Jamming-Resilient Secure Neighbor Discovery in Mobile Ad Hoc Networks

Rui Zhang, *Member, IEEE*, Jingchao Sun, *Student Member, IEEE*, Yanchao Zhang, *Senior Member, IEEE*, and Xiaoxia Huang, *Member, IEEE*

***Abstract*—Secure neighbor discovery is fundamental to mobile ad hoc networks (MANETs) deployed in hostile environments and refers to the process in which two neighboring nodes exchange messages to discover and authenticate each other. It is vulnerable to the jamming attack in which the adversary intentionally transmits radio signals to prevent neighboring nodes from exchanging messages. Anti-jamming communications often rely on spread-spectrum techniques, which depend on a spreading code common to the communicating parties but unknown to the jammer. The spread code, however, is impossible to establish before the communicating parties successfully discover each other. While several elegant approaches have been recently proposed to break this circular dependence, the unique features of neighbor discovery in MANETs make them not directly applicable. In this paper, we propose JR-SND, a jamming-resilient secure neighbor discovery scheme for MANETs based on direct-sequence spread spectrum and random spread-code predistribution. JR-SND enables neighboring nodes to securely discover each other with overwhelming probability despite the presence of omnipresent jammers. Detailed theoretical and simulation results confirm the efficacy and efficiency of JR-SND.**

***Index Terms*—Jamming, secure neighbor discovery, MANET.**

I. Introduction

SECURE neighbor discovery is a fundamental functionality in mobile ad hoc networks (MANETs) deployed in hostile environments [2]. It refers to the process that neighboring nodes exchange messages to discover and authenticate each other. As the basis of other network functionalities such as medium access control and routing, secure neighbor discovery need be frequently performed due to node mobility.

The open wireless medium in MANETs renders secure neighbor discovery particularly vulnerable to the jamming attack, in which the adversary intentionally transmits noise-like signals to prevent neighboring nodes from exchanging messages and thus discovering each other. Traditional anti-jamming communications often depend on spread-spectrum techniques [3], which all require that the communicating parties use a common spread code (unknown to the adversary) to spread the signals such that the transmissions are unpredictable and thus resilient to jamming.

Applying spread-spectrum techniques for jamming-resilient secure neighbor discovery in MANETs, unfortunately, faces several challenges. One the one hand, if all nodes share a common spread code, the adversary can acquire the spread code after compromising any node, which leads to a single point of failure. On the other hand, if each pair of nodes share a unique code so that compromising any node would not affect the spread codes shared between non-compromised node pairs, two neighboring nodes, however, do not know which spread code to use if jamming takes place before they successfully discover and authenticate each other. This situation thus leads to a circular dependency.

There are a few recent attempts such as [4]–[13] to break the circular dependency between anti-jamming communications and spread-code establishment. The unique features of MANET neighbor discovery, however, make these elegant solutions unsuitable. In particular, since node encounters are unpredictable in MANETs, each node must be always prepared to accept and validate potential neighbor discovery requests. The existing solutions [4]–[13] all depend on some publicly known communication strategies such as public spread-code sets. The adversary can thus use such public knowledge to inject arbitrary many neighbor discovery requests in the whole network, leading to a special Denial-of-Service (DoS) attack in which all nodes are forced to perform endless verifications of neighbor discovery requests (which often involve expensive digital signature verifications). Moreover, nodes may encounter for only a short while due to high mobility. This requires neighbor discovery to be done in a very short time, say a few seconds, while most existing solutions do not meet this requirement.

The above situation motivates us to design a novel solution specially tailored for jamming-resilient secure neighbor discovery in MANETs. Our key observation is that most MANETs are inherently different from the civilian applications targeted by [4]–[13]. Specifically, MANETs in hostile environments such as the battlefield are normally controlled by the same authority. It is thus feasible to preload every node with some secret spread codes shared with a few others for subsequent anti-jamming communications in the field. Such spread-code pre-distribution is nevertheless infeasible in civilian networks which lacks a single authority and features dynamic join and leave of unknowns.

Manuscript received November 21, 2014; revised February 23, 2015; accepted May 15, 2015. Date of publication June 1, 2015; date of current version October 8, 2015. A preliminary version of this paper was presented at IEEE ICDCS, Minneapolis, MN, USA, June 2011 [1]. This work was supported in part by the U.S. National Science Foundation under Grants CNS-1421999, CNS-1320906, CNS-1117462, and CNS-1422301. The work of X. Huang was supported by the Joint Program of the National Natural Science Foundation of China-Guangdong under Grant U1301256. The associate editor coordinating the review of this paper and approving it for publication was Y. Chen.

R. Zhang is with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822 USA (e-mail: ruizhang@hawaii.edu).

J. Sun and Y. Zhang are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: jcsun@asu.edu; yczhang@asu.edu).

X. Huang is with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: xx.huang@siat.ac.cn).

Digital Object Identifier 10.1109/TWC.2015.2439688

In this paper, we propose JR-SND, a novel jamming-resilient secure neighbor discovery scheme for single-authority DSSS-based MANETs. Inspired by random key pre-distribution schemes for sensor networks such as [14], JR-SND requires the MANET authority to generate a pool of secret spread-codes and pre-load every node with a constant number of spread codes randomly drawn from the pool prior to network deployment. During the network operation, two neighboring nodes can use their spread codes to conduct anti-jamming secure neighbor discovery via DSSS communication. In particular, JR-SND allows two neighboring nodes to directly discover each other if they share at least one common spread code unknown to omnipresent jammers or indirectly discover each other if there exists a multi-hop path connecting them, along which every two neighboring nodes have successfully discovered each other. As time goes by, the adversary may compromise some nodes to know their spread codes, but the non-compromised codes will remain secret. Compared to prior work [4]–[13] in terms of their use in secure neighbor discovery, JR-SND can greatly mitigate the aforementioned DoS attack because it can only be launched by the adversary using limited compromised spread codes which can fortunately be revoked after being identified.

Our main contributions are summarized as follows.

- We identify jamming-resilient secure neighbor discovery in MANETs as a related problem that cannot be addressed by existing anti-jamming techniques such as [4]–[13].
- We propose a novel Jamming-Resilient Secure Neighbor Discovery (JR-SND) scheme by combining DSSS with key pre-distribution [14], which is the first of its kind. JR-SND consists four components, including a new random spread-code pre-distribution scheme that supports the fine control of the impact of compromised spread codes, a direct neighbor discovery protocol and two indirect neighbor discovery protocols, which jointly enable any two neighboring nodes to quickly discover each other with high probability under severe jamming attacks.
- We confirm the efficacy of JR-SND by theoretical analysis and extensive simulation studies.

The rest of this paper is structured as follows. Section II discusses the related work. Section III briefly introduces the background of DSSS. Section IV introduces our network and adversary models. Section V illustrates the design of JR-SND. Section VI presents the performance evaluation of JR-SND, and Section VII concludes this paper.

II. Related Work

Several schemes have been proposed to enable two nodes to establish a secret spread code (or key) under the jamming attack. In their seminal work [5], Strasser *et al.* proposed using Uncoordinated Frequency Hopping (UFH) to enable two communication parties without a common secret to establish a secret key for the use of subsequent more efficient FHSS communications. This technique was later improved in [6], [7], [12], [13] to reduce the key-establishment latency and communication overhead. Under the above techniques, the adversary can inject arbitrary many message fragments leading to a DoS attack. In [8], Jin *et al.* addressed the same problem by proposing an intractable forward-decoding and efficient backward-decoding scheme based on DSSS. Their scheme, however, requires the sender to know the MAC address of the receiver which is unfortunately unknown before the sender successfully discovers the receiver.

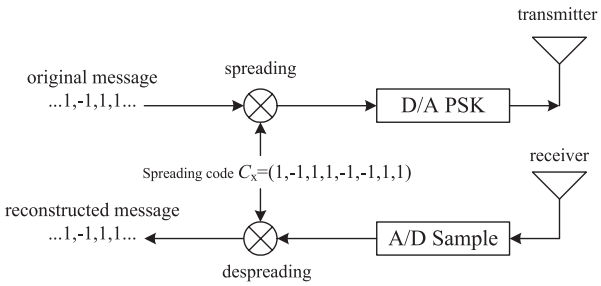

Fig. 1. A simplified system diagram of DSSS.

Another line of research has been devoted to enable jamming-resistant broadcast communication. The schemes proposed in [9]–[11] are all based on DSSS and a publicly known spread-code set and thus vulnerable to the DoS attack mentioned earlier. In [15], Xiao *et al*. proposed a collaborative UFH scheme in which earlier receivers of a broadcast message serve as relays for other nodes, and the scheme was subsequently analyzed in [16]. More recently, jamming-resistant communication against inside attacker has been studied in [17]–[19]. These schemes are based on frequency hopping and thus cannot be directly applied to our target scenarios.

There are also some work that are loosely related to ours. Anti-jamming in general CDMA systems have been investigated in [20], [21]. In [22], Lu *et al*. proposed a scheme that achieves the minimum message transmission latencies for smart grid systems under jamming attack. More recently, Yan *et al*. [23] proposed an anti-jamming scheme that explores interference cancellation and transmit precoding capabilities of MIMO technology.

III. Background on DSSS

DSSS is a modulation technique widely used in code division multiple access (CDMA) systems, e.g., IS-95. In a DSSS system (see Fig. 1), the sender *spreads* the data signal by multiplying it by an independent "noise" signal known as a *spread code*, which is a pseudorandom sequence of "1" and "-1" bit values at a frequency much higher than that of the original signal. The energy of the original signal is thus spread into a much wider band. The receiver can reconstruct the original signal by multiplying the received signal by a synchronized version of the same spread code, which is known as a de-spreading process.

To transmit a message, the sender first transforms the message into a non-return-to-zero (NRZ) sequence by replacing each bit "0" with "-1" and then multiplies each bit of the message by a spread code to get the spread message also known as the chip sequence. For example, if the message to be transmitted is "10," and the spread code is "$+1-1-1+1$," the resulting chip sequence is "$+1-1-1+1-1+1+1-1$." The chip

sequence is then converted into the RF signal through the D/A converter and the PSK modulator and finally transmitted.

On the receiver side, similar operations are performed in a reverse order. The receiver first samples and demodulates the received signal and then performs A/D conversion to obtain the chip sequence. The receiver then computes the *correlation* between the obtained chip sequence and the shared spread code, where the correlation between two NRZ sequences $(u_1, \cdots, u_N)$ and $(v_1, \cdots, v_N)$ is defined as $\frac{1}{N}\sum_{i=1}^{N} u_i v_i$. Note that the correlation of two identical sequences is one, while the expected correlation of two independent random sequences is zero. A correlation higher (lower) than a predefined threshold τ $(-\tau)$ indicates a bit 1 (-1). For example, if pseudorandom sequences of $N = 512$ bits are used as spread codes, τ can be set 0.15 to ensure correct de-spreading [9].

IV. Network and Adversary Models

A. Network Model

We consider a MANET consisting of n nodes deployed in some hostile environment such as the battle field. For simplicity, we assume that each MANET node has two DSSS antennas with a transmission speed of R b/s for anti-jamming communications, one for receiving and the other for transmitting. The extension of JR-SND to an arbitrary number of antennas is left as future work. Due to node mobility, every node need periodically perform neighbor discovery to discover others within its transmission range. We assume that each node can monitor the transmission activities associated with a few spread codes in real time, each shared with one unique neighbor. This assumption has been made in existing CDMA transmitter-based MAC protocols [24] and can be easily realized by hardware. This implies that an incoming message spread using the code under real-time monitoring can be de-spread with negligible delay. However, if a node has many spread codes, it may not be able to simultaneously monitor all of them and have to buffer the incoming signals for off-line de-spreading processing in case that these signals can be de-spread using some spread codes that are not currently being monitored. In addition, if a node does not detect any transmission with any code under real-time monitoring for a threshold amount of time, it will stop monitoring that code under the assumption that the corresponding neighbor has moved out of its transmission range.

As in [9], we choose pseudorandom codes for DSSS communications for their ease of generation and good auto/cross-correlation properties. We assume that the concurrent transmissions spread with different pseudorandom codes interfere with each other with negligible probability, which holds if the length of spread codes is sufficiently large, e.g., $N = 512$. We refer the readers to [9] for more detailed discussions about pseudorandom codes.

To prevent the adversary from impersonating legitimate nodes, neighbor discovery must be conducted in a secure fashion such that two nodes accept each other as mutual neighbors after authenticating each other's credentials issued by the MANET authority. Throughout the paper, by saying two nodes successfully discover each other, we mean that they physically detect each other's existence and also achieve mutual authentication. There are many mutual authentication methods that suffice our purpose and often involve a three-way handshake between two involved nodes. To ease our presentation, we assume an approach as in [25] based on Identity-Based Cryptography [26], in which each node A has an ID ID_A as its public key and an ID-based private key K_A^{-1} obtained from the authority before network deployment. Note that, however, JR-SND can also rest on many other mutual authentication schemes.

B. Adversary Model

We focus on defending against jamming attack in this paper and refer readers to existing rich literature for defenses against other important attacks such as wormhole attack [27]–[30].

We assume an omnipresent adversary or jammer $\mathcal{J}$ aiming to jam neighbor discovery and thus prevent neighboring nodes from discovering each other anywhere in the network. $\mathcal{J}$ is assumed to be computationally bounded, which means that if $\mathcal{J}$ does not know the spread code being used, it is infeasible for him to recover it by exhaustive search within the network lifetime. This assumption is common in DSSS systems [8], [9] and holds if the spread code is sufficiently long, e.g., $N = 512$. JR-SND relies on a large set of random spread codes chosen by the MANET authority, which are initially all kept secret from $\mathcal{J}$. As time goes by, $\mathcal{J}$ may compromise some MANET nodes and acquire the secret codes held by them. Compared to unattended sensor nodes in sensor networks, MANET nodes are more powerful and often carried and used by humans such as soldiers so that they can be under good self and mutual monitoring. It is reasonable to assume that $\mathcal{J}$ can only compromise a small fraction (say, up to 5%) of MANET nodes. JR-SND does not work well if this assumption does not hold.

To jam an ongoing DSSS transmission spread with any spread code, $\mathcal{J}$ need transmit using the same code and also synchronize with the target transmission. As in [9], we assume that $\mathcal{J}$ can always recover chip synchronization without de-spreading a message, which can be realized by energy detectors or modulation-specific characteristics. In other words, $\mathcal{J}$ only need determine which spread code to use to jam the transmission. We focus on two types of jammers.

- *Random jammer*: whenever $\mathcal{J}$ detects an ongoing transmission, $\mathcal{J}$ jams it with a random compromised spread code.
- *Reactive jammer*: whenever $\mathcal{J}$ detects an ongoing transmission, $\mathcal{J}$ first tries to identify which spread code is being used. If the code is successfully identified, it then uses it to jam the rest of the message.

Random jamming places no additional requirements on $\mathcal{J}$'s computation capability, while reactive jamming requires $\mathcal{J}$ to identify the correct spread code being used before the end of the targeted message transmission.

Besides the jamming attack, the adversary may also exploit the operations of JR-SND to launch the DoS attack by injecting arbitrary fake neighbor-discovery requests to occupy legitimate nodes with endless verifications of these fake requests. JR-SND is highly resilient to this DoS attack, as will be manifested later.

To simplify the analysis, we assume that $\mathcal{J}$ consists of multiple jamming devices with similar transmitters to those of legitimate nodes. We further assume that $\mathcal{J}$ can transmit at most z signals in parallel to attempt jamming any targeted neighbor-discovery message, where $z \ll N$. Without this limitation on $\mathcal{J}$'s capability, $\mathcal{J}$ can jam any targeted transmission without knowing the spread code by simply transmitting noise signals using z transmitters concurrently [8], in which case there is no workable solution.

V. The JR-SND Design

In this section, we present the design of JR-SND. We first present a quorum-based spread-code pre-distribution scheme as a nontrivial adaption of existing key pre-distribution schemes [14]. We then present a direct neighbor-discovery protocol (D-NDP), a multi-hop neighbor-discovery protocol (M-NDP), and a location-aware multi-hop neighbor-discovery protocol (LAM-NDP). The following terms will be used throughout.

- *Physical neighbors*: Two nodes are called *physical neighbors* if they are in each other's transmission range.
- *Logical neighbors*: Two nodes are called *logical neighbors* if they have discovered each other after executing JR-SND.

A. Random Spread-Code Pre-Distribution

Before network deployment, the MANET authority generates a pool of $s \ll 2^N$ random spread codes, denoted by $\mathbb{C} = \{\mathbf{C}_i\}_{i=1}^{s}$. Only the authority has the full knowledge of $\mathbb{C}$. The authority then uses the following method to distribute m spread codes to each node such that any $\mathbf{C}_i \in \mathbb{C}$ is shared by no more than l nodes, where the choice of l will be discussed later.

The distribution process consists of m rounds, during each of which each node is assigned one spread code. Specifically, let us temporarily assume that $n = lw$ for some integer w and then $s = wm$. In each round $i \in [1, m]$, the authority randomly partitions the n nodes into w subsets of equal cardinality l and then assigns $\mathbf{C}_{w(i-1)+j}$ to all the nodes in the jth subset. It is easy to see that after m rounds, every node is preloaded with m spread codes, and every code is exactly shared by l nodes. We will denote by $\mathbb{C}_A$ the set of spread codes of node A.

Now we consider the case where n cannot be divided by l, i.e., $n = lw - l'$ for some $0 < l' < l$. In this case, the authority can introduce l' virtual nodes during spread-code pre-distribution. This will only result in some codes being shared by less than l nodes and thus will not affect the performance very much. Moreover, the $l - l'$ virtual nodes can be reserved for nodes joining the network later.

Our scheme permits new nodes to join the network later. In particular, the authority can assign the spread codes of a virtual node to a unique new node. If there are more than l' new nodes, the authority can conduct the previous distribution process for each additional w new nodes with existing s codes, which will result in every code being shared by one more node. We do not expect too many new nodes in the target scenario, so the number of nodes sharing any code will be only slightly larger than l.

B. D-NDP: Direct Neighbor Discovery Protocol

We now introduce D-NDP by which two physical neighbors with common spread codes can directly discover each other.

During the network operation, each node periodically initiates neighbor discovery in a randomized manner. Specifically, in every interval of length T, each node initiates the D-NDP process once at a random time point. Below we use nodes A and B as an example to illustrate the process. We assume that they share at least one secret spread code, say $\mathbf{C}_i \in \mathbb{C}_A \cap \mathbb{C}_B$.

Assume that A initiates the D-NDP process prior to B. Starting from a random time point, A repeatedly broadcasts a HELLO message for r rounds, where r is a system parameter. In each round, the HELLO message is broadcasted m times, and each time a distinct code in $\mathbb{C}_A$ is used for spreading. For example, the HELLO message spread with $\mathbf{C}_i$ is

$$A \rightarrow * : \{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i},$$

where HELLO is a message type identifier of l_t bits, ID_A is A's ID, and $\{\}_*$ denotes the message spread with the spread code at the subscript. Each message in D-NDP is encoded with an error-correcting code (ECC) such as [31] to increase the transmission reliability. In particular, assuming that each node ID is of l_{id} bits, the original message is thus of $l_t + l_{\text{id}}$ bits. Node A then applies ECC to generate an encoded message of $l_h = (1 + \mu)(l_t + l_{\text{id}})$ bits, where $\mu > 0$ is a system parameter. This ECC method can tolerate up to a fraction of $\mu/(1 + \mu)$ bit errors or losses, which means that $\mathcal{J}$ must use the correct spread code $\mathbf{C}_i$ to jam at least $\mu(l_t + l_{\text{id}})$ bits to prevent B from decoding $\{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i}$.

It takes roughly time $t_h = l_h N/R$ to broadcast one HELLO message spread with one spread code and mt_h to finish one round, where R is the chip rate. There are r copies of the HELLO message spread with the same code. It is possible that B may miss the head of one $\{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i}$ copy due to improper synchronization or other reasons. However, as long as B buffers the incoming signals for a duration of at least $t_b = (m + 1)t_h$, it can certainly buffer a complete $\{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i}$.

To synchronize with and de-spread any incoming message, node B buffers the received signal and tries to identify any message in the buffer using a sliding window algorithm similar to the one used in [9]. Specifically, assume that B has buffered f chips of the incoming signal, denoted by $(p_1, \cdots, p_f)$, in which the first complete $\{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i}$ may start at any chip position. To locate it, B computes the correlation between $(p_i, \cdots, p_{i+N-1})$ and each spread code in $\mathbb{C}_B$, for all $1 \le i \le f$. The correlation between the sequence $(p_i, \cdots, p_{i+N-1})$ and code $\mathbf{C}_i$ higher (or lower) than the predefined threshold τ (or $-\tau$) for the smallest i indicates a bit "1" or "-1" spread with $\mathbf{C}_i$ starting at chip position p_i and thus the beginning of $\{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i}$. Node B then uses $\mathbf{C}_i$ to de-spread the rest of the message, i.e., computes the correlation between $(p_{i+jN}, \cdots, p_{i+(j+1)N-1})$ and $\mathbf{C}_i$ to de-spread the $(j + 1)$th bit, for all $1 \le j < l_h$.

Now we discuss the choice of r. The challenge here is that we cannot simply assume that each node can monitor the transmission activities associated with arbitrary many spread codes in real time, which otherwise requires very complex and

expensive hardware [24]. Therefore, we must take into account nodes' computation capability. Assume that it takes time ρN to compute the correlation between two chip sequences of N bits, where ρ is a constant determined by each node's computation capability. For example, if each receiver can compute 4.7×10^8 correlations of two binary sequences of 256 bits as assumed in [9]. We thus have $\rho \approx 8.3 \times 10^{-12}$ s/bit in such cases and anticipate an even higher ρ in practice. Since node B buffers totally $f = Rt_b$ incoming chips each time, it takes up to $t_p = \rho NmRt_b$ to scan all the chip positions, each requiring computing m correlations. Note that there may be multiple or no valid HELLO messages in the buffer. The former and latter cases correspond to multiple or no nodes initiating neighbor discovery with B, respectively. Therefore, even after recovering one valid HELLO message from the buffer, B still need process the rest of it. Let $\lambda = t_p/t_b = \rho NmR$ be the ratio between processing time and buffering duration. For example, if $N = 512$, $m = 1000$, and $R = 22$ Mbps, we have $\lambda \approx 94$ in the above example, which indicates the huge gap between the receiving and processing capabilities. To accommodate this gap, each node independently maintains a simple buffering and processing schedule as follows. During each duration $[it_p, (i+1)t_p]$, for all $i \geq 1$, it processes the signal buffered during $[it_p - t_b, it_p]$ and immediately deletes processed chips; it also buffers the signal arriving during $[(i+1)t_p - t_b, (i+1)t_p]$. It can be easily shown that the buffer will not overflow with this schedule. Under this schedule, it suffices to let A broadcast the HELLO message for a total duration of $rmt_h = (\lambda+1)t_b = (\lambda+1)(m+1)t_h$ to ensure that node B buffers a complete $\{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i}$, so we have $r = \lceil(\lambda+1)(m+1)/m\rceil$.

After de-spreading $\{\mathsf{HELLO}, ID_A\}_{\mathbf{C}_i}$, node B knows that A is in its transmission range and $\mathbf{C}_i \in \mathbb{C}_A \cap \mathbb{C}_B$. It then repeatedly sends an ECC-coded CONFIRM message spread with $\mathbf{C}_i$,

$$B \rightarrow A : \{\mathsf{CONFIRM}, ID_B\}_{\mathbf{C}_i}.$$

Node B then starts to monitor $\mathbf{C}_i$ in real time. Similar to A transmitting HELLO message, node B keeps transmitting the CONFIRM message for $t_p = \rho NmRt_b$ or until receiving a response from A which can be de-spread with $\mathbf{C}_i$. If B does not receive a response before its timer expires, it stops monitoring $\mathbf{C}_i$ in real time and considers A having moved away.

Node A uses the same approach to de-spread B's CONFIRM message and knows that B shares $\mathbf{C}_i$ with it. Because $\mathbf{C}_i$ may also be known by up to $l-2$ other nodes, A and B cannot authenticate each other. To conduct mutual authentication, node A computes a shared key K_{AB} using its ID-based private key K_A^{-1} and ID_B [25]. It then sends to B the following ECC-coded message spread with $\mathbf{C}_i$,

$$A \rightarrow B : \left\{ID_A, n_A, f_{K_{AB}}(ID_A|n_A)\right\}_{\mathbf{C}_i},$$

where n_A is a random nonce to defend against message replay attacks, $f_*(\cdot)$ denotes a message authentication code (MAC) with the key at the subscript, and $|$ denotes concatenation.

Since B is currently monitoring $\mathbf{C}_i$, it can de-spread the above response in real time after negligible delay. Node B proceeds to compute a shared key K_{BA} based on its ID-based private key K_B^{-1} and ID_A, which is equal to K_{AB} according to [25]. Then B uses K_{BA} to compute $f_{K_{BA}}(ID_A|n_A)$ and compares it with the received $f_{K_{AB}}(ID_A|n_A)$. If they are equal, B knows that A has computed the same key, which means that A is an authenticated logical neighbor with a valid ID-based public/private key pair issued by the MANET authority. It is worth noting that no nodes other than A and B could compute the shared key K_{AB} [25]. Node B proceeds to transmit the following ECC-coded response

$$B \rightarrow A : \left\{ID_B, n_B, f_{K_{BA}}(ID_B|n_B)\right\}_{\mathbf{C}_i},$$

where n_B is the random nonce chosen by B. Node B then computes a session spread code as $\mathbf{C}_{BA} = h_{K_{BA}}(n_B \otimes n_A)$ and starts monitoring $\mathbf{C}_{BA}$ in real time, where $h_*(\cdot)$ is a cryptographic hash function of N bits keyed with the subscript and $\otimes$ denotes bitwise XOR operation.

After de-spreading the above response, node A verifies $f_{K_{BA}}(ID_B|n_B)$ using K_{AB} similar to what B does. If the verification is successful, A accepts B as a logical neighbor and also computes $\mathbf{C}_{AB} = h_{K_{AB}}(n_A \otimes n_B)$ which is equal to $\mathbf{C}_{BA}$. Finally, A starts to monitor $\mathbf{C}_{AB}$ in real time.

In the cases that B shares $x \geq 2$ spread codes with A whereby to de-spread multiple copies of the HELLO message, D-NDP employs a redundancy design that lets B use all the x shared codes to sequentially spread the same CONFIRM message. The last two messages both are also spread by A and B with all the x codes sequentially. In other words, we can consider the execution between A and B as x separate sub-sessions involving the same four messages and establishing the same session spread code. This redundancy design can greatly enhance the jamming resilience of neighbor discovery which fails only if all the x sub-sessions fail. Consider the following example. Assume that among $x \geq 2$ shared codes, $x-1$ of them are compromised. The D-NDP execution will succeed under blind reactive jamming since B can only receive the HELLO message spread with the non-compromised code whereby to spread the subsequent messages. However, a more intelligent attack is that $\mathcal{J}$ does not jam the HELLO message but only targets at the later three transmissions. Assuming B receives x copies of the HELLO messages and randomly chooses one code from total x codes to spread the CONFIRM message, it is very likely that a compromised code will be selected. In such cases, $\mathcal{J}$ may jam the later message transmission, leading to a D-NDP failure. Under our design, this intelligent attack can no longer succeed.

C. M-NDP: Multi-Hop Neighbor Discovery Protocol

Two physical neighbors may fail to directly discover each other via D-NDP either because they have no common spread codes or because $\mathcal{J}$ has compromised their common spread codes whereby to successfully jam the D-NDP message transmissions. Now we introduce M-NDP that allows two physical neighbors to indirectly discover each other as long as there is a *jamming-resilient path* connecting them, along which every two adjacent nodes have discovered each other.

We illustrate the M-NDP operations with the scenario in Fig. 2 as an example, where both solid and dashed line segments represent jamming-resilient paths, and A and B cannot directly discover each other via D-NDP.

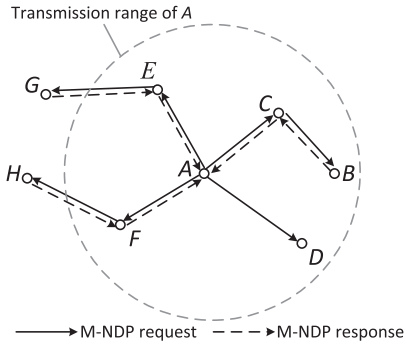

Fig. 2. Illustration of M-NDP.

As in D-NDP, all nodes need to periodically initiate the M-NDP process at some random time point of its own choice. Assume that A initiates the M-NDP process prior to B. Let $\mathcal{L}_A$ denote the set of logical neighbors of A, where $\mathcal{L}_A = (C, D, E, F)$ in Fig. 2. Node A unicasts an M-NDP request to each node in $\mathcal{L}_A$. For example, the request sent to C is

$$A \rightarrow C : \{ID_A, \mathcal{L}_A, n_A, \nu, \mathbf{SIG}_{K_A^{-1}}\}_{\mathbf{C}_{AC}},$$

where n_A is a random nonce, $\nu \geq 1$ is a parameter chosen by A determining the maximum number of hops the request can traverse, $\mathbf{SIG}_*$ denotes a digital signature operation on the prior data with the private key at the subscript, and $\mathbf{C}_{AC}$ is the session spread code shared between A and C.

After receiving the M-NDP request, C first verifies the signature $\mathbf{SIG}_{K_A^{-1}}$ using ID_A as the public key [25]. If the signature verification succeeds, C compares $\mathcal{L}_A$ with its own logical neighbor list $\mathcal{L}_C$. Then for each node in $\mathcal{L}_C - \mathcal{L}_A$, say B, node C unicasts a modified request

$$C \rightarrow B : \{ID_A, \mathcal{L}_A, n_A, \nu, \mathbf{SIG}_{K_A^{-1}}, ID_C, \mathcal{L}_C, \mathbf{SIG}_{K_C^{-1}}\}_{\mathbf{C}_{CB}}.$$

Upon receiving this new request from C, node B first verifies the signatures $\mathbf{SIG}_{K_C^{-1}}$ and $\mathbf{SIG}_{K_A^{-1}}$ using ID_C and ID_A as the public keys, respectively. If both verifications succeed, B further checks whether $C \in \mathcal{L}_A \cap \mathcal{L}_B$, i.e., whether C is indeed the common neighbor of A and B. If not, B discards the message; otherwise, B returns the following M-NDP response to C,

$$B \rightarrow C : \{ID_A, ID_C, ID_B, \mathcal{L}_B, n_B, \nu, \mathbf{SIG}_{K_B^{-1}}\}_{\mathbf{C}_{BC}},$$

where ν is copied from the M-NDP request. As in D-NDP, B also computes a shared key K_{BA} based on its private key K_B^{-1} and ID_A, by which B further derives the session spread code $\mathbf{C}_{BA} = h_{K_{BA}}(n_B \otimes n_A)$. Node B proceeds to repeatedly send a HELLO message $\{\mathsf{HELLO}, ID_B\}_{\mathbf{C}_{BA}}$ for a duration of τ_h, where τ_h is the longest transmission delay for the M-NDP response to traverse ν hops. In addition, B checks whether the number of hops that the M-NDP request has traversed is equal to ν; if not, B further forwards a modified M-NDP request to all the nodes in $\mathcal{L}_B - \mathcal{L}_A \cup \mathcal{L}_C$, i.e., adding its node ID, logical neighbor list and signature.

In general, when receiving an M-NDP request, every node does the following: verify the ID-based signatures of the sender and all previous nodes; check each node's logical neighbor list to see whether there is a legitimate path between the source and itself; derive the secret key and session spread code uniquely shared with the source and start sending the HELLO message spread with the derived session code; send a modified M-NDP request by adding its own ID and logical neighbor list to the nodes not appearing in the logical neighbor lists of the received request, if the number of hops that the request has traversed is less than ν. The request is dropped if either of the first two steps fails.

On receiving the M-NDP response, C verifies $\mathbf{SIG}_{K_B^{-1}}$ using ID_B as the public key. If the signature is correct, C forwards a modified M-NDP response to A as

$$\{ID_A, ID_C, ID_B, \mathcal{L}_B, n_B, \nu, \mathbf{SIG}_{K_B^{-1}}, \mathcal{L}_C, \mathbf{SIG}_{K_C^{-1}}\}_{\mathbf{C}_{CA}}.$$

In general, the M-NDP response is processed by each intermediate node in a similar way as M-NDP request does, i.e., each node verifies the previous signatures and adds its own ID, logical neighbor list and signature.

Upon receiving the response, node A first verifies $\mathbf{SIG}_{K_C^{-1}}$ and $\mathbf{SIG}_{K_B^{-1}}$ using ID_C and ID_B as the public keys, respectively. If both signatures are correct, A further checks whether $C \in \mathcal{L}_B$, i.e., whether there is a legitimate path between the destination and itself. If so, A uses its private key K_A^{-1} and ID_B to compute the shared key $K_{AB} = K_{BA}$ whereby to derive the session spread code $\mathbf{C}_{AB} = h_{K_{AB}}(n_A \otimes n_B)$ which is equal to $\mathbf{C}_{BA}$. It then starts to monitor $\mathbf{C}_{AB}$ in real time.

If A and B are indeed physical neighbors, then A can receive the HELLO message from B spread with $\mathbf{C}_{BA}$. If so, A accepts B as its authenticated logical neighbor and returns a CONFIRM message spread with $\mathbf{C}_{AB}$. Once receiving the CONFIRM message, node B accepts A as its authenticated logical neighbor.

Using digital signatures in M-NDP is necessary to prevent the DoS attack in which compromised nodes forge arbitrary many M-NDP requests. Consider Fig. 2 as an example. Assume that node C is compromised and sends B a forged M-NDP request claimed to be initiated from source X. Since C does not have the correct private key of X to generate its correct signature on the M-NDP request, B can immediately detect this forged request in the first step and knows that C is compromised. Such immediate detection is unlikely without digital signatures. It is worth noting that the once daunting digital signature operations are growingly trivial on even resource-constrained mobile devices.

Different from D-NDP, M-NDP may incur false positives, which means that some nodes that are not physical neighbors may falsely discover each other, e.g., A may discover nodes G and H in Fig. 2. To address this limitation, we further present a location-aware multi-hop neighbor discovery protocol (LAM-NDP) in the next subsection.

D. LAM-NDP: Location-Aware Multi-Hop Neighbor Discovery Protocol

LAM-NDP is designed to eliminate the false positives incurred by M-NDP by exploring the location information of

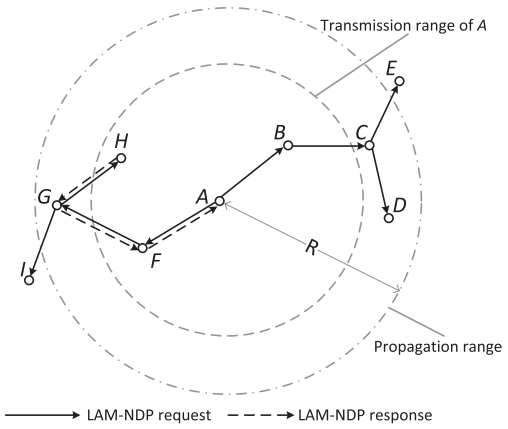

Fig. 3. Illustration of LAM-NDP.

MANET nodes. For this purpose, LAM-NDP requires every MANET node to be capable of localizing itself via GPS signals or other localization techniques such as [32], which are becoming pervasive in modern mobile devices. Different from M-NDP, each LAM-NDP initiator includes its position in its LAM-NDP request and specifies the propagation range within which the request can be forwarded, and each node replies to the LAM-NDP request only if it is within the transmission range of the LAM-NDP initiator.

We illustrate the LAM-NDP operations with the scenario in Fig. 3 as an example. As in M-NDP, all nodes need to periodically initiate the LAM-NDP process at some random time point of its own choice. Assume that node A initiates the LAM-NDP process prior to node J. Recall that $\mathcal{L}_A$ denotes the set of logical neighbors of A, where $\mathcal{L}_A = \{B, F\}$. Node A unicasts an LAM-NDP request to each node in $\mathcal{L}_A$. For example, the request sent to F is

$$A \rightarrow F : \{ID_A, \mathcal{L}_A, n_A, l_A, \mathsf{R}, \mathbf{SIG}_{K_A^{-1}}\}_{\mathbf{C}_{AF}},$$

where l_A is node A's location, R is the *propagation range* chosen by A determining the maximum distance within which the request can be forwarded, and n_A and $\mathbf{SIG}_{K_A^{-1}}$ have the same meanings as in M-NDP request.

After receiving the LAM-NDP request, node F first verifies that it is within the propagation range of node A based on l_A, R and its own location. If so, it then verifies the signature $\mathbf{SIG}_{K_A^{-1}}$ using ID_A as the public key [25]. If the signature verification succeeds, then for each node in $\mathcal{L}_F \setminus \mathcal{L}_A$, say G, F unicasts a modified request

$$F \rightarrow G : \{ID_A, \mathcal{L}_A, n_A, \mathsf{R}, \mathbf{SIG}_{K_A^{-1}}, ID_F, \mathcal{L}_F, \mathbf{SIG}_{K_F^{-1}}\}_{\mathbf{C}_{FG}}.$$

On receiving the LAM-NDP request from F, node G first checks whether it is within the propagation or transmission range of node A based on l_A, R and its own location. If it is outside of the both ranges, the LAM-NDP request is dropped. Otherwise, G verifies the signatures $\mathbf{SIG}_{K_A^{-1}}$ and $\mathbf{SIG}_{K_F^{-1}}$ using ID_A and ID_F as public keys, respectively [25]. Since node G is within the propagation range r of the request, it unicasts a modified request to every node in $\mathcal{L}_G \setminus (\mathcal{L}_A \bigcup \mathcal{L}_F) = \{H, I\}$. For example, the request sent to node H is

$$G \rightarrow H : \{ID_A, \mathcal{L}_A, n_A, \mathsf{R}, \mathbf{SIG}_{K_A^{-1}}, ID_F, \mathcal{L}_F, \mathbf{SIG}_{K_F^{-1}}, ID_G, \mathcal{L}_G, \mathbf{SIG}_{K_G^{-1}}\}_{\mathbf{C}_{GH}}.$$

Different from M-NDP, since node G is not within A's transmission range, it will not return an LAM-NDP response to A via F.

Once node H receiving the LAM-NDP request, it knows that it is within A's transmission range. H then verifies all three signatures using the corresponding node IDs as public keys and checks all the logical neighbor lists in the request to see if there is a legitimate path connecting A and itself. If all the verifications succeed, H returns the following LAM-NDP response to G,

$$H \rightarrow G : \{ID_A, l_A, ID_F, ID_G, ID_H, \mathcal{L}_H, l_h, n_H, \mathsf{R}, \mathbf{SIG}_{K_H^{-1}}, \mathcal{L}_G, \mathbf{SIG}_{K_G^{-1}}, \mathcal{L}_F, \mathbf{SIG}_{K_F^{-1}}, \}_{\mathbf{C}_{HG}},$$

where l_h is H's location, n_H is a random nonce, and R is copied from the LAM-NDP request. As in M-NDP, H also computes a shared key K_{HA} based on its private key K_H^{-1} and ID_A, by which H further derives the session spread code $\mathbf{C}_{HA} = h_{K_{HA}}(n_H \otimes n_A)$. Node H proceeds to repeatedly send a HELLO message $\{\mathsf{HELLO}, ID_H\}_{\mathbf{C}_{HA}}$ for a duration of τ_h, where τ_h is the longest transmission delay for the LAM-NDP response to traverse three hops, i.e., the number of hops the LAM-NDP request has traversed.

In general, when receiving an (modified) LAM-NDP request, every node does the following:

1) Check if it is within the propagation or transmission range of the LAM-NDP initiator. If it is within neither range, the LAM-NDP request is dropped.
2) Verify every ID-based signature in the LAM-NDP request.
3) Check each intermediate node's logical neighbor list to see whether there is a legitimate path between the LAM-NDP initiator and itself.
4) If it is within the transmission range of the LAM-NDP initiator, derive the secret key and session spread code uniquely shared with the LAM-NDP initiator and start sending a HELLO message spread with the session code.
5) If it is within the propagation range of the LAM-NDP request, unicast a modified LAM-NDP request by adding its own ID and logical neighbor list to the nodes not appearing in the logical neighbor lists of the received LAM-NDP request.

The LAM-NDP response is processed by each intermediate node in a similar way as the M-NDP, i.e., each node verifies the previous signatures and adds its own ID, logical neighbor list and signature.

On receiving the LAM-NDP response, A verifies each signature using the corresponding node ID as the public key. If all the signatures are correct, A further checks if there is a legitimate path between H and itself. If so, A uses its private key K_A^{-1} and

ID_H to compute the shared key $K_{AH} = K_{HA}$ whereby to derive the session spread code $\mathbf{C}_{AH} = h_{K_{AH}}(n_A \otimes n_H)$ which is equal to $\mathbf{C}_{HA}$. It then starts to monitor $\mathbf{C}_{AH}$ in real time.

Since A and H are indeed physical neighbors, A can receive the HELLO message from H spread with $\mathbf{C}_{HA}$. If so, A accepts H as its authenticated logical neighbor and returns a CONFIRM message spread with $\mathbf{C}_{AH}$. Once receiving the CONFIRM message, node H accepts A as its authenticated logical neighbor.

E. Discussion

Here we discuss the applicability of LAM-NDP to heterogeneous MANET and some attacks besides jamming related to JR-SND with possible solutions.

1) Applicability of LAM-NDP to Heterogeneous MANET: LAM-NDP can be easily adapted for heterogeneous MANETs where only a fraction of the MANET nodes have localization capability. Specifically, an LAM-NDP initiator can include in an LAM-NDP request both R, i.e., the propagation range within which the request can be forwarded, and ν, i.e., the maximum number of hops the request can traverse. Upon receiving the LAM-NDP request, a node with localization capability will forward the request if it is within the propagation range of the LAM-NDP initiator, whereas a node without localization capability can ignore the propagation range and forwards the request if the number of hops that the request has traversed is less than ν. Such adaptation can be viewed as a combination of M-NDP and LAM-NDP, and the resulting neighbor-discovery latency and the number of false positives thus lie between those of M-NDP and LAM-NDP.

2) Resilience to Denial of Service Attacks: As mentioned earlier, existing anti-jamming solutions such as [4]–[10] all depend on some publicly known communication strategies such as public spread-code sets. If they were adopted for secure neighbor discovery in MANETs, $\mathcal{J}$ would be able to use such public knowledge to keep injecting fake neighbor-discovery requests, thus leading to a special Denial-of-Service (DoS) attack in which all nodes are forced to perform endless verifications of neighbor-discovery requests.

In contrast, JR-SND constrains the impact of this DoS attack to the number of secret spread codes compromised by $\mathcal{J}$. Compromised spread codes can also be revoked in many ways so that non-compromised nodes will not use them for spreading/de-spreading messages. For example, a simple yet effective method is to let each node A maintain a counter for each secret code $\mathbf{C}_x$ it has. Whenever A receives an invalid neighbor-discovery request spread with $\mathbf{C}_x$ (e.g., the signature is incorrect), it increases the corresponding counter by one. Once the counter for $\mathbf{C}_x$ exceeds some predefined threshold γ, which indicates that $\mathbf{C}_x$ is compromised with high probability, A locally revokes $\mathbf{C}_x$ by removing it from its spread-code set. Consequently, subsequent messages spread with $\mathbf{C}_x$ will not be received by node A. Recall our random spread-code pre-distribution method for D-NDP and M-NDP in which each code is shared by at most l nodes. With our defense in place, $\mathcal{J}$ can use a compromised code to launch the DoS attack on other $l-1$ non-compromised nodes with the same code for at most $(l-1)\gamma$ times instead of arbitrary many.

3) Node Replication Attacks: Assume that $\mathcal{J}$ has compromised some node X. It can launch another more subtle attack in which it uses X's information to repeatedly perform seemingly legitimate neighbor discovery with all the other nodes that share some common spread codes with X. This attack can be viewed as node X being replicated throughout the network and is difficult to defend against.

One possible solution is to let some witness nodes keep track of each node's encounter history, and each node checks with a new plausible neighbor's witness nodes before accepting it even if the JR-SND operations have succeeded. For example, suppose that A and B perform neighbor discovery. After a successful execution of the JR-SND scheme, they both need to check with some witnesses of the other. If A is found to have encountered many nodes at distributed locations during a short time window, its witness nodes can raise an alarm to inform B and other nodes in the network. It is worth noting that similar ideas have been used in detecting node replication attack in sensor networks [33] and blackhole attacks in delay-tolerant networks [34]. We, however, stress that how to choose witness nodes in MANETs is still an open challenge. Since all existing neighbor-discovery schemes such as [2], [27], [28], [35] designed for MANETs are vulnerable to this attack, this issue deserves further investigation.

4) Repeater Jamming Attacks: In [36], Hang *et al.* demonstrated that repeater jamming is more effective than random jamming. This attack can be viewed as a special kind of reactive jamming. It is, however, not detrimental and can be solved by error-correcting coding. Alternatively, we can replace each spread code with a code sequence as in [9]. Further investigation on this issue is left as future work.

VI. Performance Evaluation

In this section, we evaluate JR-SND via both theoretical analysis and simulation studies.

A. Performance Analysis

1) Analysis of the Code Pre-Distribution Scheme: We first analyze the proposed spread code pre-distribution scheme. For simplicity, we assume that n can be divided by l. It can be easily seen that any two nodes are assigned the same spread code at each round with probability $\frac{l-1}{n-1}$. Since the operations in each round are independent from those of others, the probability that any two nodes share x spread codes after m rounds is given by

$$\mathsf{Pr}[x] = \binom{m}{x}\left(\frac{l-1}{n-1}\right)^x\left(\frac{n-l}{n-1}\right)^{m-x}. \tag{1}$$

In addition, assume that $\mathcal{J}$ has compromised q nodes. Every spread code in $\mathbb{C}$ is compromised with probability

$$\alpha = 1 - \frac{\binom{n-l}{q}}{\binom{n}{q}}. \tag{2}$$

The expected number of compromised codes is thus $s\alpha$.

2) Analysis of D-NDP: We have the following theorem regarding P_d, the probability that two physical neighbors can discover each other via D-NDP.

Theorem 1: Assuming that the adversary has compromised q nodes, two physical neighbors can discover each other via D-NDP with probability

$$P^- \leq P_{\rm d} \leq P^+, \tag{3}$$

where

$$P^- = 1 - \sum_{x=0}^{m} \mathsf{Pr}[x]\alpha^x,$$

$$P^+ = 1 - \sum_{x=0}^{m} \mathsf{Pr}[x]\alpha^x(\beta + \beta' - \beta\beta')^x, \tag{4}$$

$\mathsf{Pr}[x]$ is given in Eq. (1), $\alpha = 1 - \binom{n-l}{q} \Big/ \binom{n}{q}$ as given in Eq. (2), $c = s\alpha$, $\beta = \min\left\{\frac{z(1+\mu)}{c\mu}, 1\right\}$, and $\beta' = \min\left(\frac{3z(1+\mu)}{c\mu}, 1\right)$.

We give the proof of Theorem 1 in Appendix VII-A. The following theorem is about the average neighbor-discovery latency T of D-NDP.

Theorem 2: Two physical neighbors can discover each other via D-NDP with an average latency

$$\mathrm{T} \approx \frac{\rho m(3m+4)N^2 l_h}{2} + \frac{2Nl_f}{R} + 2t_{\rm key}, \tag{5}$$

where $l_h = (1+\mu)(l_t + l_{\rm id})$, $l_f = (1+\mu)(l_{\rm id} + l_n + l_{\rm mac})$, l_n and $l_{\rm mac}$ are the lengths of nonce and MAC, respectively, and $t_{\rm key}$ is the time needed to compute a shared key.

We give the proof of Theorem 1 in Appendix VII-B.

3) Analysis of M-NDP: We have the following theorem regarding the average neighbor-discovery latency T of M-NDP.

Theorem 3: Two physical neighbors connected by a ν-hop jamming-resilient path can discover each other via M-NDP with an average latency

$$\mathrm{T} = \mathrm{T}_\nu + 2\nu(\nu+1)t_{\rm ver} + 2\nu t_{\rm sig}, \tag{6}$$

where

$$\mathrm{T}_\nu = \frac{N}{R}\left(\frac{3\nu(\nu+1)}{2}\left((g+1)l_{\rm id} + 2l_{\rm sig}\right) + 2\nu(l_n + l_\nu)\right), \tag{7}$$

and g is the average number of logical neighbors each node has.

We give the proof of Theorem 3 in Appendix VII-C. In our preliminary work [1], we have analyzed the neighbor discovery probability of M-NDP for a special case when $\nu = 2$, which is omitted here. We will evaluate the neighbor discovery probability of M-NDP using simulation. Since the number of hops an LAM-NDP request traverses is a random variable, we will also use simulations to evaluate the performance of LAM-NDP.

B. Simulation Results

In this section, we use simulation results to evaluate the proposed schemes with regard to the neighbor-discovery probability P_d and latency T. We simulate 2000 MANET nodes in a $5000 \times 5000\ \mathrm{m}^2$ field, each with a transmission range of 250 m. Table I summarizes most default evaluation parameters unless specified otherwise, in which the cryptographic parameters are adopted from [25]. For our purpose, most of the simulation code is written in C++. Each measurement is the average over 100 simulation runs, each with a different random seed.

TABLE I
DEFAULT EVALUATION PARAMETERS

Para.	Val.	Para.	Val.	Para.	Val.
n	2000	m	100	l	40
q	100	N	512	R	22Mbps
ρ	10^{-11}s/bit	μ	1	ν	2
l_t	5	$l_{\rm id}$	16	l_n	20
l_f	160	l_ν	4	$l_{\rm sig}$	672
$t_{\rm key}$	11ms	$t_{\rm sig}$	5.7ms	$t_{\rm ver}$	35.5ms

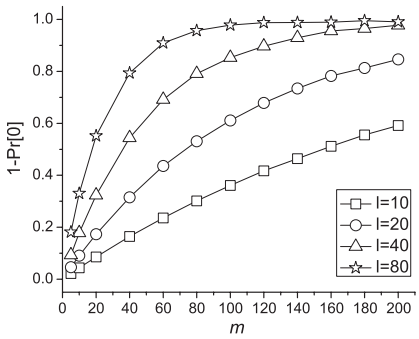

Fig. 4. The prob. that two nodes share at least one code.

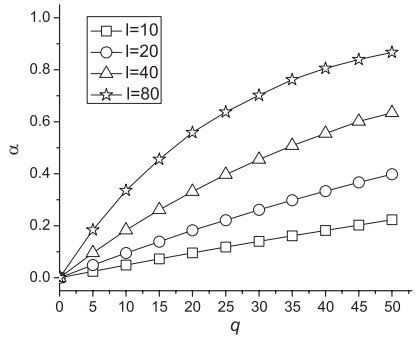

Fig. 5. The prob. of a code being compromised.

1) Evaluation of Code Pre-Distribution Scheme: Fig. 4 shows the probability that two nodes share at least one code, i.e., $1 - \mathsf{Pr}[0]$, varying with m (the number of codes preloaded to each node) and l (the number of nodes sharing the same code). We can see that the larger m and l, the higher the probability of two nodes sharing at least one code, and vice versa. This is anticipated, as the probability of two nodes sharing at least one code increases as the number of codes preloaded to each node and the number of nodes sharing the same code increase.

Fig. 5 shows the probability of each code being compromised (i.e., α in Eq. (2)) varying with l and q (the number of compromised nodes). As we can see, α increases as l or q increases, and vice versa. The reason is that the more nodes

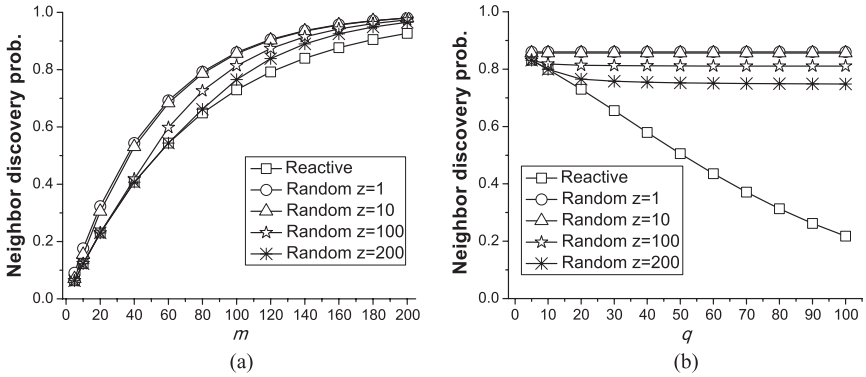

Fig. 6. D-NDP under reactive and random jamming attacks. (a) The impact of m. (b) The impact of q.

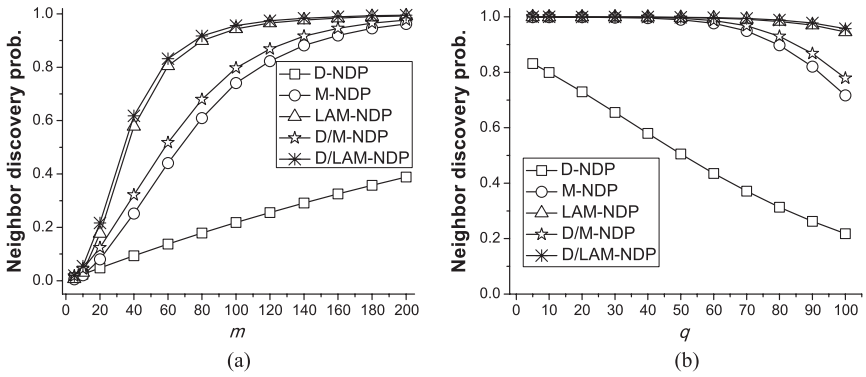

Fig. 7. Comparison of the neighbor discovery probabilities of different protocols. (a) Impact of m. (b) Impact of q.

sharing the same code, the fewer codes in total, and the higher percentage of codes are compromised for the same number of compromised nodes. Moreover, the more compromised nodes, the higher the probability of each code being compromised, which is anticipated. We will show later that this effect does not have much impact on the neighbor discovery probability of D-NDP, as one non-compromised code shared between two neighboring nodes suffices for successful neighbor discovery.

2) Comparison of Reactive and Random Jamming: Fig. 6 compares the neighbor discovery probability of D-NDP under random and reactive jamming attacks. We can see from Fig. 6(a) that the P_{d} of D-NDP increases as m increases in all five cases. The reason is that the more spread codes preloaded to each node, the higher the probability of two nodes sharing at least one non-compromised code. Moreover, the neighbor discovery probability of D-NDP under random jamming attack decreases as z (i.e., the number of concurrent jamming signals) increases, and is always higher than that under reactive jamming attack, which is expected and also observed in Fig. 6(b). In addition, we can see from Fig. 6(b) that the P_{d} of D-NDP decreases as the number of compromised nodes increases, which is of no surprise. Since reactive jamming is always more effective than random jamming in jamming neighbor discovery, we will only consider reactive jamming hereafter.

3) Comparison of Neighbor Discovery Protocols: Fig. 7(a) compares the P_{d}s of D-NDP, M-NDP, LAM-NDP, D/M-NDP (i.e., the combination of D-NDP and M-NDP), and D/LAM-NDP (i.e., the combination of D-NDP and LAM-NDP) with m varying from 5 to 200, where ν is set to 2 for M-DNP and R is set to the transmission range of each node for LAM-NDP. We can see that the P_{d}s of D-NDP, M-NDP and LAM-NDP all increase as m increases, as the M-NDP and LAM-NDP both rely on D-NDP, of which P_{d} increases as m increases. Moreover, under the default setting, LAM-NDP has the highest neighbor discovery probability, followed by M-NDP and D-NDP. In addition, the P_{d}s of D/M-NDP is always higher than that of M-NDP alone, as two nodes can discover each other either via D-NDP or M-NDP. For the same reason, the P_{d} of D/LAM-NDP is always higher than that of LAM-NDP alone.

Fig. 7(b) compares the P_{d}s of D-NDP, M-NDP, LAM-NDP, D/M-NDP, and D/LAM-NDP with q varying from 5 to 100. We can see that the P_{d}s decrease as q increases in all five cases, among which the P_{d} of D-NDP drops fastest, e.g., from 0.83 to 0.22 as q increases from 5 to 100. In contrast, the P_{d}s of M-NDP, LAM-NDP, D/M-NDP, and D/LAM-NDP decrease much slower. For example, although the P_{d} of D-NDP is only 0.22 when 100 nodes are compromised, the P_{d}s of M-NDP, LAM-NDP, D/M-NDP, and D/LAM-NDP are 0.71, 0.95, 0.78, and 0.96, respectively. Since D-NDP and M-NDP or D-NDP and LAM-NDP are always used jointly in practice, the overall performance is always sufficiently good.

4) Evaluation of M-NDP: Fig. 8(a) shows the P_{d}s of M-NDP and D/M-NDP varying with ν, the number of hops an M-NDP request can traverse, where the P_{d} of D-NDP is

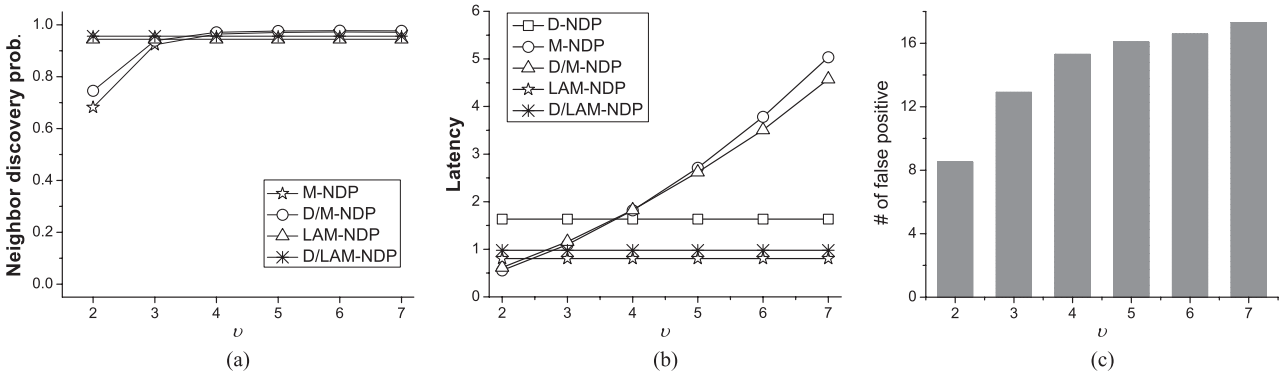

Fig. 8. The impact of ν on the neighbor discovery probability, latency and false positives of M-NDP. (a) Neighbor discovery probability. (b) Neighbor discovery latency. (c) False positives.

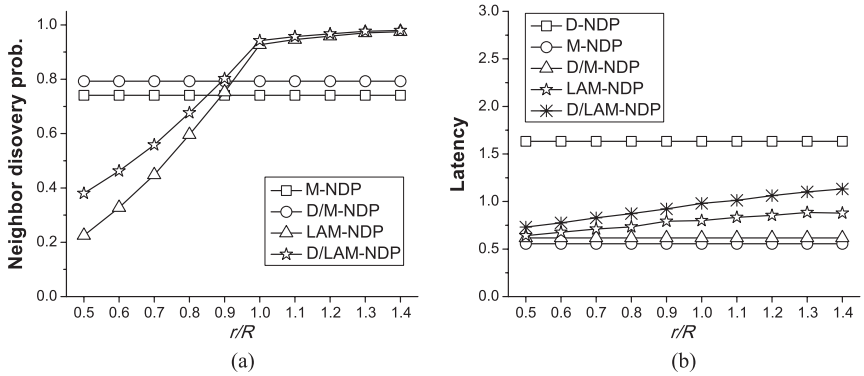

Fig. 9. The impact of propagation range on the neighbor discovery probability and latency of LAM-NDP. (a) Neighbor discovery probability. (b) Neighbor discovery latency.

0.22 corresponding to $q = 100$ as shown in Fig. 6(b) the P_{d}s of LAM-NDP and D/LAM-NDP are plotted for reference only. We can see that the larger ν, the higher the P_{d}s of M-NDP and D/M-NDP, and vice versa. In particular, when $\nu \geq 4$, the M-DNP and D/M-NDP can achieve P_{d}s over 0.96 and 0.97, respectively, which are sufficiently good in most cases. In practice, MANET nodes can dynamically adjust ν to achieve satisfactory neighbor-discovery probabilities.

Fig. 8(b) shows the neighbor discovery latencies of M-NDP and D/M-NDP varying with ν, where the Ts of D-NDP, LAM-NDP and D/LAM-NDP are plotted for reference only. We can see that the T of M-DNP increases quadratically as ν increases, which coincides with Eq. (6) in Theorem 6. In addition, under our default setting, D-NDP has a latency of 1.6 seconds and the latency of D/M-NDP is always between the Ts of D-DNP and M-NDP, as two nodes can discover each other either via D-NDP or M-NDP. Combining Figs. 8(a) and 6(b), we can see that when $\nu \geq 4$, D/M-NDP can achieve P_{d} over 0.97 with a latency of less than two seconds, which is acceptable in most cases. Moreover, the neighbor discovery probabilities of LAM-NDP and D/LAM-NDP are close to those of M-NDP and D/M-NDP, while their latencies are lower than that of M-NDP for $\nu = 2$.

Fig. 8(c) shows the average number of false positives incurred by M-NDP. We can see that the number of false positives increases as ν increases. The reason is that the larger ν, the more nodes within each node's ν-hop neighborhood that can receive the M-NDP request, and the larger fraction of these nodes are outside of the M-NDP initiator's transmission range. In contrast, LAM-NDP incurs no false positive by exploring the location information of each node.

5) Evaluation of LAM-NDP: Fig. 9(a) shows the P_{d}s of LAM-NDP and D/LAM-NDP varying with propagation range R within which an LAM-NDP request can traverse, where the P_{d} of D-NDP is 0.2 and those of M-NDP and D/M-NDP are plotted for reference only. We can see that the P_{d} of LAM-NDP increases as R increases. This is anticipated because the larger the R, the more nodes can receive an LAM-NDP request, the more neighbors can be discovered via LAM-NDP, and vice versa. We can also see from Fig. 9(a) that the P_{d} of LAM-NDP increases much faster when R is smaller than node's transmission range than it does when R is smaller than each node's transmission range. The reason is that when R is smaller than node's transmission range, all the nodes that receive a LAM-NDP request are physical neighbors of the LAM-NDP initiator and can be discovered. In contrast, as R exceeds node's transmission range, the number of nodes that receive an LAM-NDP request while outside of the LAM-NDP initiator's transmission range increases, so the increase in P_{d} becomes slower.

Fig. 9(b) shows the Ts of LAM-NDP and D/LAM-NDP with R varying between 0.5 to 1.4 times of transmission range of each node, where the Ts of M-NDP and D/M-NDP are plotted for reference only. We can see that the T of LAM-NDP increases as R increases, which is anticipated. In addition,

similar to the T of D/M-NDP, the T of D/LAM-NDP is always between those of D-DNP and LAM-NDP. Moreover, for the same P_{d}, the T of LAM-NDP is smaller than that of M-NDP, which clearly demonstrates the advantage of LAM-NDP over M-NDP.

6) Other Factors on JR-SND: We have also studied other potential factors on the performance of JR-SND, which have limited impact and thus are not shown here due to space constraints. For example, we have simulated the impact of node mobility using QualNet 4.5.1. our simulation results show that two nodes with average velocities 25 m/s encounter each other for a duration larger than 5 seconds with probability higher than 0.96, which indicates that JR-SND can satisfy the stringing timing requirement in MANETs with very high mobility. In addition, our simulation results show that the ECC coding factor μ has some impact on the neighbor-discovery latency of D-NDP, which is easy to understand.

7) Summary: We summarize the simulation results as follows.

- D-NDP can enable two neighbors to directly discover each other with high probability and low latency.
- M-NDP is built upon D-NDP and improves the neighbor-discovery probability by letting two neighbors discover each other via a multi-hop path with moderate latency. It is suitable for MANETs with moderate node density.
- LAM-NDP is built upon D-NDP and explores the location information of each node to simultaneously improve the neighbor-discovery probability and latency of M-NDP. It is suitable for MANETs with moderate node density.
- By combining D-NDP and M-NDP or LAM-NDP, two nodes can discover each other with overwhelming probability and low latency.

VII. Conclusion

In this paper, we propose JR-SND, a novel solution based on DSSS and spread-code pre-distribution to achieve jamming-resilient neighbor discovery in MANETs. JR-SND can enable two neighboring nodes to successfully discover each other with overwhelming probability despite omnipresent jammers. The efficacy and efficiency of our schemes are confirmed by detailed theoretical analysis and simulation results.

Appendix

A. Proof of Theorem 1

Proof: Assuming that $\mathcal{J}$ has compromised q nodes, the expected number of compromised spread codes is thus $c = s\alpha = s\left(1 - \binom{n-l}{q} \Big/ \binom{n}{q}\right)$ as analyzed in Section VI-A1. If the spread-code length N is sufficiently long and the spread-code pool size $s \ll 2^N$, the probability that $\mathcal{J}$ successfully jams a targeted transmission with a randomly guessed code of N bits is comparatively negligible to that of $\mathcal{J}$ using compromised codes. We thus assume that $\mathcal{J}$ will only attempt jamming using compromised codes.

We consider random and reactive jamming in this paper, as stated in Section IV-B. For any ongoing message transmission, random jamming can succeed if the spread code in use is compromised and also happens to be chosen by $\mathcal{J}$ to jam at least $\mu/(1+\mu)$ of the message. In contrast, reactive jamming can succeed if the spreading code being used is compromised and can be identified by $\mathcal{J}$ before $1/(1+\mu)$ of the message is transmitted. Apparently, reactive jamming has higher requirement on $\mathcal{J}$'s capability than random jamming but is also more effective. Consequently, the neighbor-discovery probabilities under random and reactive jamming can be considered as the upper bound (denoted by P^+) and the lower bound (denoted by P^-), respectively.

We first consider random jamming. Assume that nodes A and B share x spreading codes. Denote by $P^+(x)$ the probability that two nodes can successfully discover each other given that they share x spreading codes. Obviously, we have $P^+(0) = 0$.

Now let us consider the case of $x = 1$. Assume that the shared spreading code is compromised, which happens with probability $\alpha = 1 - \binom{n-l}{q} \Big/ \binom{n}{q}$ as given in Eq. (2). Recall that $\mathcal{J}$ can emit at most z jamming signals on a targeted transmission. Since $\mathcal{J}$ must use a code for at least $\mu/(1+\mu)$ of the message transmission time, it can try at most $z(1+\mu)/\mu$ distinct codes randomly chosen from the c compromised codes during the message transmission. To make the analysis tractable, we make the most pessimistic assumption that $\mathcal{J}$ can distinguish the four D-NDP messages and applies different jamming strategies to them. The first HELLO message can be jammed if $\mathcal{J}$ selects the correct code, which happens with probability

$$\beta = \min\left\{\frac{z(1+\mu)}{\mu c}, 1\right\}. \tag{8}$$

In contrast, the last three messages are not independent from each other, and each is spread with the same single code. The probability of at least one of the last three messages being jammed is

$$\beta' = \min\left\{\frac{3z(1+\mu)}{\mu c}, 1\right\}. \tag{9}$$

So we have

$$\begin{aligned} P^+(1) &= 1 - \alpha + \alpha(1-\beta)(1-\beta') \\ &= 1 - \alpha(\beta + \beta' - \beta\beta'). \end{aligned} \tag{10}$$

Now we consider $x \geq 2$. The D-NDP execution fails if all the x codes are compromised, which happens with probability α^x, and also all the x compromised codes are selected by $\mathcal{J}$ to jam all the x D-NDP sub-sessions, which happens with probability $(\beta + \beta' - \beta\beta')^x$. Therefore, for $x \geq 2$, we have

$$\begin{aligned} P^+(x) &= 1 - \alpha^x + \alpha^x\left(1 - (\beta + \beta' - \beta\beta')^x\right) \\ &= 1 - \alpha^x(\beta + \beta' - \beta\beta')^x. \end{aligned} \tag{11}$$

The probability that the first HELLO message being jammed is obviously lower than β and very difficult to analyze. Assume that the code chosen for the last three messages is

compromised, which occurs with probability α'. Then at least of the last three messages is also jammed with probability β'. We thus have $\hat{\mathrm{P}}^{+}(x) \leq 1-\alpha+\alpha(1-\beta')$ for $x \geq 2$.

Summarizing the above cases, we have

$$P^{+}=\sum_{x=0}^{m} P^{+}(x)\mathsf{Pr}[x]=1-\sum_{x=0}^{m}\mathsf{Pr}[x]\alpha^{x}(\beta+\beta'-\beta\beta')^{x}. \tag{12}$$

Now we consider reactive jamming under which any message spread with a compromised spread code can be jammed. Two nodes can discover each other if they share at least one non-compromised spread code. We thus can compute $P^{-}=1-\sum_{x=0}^{m}\mathsf{Pr}[x]\alpha^{x}$. The actual jamming performance of $\mathcal{J}$ is between random and reactive jamming, i.e., $P^{-} \leq P_{\mathrm{d}} \leq P^{+}$. □

B. Proof of Theorem 2

Proof: Without loss of generality, we assume that A initiates neighbor discovery with B. To ease the analysis, we also assume that whenever A or B starts to transmit a message, the other is processing the previously buffered signal and not buffering the incoming signal since $\lambda \gg 1$ in practice.

We first consider the time need by nodes A and B to exchange the first two messages, i.e., identify each other, denoted by T_{i}. We define the following timeline. A starts broadcasting the HELLO message at T_1; B starts buffering at T_2 and starts buffer processing at T_3; B de-spreads the HELLO message and starts transmitting the CONFIRM message at T_4; A starts buffering at T_5, starts buffer processing at T_6, and de-spreads the CONFIRM message at T_7.

Let $t_B^{\mathrm{r}}=T_3-T_1$ be B's residual processing time of the previous buffer, and $t_B^{\mathrm{d}}=T_4-T_3$ be the time for B to de-spread the HELLO message with $\mathbf{C}_i$. Also let $t_A^{\mathrm{r}}=T_6-T_4$ be A's residual processing time of the previous buffer, and $t_A^{\mathrm{d}}=T_7-T_6$ the time for A to de-spread the CONFIRM message using $\mathbf{C}_i$.

Since nodes are not synchronized before discovering each other, t_B^{r} and t_A^{r} are two independent random variables uniformly distributed in $[0, t_p]$. Moreover, since A transmits the HELLO message spread with its m spread codes sequentially, B may find the message spread with $\mathbf{C}_i$ at any buffer place. So t_B^{d} is also a random variable uniformly distributed in $[0, t_p]$. In contrast, A can de-spread $\{\mathsf{CONFIRM}, ID_B\}_{\mathbf{C}_i}$ after processing at most the first N chip positions, so t_A^{d} is a random variable uniformly distributed in $[0, \lambda t_h]$. Let $\mathsf{E}[\cdot]$ denote expectation. We have

$$\mathsf{E}[\mathrm{T_i}] \approx \mathsf{E}\left[t_B^{\mathrm{r}}\right]+\mathsf{E}\left[t_B^{\mathrm{d}}\right]+\mathsf{E}\left[t_A^{\mathrm{r}}\right]+\mathsf{E}\left[t_A^{\mathrm{d}}\right]=\frac{t_p}{2}+\frac{t_p}{2}+\frac{t_p}{2}+\frac{\lambda t_h}{2}$$

$$=\frac{3\lambda(m+1)t_h}{2}+\frac{\lambda t_h}{2}=\frac{\rho m(3m+4)N^2 l_h}{2}. \tag{13}$$

Now we consider the time for A and B to authenticate each other, denoted by $\mathrm{T_a}$. The last two messages during mutual authentication involve negligible de-spreading delay but are much longer than the first two, so we need to consider the related transmission delays which are both $N(1+\mu)(l_{\mathrm{id}}+l_n+l_{\mathrm{mac}})/R$. Each node also needs to computing the shared key. We then have

$$\mathsf{E}[\mathrm{T_a}]=\frac{2N(1+\mu)(l_{\mathrm{id}}+l_n+l_{\mathrm{mac}})}{R}+2t_{\mathrm{key}}. \tag{14}$$

Finally, we have $\mathrm{T}=\mathsf{E}[\mathrm{T_i}]+\mathsf{E}[\mathrm{T_a}]$ as shown in Eq. (5). □

C. Proof of Theorem 3

Proof: Assume that two physical neighbors A and B are connected by a ν-hop jamming resilient path. Consider the M-NDP request from A to B. The expected size of the M-NDP request on the ith hop can be computed as $l_i=i(g+1)l_{\mathrm{id}}+l_n+l_\nu+il_{\mathrm{sig}}$, where $l_n, l_\nu, l_{\mathrm{sig}}$ are the lengths of the nonce, ν and the signature, respectively. The expected total transmission delay of the M-NDP request across ν hops is then given by

$$\mathrm{T}_{\nu,\mathrm{req}}=\frac{N}{R}\sum_{i=1}^{\nu} l_i=\frac{N}{R}\left(\frac{\nu(\nu+1)}{2}\left((g+1)l_{\mathrm{id}}+l_{\mathrm{sig}}\right)+\nu(l_n+l_\nu)\right).$$

In addition, the ith node on the path need verify i signatures and generate its own signature, which takes time $it_{ver}+t_{sig}$. Here t_{ver} and t_{sig} denote the time needed for one signature verification and one signature generation, respectively. Similarly, the total transmission delay of the M-NDP response from B to A can be computed as

$$\mathrm{T}_{\nu,\mathrm{rsp}}=\frac{N}{R}\left(\frac{\nu(\nu+1)}{2}\left((g+2)l_{\mathrm{id}}+l_{\mathrm{sig}}\right)+\nu(l_n+l_\nu)\right).$$

Let $\mathrm{T}_\nu=\mathrm{T}_{\nu,\mathrm{req}}+\mathrm{T}_{\nu,\mathrm{rsp}}$, we have $\mathrm{T}=\mathrm{T}_\nu+2\nu(\nu+1)t_{\mathrm{ver}}+2\nu t_{\mathrm{sig}}$, where $\mathrm{T}_\nu=\frac{N}{R}\left(\frac{\nu(\nu+1)}{2}((2g+3)l_{\mathrm{id}}+2l_{\mathrm{sig}})+2\nu(l_n+l_\nu)\right)$. □

ACKNOWLEDGMENT

We would also like to thank anonymous reviewers for their constructive comments and helpful advice.

REFERENCES

[1] R. Zhang, Y. Zhang, and X. Huang, "JR-SND: Jamming-resilient secure neighbor discovery in mobile ad-hoc networks," in *Proc. IEEE ICDCS*, Minneapolis, MN, USA, Jun. 2011, pp. 529–538.

[2] P. Papadimitratos *et al.*, "Secure neighborhood discovery: A fundamental element for mobile ad hoc networking," *IEEE Commun. Mag.*, vol. 46, no. 2, pp. 132–139, Feb. 2008.

[3] R. Pickholtz, D. Schilling, and L. Milstein, "Theory of spread-spectrum communications-A tutorial," *IEEE Trans. Commun.*, vol. COM-30, no. 5, pp. 855–884, May 1982.

[4] L. Baird, W. Bahn, M. Collins, C. Carlisle, and C. Butler, "Keyless jam resistance," in *Proc. IEEE Inf. Assurance Security Workshop*, Montreal, CA, USA, Jun. 2007, pp. 143–150.

[5] M. Strasser, C. Popper, S. Capkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *Proc. IEEE S&P*, Berkeley/Oakland, CA, USA, May 2008, pp. 64–78.

[6] M. Strasser, C. Popper, and S. Capkun, "Efficient uncoordinated FHSS anti-jamming communication," in *Proc. ACM MobiHoc*, Apr. 2009, pp. 207–218.

[7] D. Slater, P. Tague, R. Poovendran, and B. J. Matt, "A coding-theoretic approach for efficient message verification over insecure channels," in *Proc. ACM WiSec*, Zurich, Switzerland, Mar. 2009, pp. 151–160.

[8] T. Jin, G. Noubir, and B. Thapa, "Zero pre-shared secret key establishment in the presence of jammers," in *Proc. ACM MobiHoc*, Apr. 2009, pp. 219–228.

[9] C. Popper, M. Strasser, and S. Capkun, "Jamming-resistant broadcast communication without shared keys," in *Proc. USENIX Security*, Aug. 2009, pp. 231–248.

[10] Y. Liu, P. Ning, H. Dai, and A. Liu, "Randomized differential DSSS: Jamming-resistant wireless broadcast communication," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.
[11] A. Liu, P. Ning, H. Dai, Y. Liu, and C. Wang, "Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure," in *Proc. ACSAC*, Austin, TX, USA, Dec. 2010, pp. 367–376.
[12] Q. Wang, P. Xu, K. Ren, and M. Li, "Delay-bounded adaptive UFH-based anti-jamming wireless communication," in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 1413–1421.
[13] Q. Wang, P. Xu, K. Ren, and X.-Y. Li, "Towards optimal adaptive UFH-based anti-jamming wireless communication," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 16–30, Jan. 2012.
[14] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proc. IEEE S&P*, Oakland, CA, USA, May 2003, pp. 197–213.
[15] L. Xiao, H. Dai, and P. Ning, "Jamming-resistant collaborative broadcast using uncoordinated frequency hopping," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 297–309, Feb. 2012.
[16] C. Li, H. Dai, L. Xiao, and P. Ning, "Communication efficiency of anti-jamming broadcast in large-scale multi-channel wireless networks," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5281–5292, Oct. 2012.
[17] H. Wang, L. Zhang, T. Li, and J. Tugnait, "Spectrally efficient jamming mitigation based on code-controlled frequency hopping," *IEEE Trans. Wireless Commun.*, vol. 10, no. 3, pp. 728–732, Mar. 2011.
[18] S. Liu, L. Lazos, and M. Krunz, "Thwarting control-channel jamming attacks from inside jammers," *IEEE Trans. Mobile Comput.*, vol. 11, no. 9, pp. 1545–1558, Sep. 2012.
[19] H. Liu, Y. Chen, M. C. Chuah, and J. Yang, "Towards self-healing smart grid via intelligent local controller switching under jamming," in *Proc. IEEE CNS*, Washington, DC, USA, Oct. 2013, pp. 127–135.
[20] R. Nikjah and N. C. Beaulieu, "On antijamming in general CDMA systems-Part I: Multiuser capacity analysis," *IEEE Trans. Wireless Commun.*, vol. 7, no. 5, pp. 1646–1655, May 2008.
[21] R. Nikjah and N. C. Beaulieu, "On antijamming in general CDMA systems-Part II: Antijamming performance of coded multicarrier frequency-hopping spread spectrum systems," *IEEE Trans. Wireless Commun.*, vol. 7, no. 3, pp. 888–897, Mar. 2008.
[22] Z. Lu, W. Wang, and C. Wang, "Hiding traffic with camouflage: Minimizing message delay in the smart grid under jamming," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 3066–3070.
[23] Q. Yan *et al.*, "MIMO-based jamming resilient communication in wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 2697–2706.
[24] E. Sousa and J. Silvester, "Optimum transmission ranges in a direct-sequence spread-spectrum multihop packet radio network," *IEEE J. Sel. Areas Commun.*, vol. 8, no. 5, pp. 762–771, Jun. 1990.
[25] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing mobile ad hoc networks with certificateless public keys," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 386–399, Oct.–Dec. 2006.
[26] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. CRYPTO*, Santa Barbara, CA, USA, Aug. 2001, pp. 213–229.
[27] M. Poturalski, P. Papadimitratos, and J.-P. Hubaux, "Secure neighbor discovery in wireless networks: Formal investigation of possibility," in *Proc. ACM ASIACCS*, Tokyo, Japan, Mar. 2008, pp. 189–200.
[28] M. Poturalski, P. Papadimitratos, and J.-P. Hubaux, "Towards provable secure neighbor discovery in wireless networks," in *Proc. 6th ACM Workshop Formal Methods Security Eng.*, Alexandria, VA, USA, Oct. 2008, pp. 31–42.
[29] R. Zhang and Y. Zhang, "Wormhole-resilient neighbor discovery in underwater acoustic networks," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.
[30] R. Stoleru, H. Wu, and H. Chenji, "Secure neighbor discovery and wormhole localization in mobile ad hoc networks," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1179–1190, 2012.
[31] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.
[32] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Pers. Commun. Mag.*, vol. 7, no. 5, pp. 28–34, Oct. 2000.
[33] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *Proc. IEEE S&P*, Oakland, CA, USA, May 2005, pp. 49–63.
[34] F. Li, J. Wu, and A. Srinivasan, "Thwarting blackhole attacks in disruption-tolerant networks using encounter tickets," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2428–2436.
[35] D. Liu, "Protecting neighbor discovery against node compromises in sensor networks," in *Proc. IEEE ICDCS*, Montreal, QC, USA, Oct. 2009, pp. 579–588.
[36] W. Hang, W. Zanji, and G. Jingbo, "Performance of DSSS against repeater jamming," in *Proc. ICECS*, Nice, France, Dec. 2006, pp. 858–861.

Rui Zhang (S'09–M'12) received the B.E. degree in communication engineering and the M.E. degree in communication and information system from Huazhong University of Science and Technology, Wuhan, China, in 2001 and 2005, respectively, and the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2013. From 2005 to 2007, he was a Software Engineer with UTStarcom Shenzhen R&D Center. Since July 2013, he has been an Assistant Professor with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI, USA. His research interests include security and privacy in networked and distributed systems, wireless networking, and mobile computing.

Jingchao Sun (S'14) received the B.E. degree in electronics and information engineering and the M.E. degree in communication and information system from Huazhong University of Science and Technology, Wuhan, China, in 2008 and 2011, respectively. He is currently working toward the Ph.D. degree with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. His primary research interests are network and distributed system security and privacy, wireless networking, and mobile computing.

Yanchao Zhang (S'03–M'06–SM'11) received the B.E. degree in computer science and technology from Nanjing University of Posts and Telecommunications, Nanjing, China, in 1999; the M.E. degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2002; and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2006. From 2006 to 2010, he was an Assistant Professor of electrical and computer engineering with New Jersey Institute of Technology. He is currently an Associate Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. His primary research interests are network and distributed system security, wireless networking, and mobile computing. He is (was) an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY; a Feature Editor of IEEE WIRELESS COMMUNICATIONS; a Guest Editor of IEEE Wireless Communications Special Issue on Security and Privacy in Emerging Wireless Networks in 2010; and a TPC Cochair of Communication and Information System Security Symposium, IEEE GLOBECOM 2010. He was a recipient of the NSF CAREER Award in 2009.

Xiaoxia Huang (S'95–M'07) received the B.E. and M.E. degrees in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer Engineering from University of Florida, Gainesville, FL, USA, in 2007. She is currently a Professor with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. Her research interests include cognitive radio networks, wireless communications, and mobile computing.