2011 31st International Conference on Distributed Computing Systems

JR-SND: Jamming-Resilient Secure Neighbor Discovery in Mobile Ad Hoc Networks

Rui Zhang, Yanchao Zhang
School of Electrical, Computer, and Energy Engineering
Arizona State University
Tempe, AZ, USA
{*ruizhang,yczhang*}*@asu.edu*

Xiaoxia Huang
Shenzhen Institutes of Advanced Technology
Chinese Academy of Sciences
Shenzhen, Guangdong, China
xx.huang@sub.siat.ac.cn

***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 sends 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 is, however, impossible to establish before the communicating parties successfully discover each other. While several elegant approaches have been recently proposed to break this circular dependency, 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 pre-distribution. 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.**

***Keywords*-Jamming; secure neighbor discovery; MANET**

I. Introduction

Secure neighbor discovery is fundamental to mobile ad hoc Networks (MANETs) deployed in hostile environments. 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 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 [1], 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.

One may think about two intuitive ways to apply spread-spectrum techniques for jamming-resilient secure neighbor discovery in a MANET. One is to let all nodes share a common spread code. This approach, however, suffers from the single point of failure: the adversary can know the spread code after compromising any node. A much safer alternative is to let each pair of nodes share a unique code so that the spread codes between non-compromised node pairs remain secret no matter how many nodes the adversary has compromised. With this approach, 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 [2]–[10] 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. First, node encounters are unpredictable in MANETs, so each node must be always prepared to accept and validate potential neighbor discovery requests. The existing solutions [2]–[10] 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). Second, 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 [2]–[10]. 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.

In this paper, we propose a novel Jamming-Resilient Secure Neighbor Discovery (JR-SND) scheme for single-authority DSSS-based MANETs. Motivated by random key pre-distribution schemes for sensor networks such as [11], JR-SND requires the MANET authority to generate a pool of spread codes which are kept secret. Prior to network deployment, every node is loaded with a constant number of spread codes drawn from the spread-code pool such that any two nodes have some common codes with certain probability. During the network operation, the adversary may compromise some nodes to know the spread codes they have, but the non-compromised codes will remain secret. With JR-SND in place, two neighboring nodes can conduct anti-jamming secure neighbor discovery by DSSS communications based on their spread codes. They can directly discover each other if they have at least one common spread code unknown to omnipresent jammers; otherwise, they can indirectly discover each other if there is a multi-hop path connecting them, along which every two neighboring nodes have successfully discovered each other. Compared to prior work [2]–[10] 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 properly addressed by existing anti-jamming techniques such as [2]–[10].
- To the best of our knowledge, we are the first to combine DSSS with key pre-distribution [11] for anti-jamming communications in MANETs. We also propose a new random spread-code pre-distribution scheme to enable the fine control of the damage from compromised spread codes.
- We propose a jamming-resilient secure neighbor discovery scheme based on random spread-code pre-distribution, which enables any two neighboring nodes to quickly discover each other with overwhelming probability under severe jamming attacks. The efficacy of our scheme is confirmed by thorough theoretical analysis and simulation results.

II. Related Work

Several schemes have been recently proposed to enable two nodes to establish a secret spread code (or key) under the jamming attack. In their seminal work [3], 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 [4], [5] 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 [6], 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 in our scenario before the sender successfully discovers the receiver.

Anti-jamming broadcast communications have been recently studied in [7]–[10], respectively. Both schemes are based on DSSS and a publicly known spread-code set and thus vulnerable to the DoS attack mentioned earlier.

III. Background on DSSS

In a DSSS system, 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 $\frac{1}{N}\sum_{i=1}^{N} u_i v_i$. In practice, 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 [7].

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 [12] and can be easily realized by hardware. This implies that an incoming message spread using the code under realtime 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 realtime 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 [7], we choose pseudorandom codes for DSSS communications and 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., $N = 512$.

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 parties. To ease our presentation, we assume an approach as in [13] based on Identity-Based Cryptography [14], 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 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 [6], [7] 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 [7], 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 in this paper.

- *Random jammer*: whenever $\mathcal{J}$ detects an ongoing transmission, $\mathcal{J}$ jams it with a random chosen 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 [6], 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 introduce a random spread-code pre-distribution scheme as a nontrivial adaption of existing key pre-distribution schemes [11]. We then present a direct neighbor-discovery protocol (D-NDP) and a multi-hop neighbor-discovery protocol (M-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 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.

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

$$A \to * : \{\text{HELLO}, ID_A\}_{\mathbf{C}_i} ,$$

where HELLO is an 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 [15] to increase the transmission reliability. In particular, assuming that each node ID is of l_{id} bits, the original message is thus of $l_t + l_{id}$ bits. Node A then applies ECC to generate an encoded message of $l_h = (1+\mu)(l_t+l_{id})$ 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 $\mu(l_t+l_{id})$ bits to prevent B from decoding $\{\text{HELLO}, ID_A\}_{\mathbf{C}_i}$.

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 $\{\text{HELLO}, ID_A\}_{\mathbf{C}_i}$ 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 $\{\text{HELLO}, ID_A\}_{\mathbf{C}_i}$.

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 [7]. Specifically, assume that B has buffered f chips of the incoming signal, denoted by $(p_1, \cdots, p_f)$, in which the first complete $\{\text{HELLO}, ID_A\}_{\mathbf{C}_i}$ 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 $1 \le i \le f$.[1] 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 $\{\text{HELLO}, ID_A\}_{\mathbf{C}_i}$. Node B then uses $\mathbf{C}_i$ to de-spread the rest of the message, i.e., compute 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 [12]. 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 [7]. We thus have $\rho \approx 8.3 \times 10^{-12}$

[1]In fact, B only need process the first $f - Nl_h + 1$ chip positions. We make the approximation to simplify the expression.

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 significant 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 $i \geq 1$,[2] 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 $\{\text{HELLO}, ID_A\}_{\mathbf{C}_i}$, so we have $r = \lceil(\lambda+1)(m+1)/m\rceil$.

After de-spreading $\{\text{HELLO}, ID_A\}_{\mathbf{C}_i}$, 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 $\mathbf{C}_i$,

$$B \to A : \{\text{CONFIRM}, ID_B\}_{\mathbf{C}_i} .$$

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 $\mathbf{C}_i$ in real time. 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 with B, node A computes a shared key K_{AB} using its ID-based private key K_A^{-1} and ID_B by the same method in [13]. It then sends to B the following ECC-coded message spread with $\mathbf{C}_i$,

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

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.

[2]Assume each neighbor discovery process happens after t_p.

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 [13]. 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} [13]. Node B proceeds to transmit the following ECC-coded response

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

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

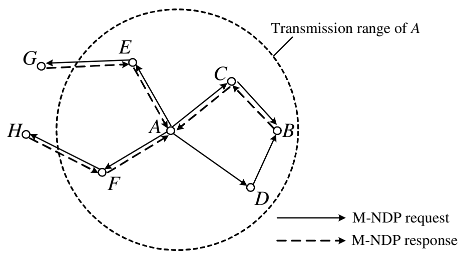


Figure 1. Illustration of M-NDP.

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 via D-NDP or M-NDP.

We illustrate the M-NDP operations with the scenario in Fig. 1 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.

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. 1. Node A unicasts an M-NDP request to each node in $\mathcal{L}_A$. For example, the request sent to C is

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

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 [13]. 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

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

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,

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

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 $\{\text{HELLO}, ID_B\}_{\mathbf{C}_{BA}}$ 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 add 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.

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. 1. The number of false positives is upper-bounded by the number of ν-hop physical neighbors. To eliminate such false positives, each node can include its position in its M-NDP request and only reply to an M-NDP request if the source location is within its transmission range. This method requires each node to have localization capability such as GPS receivers.

D. Resilience to Denial of Service Attacks

As mentioned earlier, existing anti-jamming solutions such as [2]–[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.

VI. Performance Evaluation

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

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

Now we analyze the resilience of the scheme to node compromise attacks. 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 spread codes is thus $s\alpha$. The impact of node compromise attacks on neighbor discovery will be shown in Section VI-B.

2) Analysis of D-NDP: We have the following theorem regarding $\hat{\mathrm{P}}_{\mathrm{D}}$, 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 $\hat{\mathrm{P}}^- \leq \hat{\mathrm{P}}_{\mathrm{D}} \leq \hat{\mathrm{P}}^+$, where $\hat{\mathrm{P}}^- = 1 - \sum_{x=0}^{m} \mathsf{Pr}[x]\alpha^x$, $\hat{\mathrm{P}}^+ = 1 - \sum_{x=0}^{m} \mathsf{Pr}[x]\alpha^x(\beta + \beta' - \beta\beta')^x$, $\mathsf{Pr}[x]$ is given in Eq. (1), $\alpha = 1 - \binom{n-l}{q}/\binom{n}{q}$ as given in Eq. (2), $c = s\alpha$, $\beta = \min\{\frac{z(1+\mu)}{c\mu}, 1\}$, and $\beta' = \min(\frac{3z(1+\mu)}{c\mu}, 1)$.*

Proof: Assuming that $\mathcal{J}$ has compromised q nodes, the expected number of compromised spread codes is thus $c = s\alpha = s(1 - \binom{n-l}{q}/\binom{n}{q})$ 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 $\hat{\mathrm{P}}^+$) and the lower bound (denoted by $\hat{\mathrm{P}}^-$), respectively.

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

Now let us consider the case of $x = 1$. Assume that the shared spreading code is compromised, which happens with probability $\alpha = 1 - \binom{n-l}{q}/\binom{n}{q}$ 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 apply different jamming strategies to them. The first HELLO message can be jammed if $\mathcal{J}$ selects the correct code, which happens with probability $\beta = \min\{\frac{z(1+\mu)}{\mu c}, 1\}$. 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 $\beta' = \min\{\frac{3z(1+\mu)}{\mu c}, 1\}$. So we have $\hat{\mathrm{P}}^{+}(1) = 1 - \alpha + \alpha(1-\beta)(1-\beta') = 1 - \alpha(\beta + \beta' - \beta\beta')$.

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 $\hat{\mathrm{P}}^{+}(x) = 1 - \alpha^x + \alpha^x(1 - (\beta + \beta' - \beta\beta')^x) = 1 - \alpha^x(\beta + \beta' - \beta\beta')^x$.

Summarizing the above cases, we have $\hat{\mathrm{P}}^{+} = \sum_{x=0}^{m} \hat{\mathrm{P}}^{+}(x)\mathsf{Pr}[x] = 1 - \sum_{x=0}^{m} \mathsf{Pr}[x]\alpha^x(\beta + \beta' - \beta\beta')^x$.

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 $\hat{\mathrm{P}}^{-} = 1 - \sum_{x=0}^{m} \mathsf{Pr}[x]\alpha^x$. The actual jamming performance of $\mathcal{J}$ is between random and reactive jamming, i.e., $\hat{\mathrm{P}}^{-} \leq \hat{\mathrm{P}}_{\mathrm{D}} \leq \hat{\mathrm{P}}^{+}$. ■

The following theorem is about the average neighbor-discovery latency $\bar{\mathrm{T}}$ of D-NDP.

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

$$\bar{\mathrm{T}}_{\mathrm{D}} \approx \frac{\rho m(3m+4)N^2 l_h}{2} + \frac{2Nl_f}{R} + 2t_{key}\ , \qquad (3)$$

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

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 $\mathrm{T_i}$. 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 $t_A^{\mathrm{d}} = T_7 - T_6$ be 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 $\{\mathrm{CONFIRM}, ID_B\}_{\mathbf{C}_i}$ 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

$$\begin{aligned}\mathsf{E}[\mathrm{T_i}] &\approx \mathsf{E}[t_B^{\mathrm{r}}] + \mathsf{E}[t_B^{\mathrm{d}}] + \mathsf{E}[t_A^{\mathrm{r}}] + \mathsf{E}[t_A^{\mathrm{d}}] \\ &= \frac{t_p}{2} + \frac{t_p}{2} + \frac{t_p}{2} + \frac{\lambda t_h}{2} \\ &= \frac{\rho m(3m+4)N^2 l_h}{2}.\end{aligned} \qquad (4)$$

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 $N(1+\mu)(l_{id} + l_n + l_{mac})/R$. Each node also needs computing the shared key. We then have

$$\mathsf{E}[\mathrm{T_a}] = \frac{2N(1+\mu)(l_{id} + l_n + l_{mac})}{R} + 2t_{key}\ . \qquad (5)$$

Finally, we have $\bar{\mathrm{T}}_{\mathrm{D}} = \mathsf{E}[\mathrm{T_i}] + \mathsf{E}[\mathrm{T_a}]$ as shown in Eq. (3). ■

3) Analysis of M-NDP: We have the following theorem regarding the neighbor-discovery probability $\hat{\mathrm{P}}_{\mathrm{M}}$ of M-NDP.

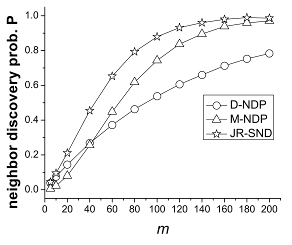
Theorem 3: *For $\nu = 2$, two physical neighbors can discover each other via M-NDP with probability $\hat{\mathrm{P}}_{\mathrm{M}} \geq 1 - (1 - \hat{\mathrm{P}}_{\mathrm{D}}^2)^{g(1-\frac{3\sqrt{3}}{4\pi})-1}$, where g denotes the average number of physical neighbors of MANET nodes.*

Proof: Assuming that no nodes have performed M-NDP yet, any two physical neighbors can discover each other and establish a session spread code via D-NDP with probability $\hat{\mathrm{P}}_{\mathrm{D}}$. Assume that A and B are two physical neighbors but cannot discover each other via D-NDP. They, however, can discover each other via a 2-hop M-NDP process if they have at least one common neighbor who has established session codes via D-NDP with both of them. For each common physical neighbor, the probability that it can discover both A and B via D-NDP is $\hat{\mathrm{P}}_{\mathrm{D}}^2$.

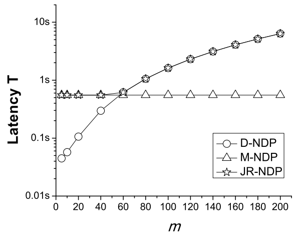
The expected overlapping area of A's and B's transmission ranges can be derived as $(\pi - \frac{3\sqrt{3}}{4})a^2$, where a is the radius of each node's transmission range [11]. Let g be the average number of physical neighbors that each node has, and g depends on the total number of nodes n, node mobility and distribution, and the network shape. The expected number of common physical neighbors of A and B is then approximately $g(1 - \frac{3\sqrt{3}}{4\pi}) - 1$, excluding A or B themselves. Nodes A and B can perform 2-hop M-NDP with probability $1 - (1 - \hat{\mathrm{P}}_{\mathrm{D}}^2)^{g(1-\frac{3\sqrt{3}}{4\pi})-1}$. In practice, some other nodes may have already discovered each other via M-NDP. Therefore, we have $\hat{\mathrm{P}}_{\mathrm{M}} \geq 1 - (1 - \hat{\mathrm{P}}_{\mathrm{D}}^2)^{g(1-\frac{3\sqrt{3}}{4\pi})-1}$. ■

Table I
DEFAULT EVALUATION PARAMETERS

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



(a) $\hat{\mathrm{P}}$



(b) $\bar{\mathrm{T}}$

Figure 2. Impact of m.

We have not been able to give a closed-form solution to $\hat{\mathrm{P}}_{\mathrm{M}}$ for $\nu \geq 3$, which instead will be evaluated via simulations later. We also have the following theorem regarding the average neighbor-discovery latency $\bar{\mathrm{T}}$ of M-NDP without giving the straightforward proof.

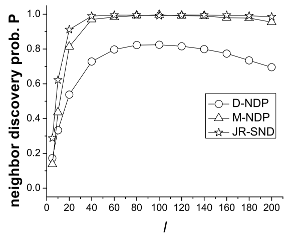
THEOREM 4: *Two physical neighbors connected by a ν-hop jamming-resilient path can discover each other via M-NDP with an average latency* $\bar{\mathrm{T}}_{\mathrm{M}} = \mathrm{T}_\nu + 2\nu(\nu+1)t_{ver} + 2\nu t_{sig}$, *where* $\mathrm{T}_\nu = \frac{N}{R}(\frac{3\nu(\nu+1)}{2}((g+1)l_{id}+2l_{sig})+2\nu(l_n+l_\nu))$.

It follows that JR-SND enables two physical neighbors to discover each other with probability $\hat{\mathrm{P}} = \hat{\mathrm{P}}_{\mathrm{D}}+(1-\hat{\mathrm{P}}_{\mathrm{D}})\hat{\mathrm{P}}_{\mathrm{M}}$. Moreover, the average neighbor-discovery latency $\bar{\mathrm{T}}$ of the JR-SND is bounded by the larger one between that of D-NDP and that of M-NDP, i.e., $\bar{\mathrm{T}} = \max(\bar{\mathrm{T}}_{\mathrm{D}}, \bar{\mathrm{T}}_{\mathrm{M}})$.

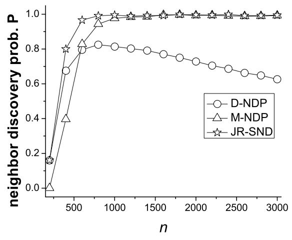
B. Simulation Results

We simulate 2000 MANET nodes in a 5000×5000 m^2 field, each with a transmission range of 300 m. Table I summarizes most default evaluation parameters unless specified otherwise, in which the cryptographic parameters are adopted from [13]. 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. In our simulations, reactive jamming is always more effective in jamming neighbor discovery than random jamming, which coincides with the intuition. For lack of space, we only show the results under reactive jamming, which correspond to the worst-case scenario.

Fig. 2 show the impact of m. In general, the larger m, the higher $\hat{\mathrm{P}}$ for D-NDP, M-NDP, and JR-SND. The $\bar{\mathrm{T}}$ of D-NDP increases quadratically as m increases and exceeds that of M-NDP when $m > 60$. The $\bar{\mathrm{T}}$ of JR-SND is the larger one between that of D-NDP and M-NDP. Under default

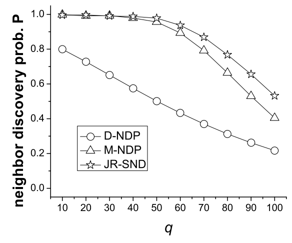


(a) $\hat{\mathrm{P}}$ vs. l

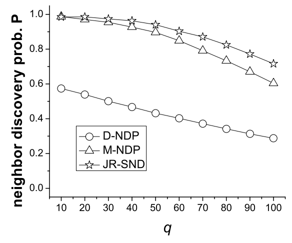


(b) $\hat{\mathrm{P}}$ vs. n

Figure 3. Impact of l and n.



(a) $l = 40$



(b) $l = 20$

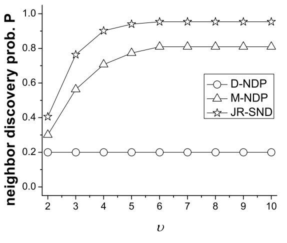
Figure 4. Impact of q.

configuration, i.e., $m = 100$, JR-SND has a latency under 2 seconds, which makes it very suitable for MANETs with high mobility.

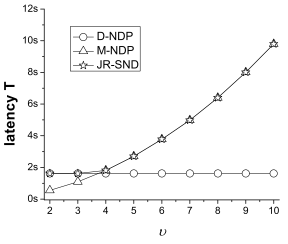
Fig. 3 show the impact of l and n (the number of nodes). As shown in Fig. 3(a), $\hat{\mathrm{P}}$ of JR-SND increases as l increases from 5 to 100, after which it decreases slowly. The reason is that as l increases, the probability that two nodes share at least one codes increases, leading to the increase in $\hat{\mathrm{P}}$. On the other hand, given a fixed number of compromised nodes, the probability of any code being compromised also increases with l. After l exceeds some threshold, the later effect overwhelms the former, leading to the decrease of $\hat{\mathrm{P}}$ for D-NDP, M-NDP, and JR-SND.

A similar trend for the impact of n on $\hat{\mathrm{P}}$ can be observed from Fig. 3(b). For fixed l, m, and q, the probability α of any code being compromised decreases with n increases, leading to the increase of $\hat{\mathrm{P}}$ for D-NDP and M-NDP when n is small. On the other hand, the probability that two nodes share at least one code decreases with n. When n exceeds some threshold, the later effect dominates and results in the decrease of $\hat{\mathrm{P}}$ for D-NDP. Also, a larger n increases the node density, which eventually leads to higher $\hat{\mathrm{P}}$ for M-NDP. Since the performance of JR-SND depends on the combination of D-NDP and M-NDP, the overall performance is always sufficiently good.

Fig. 4 show the impact of q (the number of compromised nodes), where $l = 40$ and $l = 20$, respectively. As q increases, $\hat{\mathrm{P}}$ of D-NDP, M-NDP, and JR-SND all decrease, which is anticipated. For example, when $l = 40$ and $q = 60$, $\hat{\mathrm{P}}$ of JR-SND drops to approximately 0.5. However, as argued in Section IV-B, we believe that node compromise



(a) $\hat{\mathrm{P}}_{\mathrm{D}} = 0.2$



(b) $\bar{\mathrm{T}}$

Figure 5. Impact of ν.

attacks are much more difficult to launch in MANETs than in unattended sensor networks. In addition, we can greatly improve the resilience of M-NDP and thus JR-SND to node compromise attacks by increasing ν, the number of hops an M-NDP request can traverse.

Fig. 5(a) shows the impact of ν on $\hat{\mathrm{P}}_M$, the neighbor-discovery probability of M-NDP, where $\hat{\mathrm{P}}_{\mathrm{D}} = 0.2$ which corresponds to $q = 100$ as shown in Fig. 4(a). D-NDP is not affected by ν, so $\hat{\mathrm{P}}_{\mathrm{D}}$ is plotted for reference only. We can see that the larger ν, the larger $\hat{\mathrm{P}}_{\mathrm{M}}$ and $\hat{\mathrm{P}}$ with an always reasonably low latency $\bar{\mathrm{T}}$, and vice versa. In particular, when $\nu \geq 6$, the M-DNP and JR-SND can achieve $\hat{\mathrm{P}}$ over 0.9. In practice, MANET nodes may dynamically adjust ν to achieve satisfactory neighbor-discovery probabilities. In addition, Fig. 5(b) shows that the $\bar{\mathrm{T}}$ of M-NDP increases as ν increases. For example, when $\nu = 6$, $\bar{\mathrm{T}}$ is about 4 seconds, which is acceptable in most cases.

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.

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 can improve the neighbor-discovery probability by letting two neighbors discover each other via a multi-hop path. It is suitable for MANETs with moderate node density. By combining D-NDP and M-NDP, two nodes can discover each other with overwhelming probability and low latenc

VII. Conclusion

In this paper, we propose JR-SND, a novel scheme 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. As our future work, we intend to further evaluate JR-SND with experimental results.

Acknowledgement

This work was supported in part by the US National Science Foundation under grants CNS-0716302 and CNS-0844972 (CAREER).

References

[1] R. Pickholtz, D. Schilling, and L. Milstein, "Theory of spread-spectrum communications–a tutorial," *IEEE Transactions on Communications*, vol. 30, no. 5, pp. 855–884, May 1982.

[2] L. Baird, W. Bahn, M. Collins, C. Carlisle, and C. Butler, "Keyless jam resistance," in *IEEE IWA'07*, Montreal, CA, June 2007, pp. 143–150.

[3] M. Strasser, C. Popper, S. Capkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *IEEE S&P'08*, Oakland, CA, May 2008.

[4] M. Strasser, C. Popper, and S. Capkun, "Efficient uncoordinated fhss anti-jamming communication," in *MobiHoc'09*, Apr. 2009, pp. 207–218.

[5] D. Slater, P. Tague, R. Poovendran, and B. J. Matt, "A coding-theoretic approach for efficient message verification over insecure channels," in *WiSec'09*, Zurich, Switzerland, Mar. 2009, pp. 151–160.

[6] T. Jin, G. Noubir, and B. Thapa, "Zero pre-shared secret key establishment in the presence of jammers," in *MobiHoc'09*, Apr. 2009, pp. 219–228.

[7] C. Popper, M. Strasser, and S. Capkun, "Jamming-resistant broadcast communication without shared keys." in *USENIX Security'09*, Aug. 2009.

[8] Y. Liu, P. Ning, H. Dai, and A. Liu, "Randomized differential DSSS: Jamming-resistant wireless broadcast communication," in *INFOCOM'10*, San Diego, CA, Mar. 2010.

[9] A. Liu, P. Ning, H. Dai, and Y. Liu, "USD-FH: Jamming-resistant wireless communication using frequency hopping with uncoordinated seed disclosure," in *MASS'10*, San Francisco, CA, Nov. 2010, pp. 41–50.

[10] A. Liu, P. Ning, H. Dai, Y. Liu, and C. Wang, "Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure," in *ACSAC'10*, Austin, Texas, Dec. 2010, pp. 367–376.

[11] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE S&P'03*, Oakland, CA, May 2003, pp. 197–213.

[12] E. Sousa and J. Silvester, "Optimum transmission ranges in a direct-sequence spread-spectrum multihop packet radio network," *IEEE J. Select. Areas Commun.*, vol. 8, no. 5, pp. 762–771, Jun 1990.

[13] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing mobile ad hoc networks with certificateless public keys," *IEEE Transactions on Dependable and Secure Computing,*, vol. 3, no. 4, pp. 386–399, Oct.-Dec. 2006.

[14] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO'01*, Santa Barbara, CA, Aug. 2001, pp. 213–229.

[15] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, vol. 8, no. 2, pp. 300–304, June 1960.