M. Jafari Siavoshani, S. Diggavi, C. Fragouli, U. K. Pulleti, and K. Argyraki, "Group Secret Key Generation over Broadcast Erasure Channels," in *Asilomar Conference on Signals, Systems, and Computers*, 2010, pp. 719–723.

[2] S. Xiao, W. Gong, and D. Towsley, "Secure wireless communication with dynamic secrets," in *INFOCOM*, 2010.

[3] I. Safaka, C. Fragouli, K. Argyraki, and S. Diggavi, "Exchanging pairwise secrets efficiently," in *INFOCOM*, 2013.

[4] K. Argyraki, S. Diggavi, M. Duarte, C. Fragouli, M. Gatzianas, and P. Kostopoulos, "Creating Secrets out of Erasures," in *MobiCom*, 2013.

[5] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, 2nd ed. North-holland Publishing Company, 1978.

[6] N. Cai and R. Yeung, "Secure network coding on a wiretap network," *IEEE Transactions on Information Theory,*, vol. 57, no. 1, 2011.

[7] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A performance comparison study of ad hoc wireless multicast protocols," in *INFOCOM*, 2000.

[8] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "A network coding approach to energy efficient broadcasting: from theory to practice," Tech. Rep., 2005.

[9] R. Barr, Z. J. Haas, and R. van Renesse, "JiST: An efficient approach to simulation using virtual machines," *Software: Practice and Experience*, vol. 35, no. 6, pp. 539–576, 2005.

[10] R. Barr, Z. J. Haas, and R. Van Renesse, "Scalable wireless ad hoc network simulation," *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*, 2005.

[11] www.cs.technion.ac.il/~gabik/Jist-Swans/.

[12] G. Pei and T. R. Henderson, "Validation of OFDM error rate model in ns-3," 2010, http://www.nsnam.org/~pei/80211ofdm.pdf. [Online]. Available: http://www.nsnam.org/~pei/80211ofdm.pdf

[13] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, 2000.

[14] A. D. Wyner, "The wire-tap channel," *The Bell system Technical Journal*, vol. 54, no. 8, pp. 1355–1387, 1975.

[15] U. Maurer, "Secret key agreement by public discussion from common information," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 733–742, May 1993.

[16] I. Csiszár and P. Narayan, "Secrecy capacities for multiterminal channels," *IEEE Transactions on Information Theory*, vol. 54, no. 8, 2008.

[17] E. Ekrem and S. Ulukus, "Secrecy Capacity of a Class of Broadcast Channels with an Eavesdropper," *EURASIP J. Wireless Comm. and Networking*, 2009.

[18] A. Mills, B. Smith, T. Clancy, E. Soljanin, and S. Vishwanath, "On secure communication over wireless erasure networks," in *IEEE International Symposium on Information Theory (ISIT)*, 2008, pp. 161–165.

[19] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener, "Robust key generation from signal envelopes in wireless networks," in *ACM conference on Computer and communications security*, 2007.

[20] C. Ye, S. Mathur, A. Reznik, Y. Shah, W. Trappe, and N. B. Mandayam, "Information-theoretically secret key generation for fading wireless channels," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 240–254, 2010.

[21] J. Croft, N. Patwari, and S. K. Kasera, "Robust uncorrelated bit extraction methodologies for wireless sensors," in *ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010.

[22] H. Liu, Y. Wang, J. Yang, and Y. Chen, "Fast and practical secret key extraction by exploiting channel response," in *INFOCOM*. IEEE, 2013.

[23] S. Gollakota and D. Katabi, "Physical layer wireless security made fast and channel independent," in *INFOCOM*, 2011.