From Time Domain to Space Domain: Detecting Replica Attacks in Mobile Ad Hoc Networks

Kai Xing
Dept. of Computer Sci. & Tech. / Suzhou Inst. for Advanced Study
University of Science & Technology of China
Hefei, Anhui 230027, P. R. China
xingkai@ustc.edu

Xiuzhen Cheng
Department of Computer Science
The George Washington University
Washington, DC 20052, USA
cheng@gwu.edu

***Abstract*—A common vulnerability of wireless networks, in particular, the mobile ad hoc network (MANET), is their susceptibility to node compromise/physical capture attacks since the wireless devices are often not protected by tamper-resistant hardware due to small form factors and low cost, and can be easily stolen/lost or temporarily controlled by unauthorized entities due to their harsh working environments. A serious consequence of the device capture attack is the node replication attacks in which adversaries deploy a large number of replicas of the compromised/captured nodes throughout the network. Replicated nodes have all legitimate security credentials and therefore can launch various insider attacks, or even take over the network easily. They are indeed "attack multipliers" and therefore are extremely destructive to the network.**

Detecting replication attacks is a nontrivial problem in MANETs due to the challenges resulted from node mobility, cloned/compromised node collusion, and the large number and wide spread of the replicas. Existing approaches either fail in mobile environments due to the limitations caused by local views or their dependence on invariant claims such as location and neighbor list, or are constrained by the number, distribution, and colluding activities of the replicas. In this paper, we propose two replication detection schemes (TDD and SDD) to tackle all these challenges from both the time domain and the space domain. Our theoretical analysis indicates that TDD and SDD provide high detection accuracy and excellent resilience against smart and colluding replicas, have no restriction on the number and distribution of replicas, and incur low communication/computation overhead. To our best knowledge, TDD and SDD are the only approaches that support mobile networks while place no restrictions on the number and distribution of the cloned frauds and on whether the replicas collude or not.

Keywords: mobile ad hoc networks (MANETs), node replication attacks, clone attacks, insider attacks

I. Introduction

A recent study [1] on vulnerabilities and attacks in mobile ad hoc networks (MANETs) has particularly addressed the high possibility of node compromise/physical capture by adversaries. A serious consequence of this security threat is the launch of the *node replication attack* in which adversaries deploy the replicas of the compromised/captured nodes throughout the network. Tamper-resistant hardware to protect against device clones is both unrealistic and inadequate due to the low cost, small form factor, and the limited resources in MANET devices [1].

Replicated nodes may threaten the network functionalities significantly. In MANETs, all the communication protocols heavily rely on message relay/forwarding. To mislead these protocols, the replicas may collude with each other to selectively block traffics in the network, jam the communication channels, and launch wormhole attacks [2], [3] by creating low latency and high bandwidth channels among different locations. The replicas may also collude with each other to defeat typical fault tolerant mechanisms.

Additionally, since the replicas have legitimate security credentials (legitimate IDs, keys, etc.), they can access the network and participate in the network operations in the same way as a legitimate node does. This means that replicas can trigger a large variety of insider attacks [4]–[9], or even take over the network. In other words, replicas are indeed "attack multipliers" and therefore are extremely destructive. Thus it is imperative to provide effective and efficient solutions for replica detection in MANETs to limit their damage.

A. Inadequacy of Current Solutions

Detecting node replication attacks in MANETs must handle the challenges caused by node mobility, cloned/compromised node collusion, and the large number and wide spread of the replicas. Existing approaches either fail due to node mobility [10]–[15] and/or the limited view on the whole network by the detectors [4], [9], [16], [17], or have limited ability for detecting smart replicas due to node collusion [18], [19].

In [20], Shaw *et al.* proposed to store the unique device fingerprint characterizing its signal transmission at the base station, and thus the replicas can be detected accordingly. A similar procedure has been employed by cellular network operators to prevent phone cloning fraud. However, in multi-hop mobile environments, it is almost impossible for base stations to track the signal characteristics of nodes multiple-hop away. While in localized voting/misbehavior detection schemes [4], [16], [17], [21], nodes only have their local observations and thus lack the ability to detect distributed replicas that may appear anywhere in the network.

To our best knowledge, the first non-naive distributed node replication detection scheme is due to Parno *et al.* [10], which proposed the randomized multicast (RM) and the line-selected multicast (LSM) schemes to detect replicas in stationary sensor networks. In these approaches, each node reports its location claims to its witness nodes that can be selected either randomly or from a routing path. A witness node having received conflicting reports of a sensor reports the node replication attack. Conti *et al.* [13] further improved the detection accuracy and robustness of the RM scheme by deterministically selecting the witness nodes at each round based on a random number generated from a seed that varies randomly from round to round. In [12], location claims were stored at all nodes (witnesses) within a single cell or multiple cells determined based on a probabilistic function.

A centralized detection scheme was proposed by Choi *et al.* [11] for stationary sensor networks, where the network is divided into exclusive subregions organized hierarchically (e.g. quad tree). Each subregion enumerates a header node that is in charge of reporting the list of members in the subregion. All headers form a hierarchical tree along which the member lists of all subregions are reported to the base station. When uniting the member lists reported by the children nodes, a header checks the intersection of the two reports and any non-empty intersection indicates the existence of cloned sensors. Xing *et al.* [14] proposed a hybrid approach, which detects replicas either locally or at the base station. In this scheme, each node computes a social fingerprint characterizing the local neighborhood (neighbors) for each neighbor and itself. The fingerprint of a node may be piggybacked to the base station. The neighboring nodes and the base station perform replica detection by using the property that fingerprints of replicas conflict each other. A different approach was adopted by Brooks *et al.* [15], which proposed a centralized detection method in which the base station collects the random (master) key usage statistics[1] from the network and detects replica attacks at frequency domain, i.e., if a key is used too often, the base station reports a replica attack. This approach heavily relies on the underlying basic random key establishment scheme [22], in which a random key ring is preloaded to each node and two neighboring nodes establish a shared key if their key rings have at least one key in common. The effectiveness of this approach is based on the following assumption: the number of node pairs that select the same random key is independent and identically distributed.

We claim that none of the approaches mentioned above is capable of detecting replica attacks in mobile environments. If the approaches in [**?**], [10]–[13] are applied in MANETs, the location-related claims of a node (e.g., locations, neighbor lists, social fingerprints, etc.) will change when node moves, and thus not only the replicas but also the benign nodes will generate contradictory location related claims in the network, which nullifies the capability of these approaches in detecting replica attacks. Similarly, if the scheme in [15] is applied in MANETs, the key usage in the network is not necessarily identically distributed due to node mobility.

In addition, all the centralized approaches and distributed approaches mentioned above [10]–[15] may fail to detect replica attacks if there are enough number of replicas/compromised nodes colluding with each other in the network. For example, the replicas/compromised nodes may block certain reports to witness nodes and thus prevent themselves from being detected. In other words, these approaches detect replicas by assuming a limited number of replicas/compromised nodes in the network. However, this assumption is not necessarily true: even if an adversary can compromise only a limited amount of nodes, it can deploy unlimited number of replicas in the network and thus take over the network and manipulate/block selected network traffics to avoid being detected.

There have been two recent studies for replica attack detection in mobile environments [18] [19]. Yu *et al.* [18] proposed a localized solution termed as XED, in which each node remembers the latest random numbers it has exchanged with the nodes it meets. When two nodes meet again, they check with each other the random number they have exchanged last time. If the random number does not match the one in their memories, they report the detection of a replica. This scheme has no limitation on the number of replicas/compromised nodes in the network and incurs extremely low computation/communication overhead. However, it may easily lose its power to detect replicas if the replicas collude with each other to synchronize their random number to respond to the benign nodes. In [19], Ho *et al.* proposed a centralized solution, in which the base station collects location claims from the network to compute the node speed and launches the sequential probability ratio test (SPRT) to detect replicas. If the measured speed of a node is beyond the system-configured maximum speed, there likely exists a replica attack in the network. If the abnormal speed measurements of an identity appear multiple times within a certain duration, SPRT reports a node replication attack on that identity. A drawback of SPRT is its lacks of the ability to detect collusive replicas, e.g., the replicas may collude with each other to block selected traffics to the base station, or they could schedule their movements and the time slots to be silent such that their measured speeds never exceed the system-configured maximum speed.

As discussed above, the existent approaches can not handle all the challenges faced by replication attacks in mobile networks. They may not be applicable when compromised and cloned nodes collude, or when there exist a large number of replicas distributed in a wide range. Our approach is superior in that a high detection accuracy and excellent resilience against collusive replicas can be achieved at the cost of a relatively low communication/computation overhead. Furthermore, our scheme has no restriction on the number and distribution of the replicas/compromised nodes.

B. Our Contributions

In this paper, we propose to detect node replication attacks at time domain (TDD) and space domain (SDD) in MANETs. In our approach, a cryptographic one-way hash function is utilized to force replicas to keep on generating paradoxes whose detection reveals the existence of replica attacks in the network. Comparing with the related literature summarized in

[1]Here key usage refers to the number of times a key is used to set up connections between neighboring nodes, i.e., the number of times a key is used for authentication, not for message encryption/decryptioin.

Section I-A, we have identified the following unique contributions of this paper.

- Our schemes have the power to force the replicas in MANETs to keep on generating paradoxes along their movements and when they communicate. By looking up these paradoxes, our schemes can detect the replicas at a very high accuracy.
- Our schemes provide excellent resilience against collusive replicas/compromised nodes and have no limitation on the number and distribution of replicas/compromised nodes, both of which are significant improvements compared to the previous attempts [10]–[15], [18], [19].
- Our schemes explore the simple one-way hash function for replica attack detection, which results in low computation overhead.
- Our SDD scheme is a purely localized scheme that produces low communication overhead.
- Our schemes make no assumption on the underlying mobility model. Therefore they are applicable to a wide range of mobile networks.

C. Organization of the Paper

The rest of the paper is organized as follows. In Section II, we introduce the network and the security models. In Section III, we detail TDD and SDD, the two novel replication detection schemes, and their underlying rationales. The security analysis and performance evaluation are reported in Section IV. Section V summarizes the work and concludes the paper.

II. Models, Assumptions and Notations

A. Network Model

We consider a mobile ad hoc network comprised of N nodes, with each having the minimum and maximum communication ranges of R_{min} and R_{max}, respectively. Before deployment, each node u is preassigned with a unique identity and a pair of public and private keys[2]. u is also preloaded with a public one-way hash function F and a private unique seed, based on which u can generate a one-way hash chain (called *challenge chain* thereafter)[3] denoted by $ch_0^u, ch_1^u, ch_2^u, ..., ch_i^u, ch_{i+1}^u, ...$, where $ch_i^u = F(ch_{i+1}^u)$. Besides, every node v in the network is preloaded with u's challenge ch_0^u.

We also assume that each node stores a public location generation function H_{loc} and a public time generation function H_{time}, where $H_{loc}(u, t)$ computes a location in the network given a node id u and a time t, and $H_{time}(u, ch, T_i)$ computes a time moment in the time interval T_i given an id u, a challenge ch, and a time interval T_i. In our study, $H_{loc}(u, t)$ is a simple hash function mapping the node u at time t to a random location in the network. To compute $t = H_{time}(u, ch, T_i)$, we first divide T_i into N time epochs of equal length, i.e., $t_{i_1}, t_{i_2}, \cdots, t_{i_N}$. Suppose each identity u corresponds to an integer j_1 in $[1, N]$, we then use the challenge ch as the seed to map j_1 to an integer j_2 in $[1, N]$. Finally, we set $t = t_{i_{j_2}}$.

After deployment, each node in the network may exercise different mobility models upon needs and vary its speed between 0 and $Speed_{max}$. In our model, we further assume that every node is able to obtain its location information and verify the locations of its neighbors via GPS [25] or other positioning schemes such as those proposed in [26]–[32]. Furthermore, we assume that the clocks of all nodes are loosely synchronized [33]–[35]. In particular, we use $t_1, t_2, \cdots, t_i, \cdots, t_j, \cdots$ to represent the time instants in the network, where $t_i < t_j$ given $i < j$, $i, j \in Z^+$.

B. Security Model

For simplicity, we assume that there exists a trusted node TN (e.g., base station) that is sufficiently powerful to defend itself against all kinds of security threats. This TN is used by our time-domain detection scheme (TDD) only.

In accordance with the existent work, we assume that the compromise or capture of a node releases all its security information to the attacker. An adversary that has captured a node u can deploy replicas of u anywhere in the network. Note that the replicas own all the legitimate information of the compromised node (e.g. ID, keys, code, etc.). Thus they can easily participate in network operations in the same way as a legitimate node does, and therefore launch various internal attacks afterward. For example, a replica can fabricate messages to mislead the decision makers, or keep injecting bogus data to cause network outage.

In addition, we assume the replicas under the control of the adversary have the ability to simultaneously communicate and collaborate with each other; they are smart and rational, and thus may attempt to avoid detection by colluding with each other or being silent at selected time period. Under this assumption, all existing approaches lack the ability to detect certain types of node replication attacks in MANETs.

III. Detection Protocols Against Node Replication Attacks

In this section, we will tackle the problem of detecting node replication attacks from both the time domain and the space domain. Correspondingly, we will detail the Time-Domain Detection (TDD) scheme and the Space-Domain Detection (SDD) scheme. To proceed, we need to introduce our local information exchange protocol first, which serves as the basis for both TDD and SDD.

A. Local Information Exchange

Whenever two nodes u and v meet each other, they exchange the following authenticated messages:

$$Message_{u \rightarrow v} : (u||v||t^u||loc^u||ch_i^u||Sig^u),$$

$$Message_{v \rightarrow u} : (v||u||t^v||loc^v||ch_j^v||Sig^v),$$

[2]Recent works [23], [24] indicate that public key algorithms are practical for low-cost hardware.

[3]Because of the one-way property of the hash function F, given ch_i^u in the challenge chain, anyone can compute all the previous challenges $ch_{j_1}^u$, $0 \le j_1 < i$, but no one can compute any latter challenge $ch_{j_2}^u$, $i + 1 \le j_2$.

where t and loc represent the time and the location at which they meet each other, ch_i^u (ch_j^v) represents u's (v's) challenge whose index is the smallest among all unissued challenges in u's (v's) challenge chain, and Sig^u (Sig^v) is the signature signed by u (v) with u's (v's) public key for $Message_{u \to v}$ ($Message_{v \to u}$).

After the exchange process, u and v remove ch_i^u and ch_j^v from their challenge chains, respectively. The challenge with the smallest index among all unissued challenges in u's (v's) challenge chain is then ch_{i+1}^u (ch_{j+1}^v).

The nodes $v_1, v_2, \cdots$ that have exchanged information with u during a fixed past time duration $2D$ are the witness nodes of u, where the setting of D will be clearly explained in Section IV. To store the information that u receives from these witness nodes, u maintains a table, as shown in Table I.

The nodes u has met	v_1	v_2	...
			...
The information u has recorded	$(t_{i_1}^{v_1}, loc_{i_1}^{v_1}, ch_{i_1}^{v_1}, Sig_{i_1}^{v_1})$	$(t_{j_1}^{v_2}, loc_{j_1}^{v_2}, ch_{j_1}^{v_2}, Sig_{j_1}^{v_2})$	...
	NA	$(t_{j_2}^{v_2}, loc_{j_2}^{v_2}, ch_{j_2}^{v_2}, Sig_{j_2}^{v_2})$	...
	NA	...	...
	...	...	...

TABLE I
THE TABLE OF u WHICH RECORDS THE INFORMATION THAT u HAS RECEIVED FROM THE NODES IT HAS ENCOUNTERED WITHIN THE PAST TIME DURATION $2D$.

In order to force every node to faithfully follow the information exchange process, we apply the following strategies:

- Every node periodically beacons its neighborhood information to its neighbors, and thus every node has the knowledge of the nodes within its two-hop distance.
- Once u is leaving from v's communication range, e.g., v does not receive u's beacon at the expected time, v broadcasts the most recently heard challenge of u to its neighbors so that the nodes around u always have the most recently released challenge of u.
- All nodes within the one-hop neighborhood of u at time t, denoted by $\mathcal{N}(u,t)$, monitor u (e.g., watchdog [36]) to ensure that the challenges of u they have recently heard from the air preserve the order in u's challenge chain. Specifically, they should not only preserve one-way property but also are close to each other in u's challenge chain. This challenge check could be done by merely performing hash function operations.

Given the above strategies, a node u:

- cannot appear as v's one-hop neighbor without being v's two-hop neighbor first, and can't disappear from v's one-hop neighborhood without being v's two-hop neighbor, unless the network is sufficiently sparse;
- has to launch the information exchange process with every node v it meets because otherwise a node around u that has the knowledge of u's neighborhood and is expecting to hear this information exchange process in the air, may detect u's abnormal silence;
- cannot manipulate its challenges because the challenges that u issues and the order of their issuance is monitored and checked by u'sneighbors: (i) u cannot issue duplicate challenges since each challenge in u's challenge chain is unique; (ii) newly issued challenges must preserve the one-way property, i.e., they can be used to derive the ones issued earlier but cannot be the other way; and (iii) u cannot skip a challenge in its challenge chain if it is continuously monitored by its neighbors.

Remark: This local information exchange protocol prevents compromised nodes and their clones from colluding to synchronize the bindings of *time*, *location*, and *challenge* since neighboring nodes can easily detect a node reporting wrong time and location information.

B. The Underlying Rationales of Our Approach

The local information exchange protocol implies the following facts:

Fact 3.1: At any specific time, the location presence and the challenge presence of an identity in the network should be unique.

Fact 3.2: The location presence and the challenge presence of an identity should be in consecutive order along the locus and the time coordinates.

Given the following two messages that v and w have received from u,

$$(u||v||t_i^u||loc_i^u||ch_{k_i}^u||Sig_i^u),$$
$$(u||w||t_j^u||loc_j^u||ch_{k_j}^u||Sig_j^u),$$

according to Fact 3.1 and Fact 3.2, the information in these two messages must satisfy the following lemmas. Violating any of them indicates a node replication attack on identity u.

Lemma 3.1: The ordinal order of $ch_{k_i}^u$ and $ch_{k_j}^u$ in u's challenge chain must be the same as that of t_i and t_j, namely

$$k_i < k_j, \quad \text{if and only if} \quad t_i < t_j. \tag{1}$$

Besides, if $v \neq w$, $ch_{k_i}^u \neq ch_{k_j}^u$

Proof: The lemma holds according to Fact 3.1 and Fact 3.2. ■

Lemma 3.2: The linear distance between loc_i^u and loc_j^u must be less than or equal to the maximum possible distance u can traverse during the time interval $|t_j - t_i|$, namely

$$||loc_i^u - loc_j^u|| \leq Speed_{max} \times |t_j - t_i|. \tag{2}$$

Proof: This lemma holds trivially. ■

Lemma 3.3: The number of nodes that u meets between t_j and t_i should be less than or equal to the maximum possible number of nodes it can meet during the time interval $|t_j - t_i|$. For example,

$$\begin{aligned} |k_i - k_j| \quad \leq \quad & (Speed_{max} \times |t_j - t_i| \times 2R_{max} + \pi R_{max}^2) \\ & \times (\rho + 3\sigma), \end{aligned} \tag{3}$$

holds with a probability at least 99% if the node density satisfies $\mathcal{N}(\rho, \sigma)$ with the mean ρ and the standard derivation σ.

Proof: The lemma holds trivially. ■

C. TDD: Time-Domain Detection

In this section, we present our TDD scheme, a distributed mechanism to detect node replication attacks in MANETs. In our consideration, every node has the responsibility to take part in the detection procedure.

Algorithm 1 TDD Check on Identity u at Time Interval T_k

Input: u, ch_k^{TN}, T_k
Output: the check result of identity u: 1 =node replication attack detected, 0 =normal, N/A=not available.

```
1:  function TDD($u, ch_k^{TN}, T_k$)
2:      $\forall\ v$ in the network, where $v \neq u$, $v$ computes
3:      $t \leftarrow H_{time}(u, ch_k^{TN}, T_{k-1})$    ▷ Compute a time $t$ for $u$.
4:      if $v$ did meet $u$ at around time $t$ ($t \pm \Delta$) then
5:          $Destination \leftarrow H_{loc}(u,t)$    ▷ Compute the location where to send the report.
6:          $Destination \Leftarrow Message_{u\rightarrow v} || sig^v$  ▷ $Message_{u\rightarrow v}$, sent from $u$ to $v$ at time $t \pm \Delta$, is sent to location $H_{loc}(u,t)$ along with $v$'s signature.
7:      else
8:          Return N/A    ▷ $v$ stops the reporting procedure on $u$.
9:      end if
10:     According to Lemma 3.1-3.3,
11:     if all $Message_{u\rightarrow v}$ received at the location $H_{loc}(u,t)$ are consistent with each other then
12:         Return 0
13:     else
14:         Return 1
15:     end if
16: end function
```

The basic idea of TDD is sketched as follows: Let $ch_0^{TN}, ch_1^{TN}, ch_2^{TN}, \ldots, ch_k^{TN}, \cdots$ denote the challenge chain of the trusted node TN. We divide time into equal-length intervals $T_0, T_1, T_2, \cdots, T_k, \cdots$, and associate the time interval T_k with the challenge ch_k^{TN}. At the beginning of each time interval T_k, TN broadcasts the challenge ch_k^{TN} to every node v in the network. Based on the one-way property of the hash chain, v can easily verify the authenticity of ch_k^{TN} by using any of the previously verified challenge or the preloaded ch_0^{TN}.

Once receiving ch_k^{TN}, v computes a time $t \in T_{k-1}$ for each node u in the network using the time generation function H_{time} taking the identity u, the challenge ch_k^{TN}, and T_{k-1} as inputs:

$$t = H_{time}(u, ch_k^{TN}, T_{k-1}), t \in T_{k-1}$$

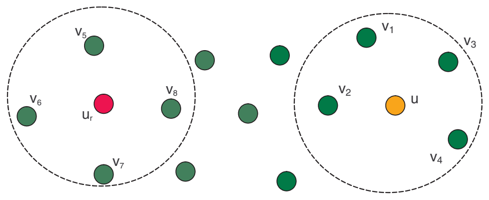
If u and v did meet each other at a time interval $t \pm \Delta$, where Δ is a system-defined parameter whose setting will be analyzed in Section IV, v reports the information that it has obtained from u during $t \pm \Delta$ to the location $H_{loc}(u,t)$. Otherwise v stops its reporting procedure on u.

At the location $H_{loc}(u,t)$, the node closest to $H_{loc}(u,t)$ is responsible for receiving these reports and launching a check on them. Messages from different reporters should have consistent location and challenge claims on u according to the lemmas elaborated in Section III-B. Any violation of these rules signals a node replication attack on identity u.

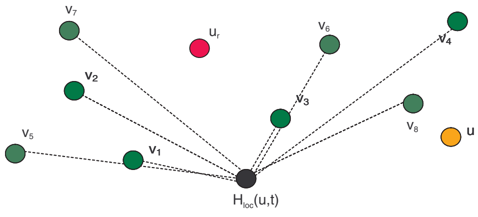
Note that $t \in T_{k-1}$, but the computation of t and the reporting procedure is deferred to the time interval T_k. With this postponed reporting strategy, we can see that

1) the time $t \in T_{k-1}$ can not be determined before the trusted node TN releases the challenge ch_k^{TN} at the next time interval T_k. According to the one-way property of TN's challenge chain, during time interval T_{k-1}, no one can determine the time t, the reporters, and the destination of the reports, therefore effectively preventing colluding clones from escaping detection.
2) when launching the postponed reporting procedure, the reporters of u may have moved to anywhere in the network. Namely the reports are sent from various places in the network, which may greatly increase the cost for the replicas/attackers to stop from being reported.

Remark: The parameter Δ affects the timely detection of replica attacks. A too small value will delay the detection while a too large value will increase the communication overhead. In Section IV, we will study the impact of this parameter on the performance of TDD and provide a proper setting to Δ in Section IV.



(a) At the time interval $t \pm \Delta$, $t \in T_{k-1}$



(b) At the beginning of time interval T_k

Fig. 1. (a) There are two nodes u and u_r using the same identity u in the network. Nodes $v_1, v_2, ..., v_8$ are the witnesses that encounter either u or u_r at the time interval $t \pm \Delta$, where $t \in T_{k-1}$. (b) At the beginning of the time interval T_k, $v_1, v_2, ..., v_8$ report to location $H_{loc}(u,t)$ the information that they have received from u at time $t \pm \Delta$.

Example: Given u and its replica u_r, as shown in Fig.1(a), the appearance of identity u in the network at time $t \in T_{k-1}$ is not unique. According to Fact 3.1 and Lemma 3.2, this could be taken as an indicator of node replication attacks. Suppose $v_1, v_2, ..., v_8$ are the witness nodes that met u or u_r at the time interval $t \pm \Delta$. At T_k, we let $v_1, v_2, ..., v_8$ report to location $H_{loc}(u,t)$ the information they have received from u at the time $t \in T_{k-1}$, as shown in Fig.1(b). At location $H_{loc}(u,t)$, messages reported from different reporters v_i will exhibit inconsistent location claims of u, which indicates a node replication attack on identity u.

D. SDD: Space-Domain Detection

In this section, we present our localized SDD scheme for detecting node replication attacks. This detection scheme consists of two phases: the *local check phase* and the *local witness check phase*.

1) Local Check: The basic idea of the local check procedure is sketched as follows. When two nodes v and u meet each other, they exchange information according to the local information exchange protocol elaborated in Section III-A. After v receives from u $Message_{u \to v}$: $(u||v||t_i^u||loc_i^u||ch_{i_k}^u||Sig_i^u)$, v first verifies the authenticity of this message by the public key of u. If the message fails to pass the authenticity check, v reports an external attack on u (namely the message is faked). Otherwise, v records this message in its table $table_v$, as shown in Table I, and check whether the newly received information of u is consistent with the information it has recorded for u in $table_v$. If $Message_{u \to v}$ violates any of the lemmas in Section III-B, there must be a node replication attack on identity u. This procedure is detailed in Algorithm 2.

Algorithm 2 SDD-LC: Local Check on Identity u

Input: $u, v, table_v, Message_{u \to v} : (u||v||t_i^u||loc_i^u||ch_{i_k}^u||Sig_i^u)$
Output: SDD-LC(u), the check result of identity u: 0 =normal, 1 =node replication attack detected, 2 =external attack, N/A=not available.

```
1: function SDD-LC($u$, $v$, $table_v$, $Message_{u \to v}$ : $(u||v||t_i^u||loc_i^u||ch_{i_k}^u||Sig_i^u)$)
2:     if $Sig_i^u$ is valid then
3:         $table_v \leftarrow (t_i^u||loc_i^u||ch_{i_k}^u||Sig_i^u)$    ▷ $v$ records $Message_{u \to v}$ into its table $table_v$.
4:         if $Message_{u \to v}$ is the only information $v$ has recorded for $u$ then
5:             Return N/A    ▷ The information that $v$ has recorded for $u$ is insufficient to launch the local check on $u$.
6:         else
7:             According to Lemma 3.1-3.3
8:             if $Message_{u \to v}$ is consistent with all the previous messages $v$ has recorded in its table for $u$ then
9:                 Return 0
10:            else
11:                Return 1
12:            end if
13:        end if
14:    else
15:        Return 2
16:    end if
17: end function
```

2) Local Witness Check: The local witness check procedure is based on the following observation. Given a node u and its replica u_r, u and u_r inevitably generate information contradicting Fact 3.1 and Fact 3.2 in the network. This information may be recorded by the witness nodes of u and u_r during the information exchange process stated in Section III-A. Once the witness nodes meet each other and exchange their recorded information about identity u, they may find the contradictory information breaking the lemmas in Section III-B. Thus the node replication attack on identity u can be detected.

This procedure is detailed in Algorithm 3. Note that we require any two witness nodes v_1 and v_2 randomly pick d nodes to check and exchange their recorded information about these d nodes when they meet, for the purpose of conserving communication resources. Of course v_1 and v_2 could exchange their tables such that node replica attacks can be detected at an earlier time.

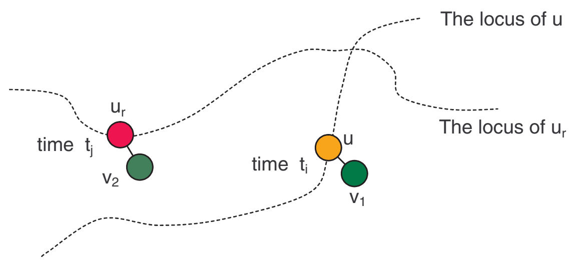
Remark: SDD can successfully detect colluding replicas of identity u when benign witness nodes v and w place a check on u since the bindings of time, location, and the corresponding released challenge among all colluding nodes can not be synchronized due to the local information exchange protocol proposed in Section III-A.

Algorithm 3 SDD-LWC: Local Witness Check on Identity u

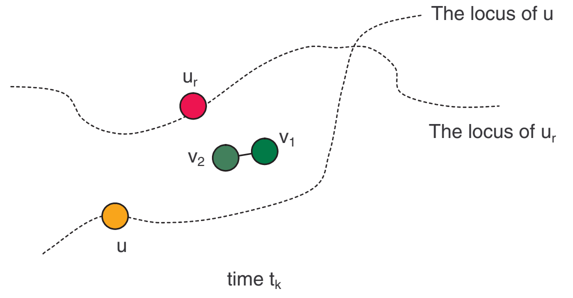
Input: witness nodes v_1, v_2, and their tables $table_{v_1}, table_{v_2}$
Output: SDD-LWC(u), the check result of identity u: 0 =normal, 1 =node replication attack detected, N/A=not available.

```
1: function SDD-LWC($v_1, v_2, table_{v_1}, table_{v_2}$)
2:     $v_1$ and $v_2$ randomly pick $d$ nodes to check
3:     if $u$ is picked then
4:         if ($table_{v_1}(u) \neq \emptyset$ and $table_{v_2}(u) \neq \emptyset$) then    ▷ The records that $v_1$ and $v_2$ have recorded for $u$, denoted by $table_{v_1}(u)$ and $table_{v_2}(u)$, respectively, are not empty.
5:             $v_1 \leftarrow table_{v_2}(u)$  ▷ $v_2$ sends the information it has recorded for $u$ to $v_1$.
6:             $v_2 \leftarrow table_{v_1}(u)$  ▷ $v_1$ sends the information it has recorded for $u$ to $v_2$.
7:             According to Lemma 3.1-3.3
8:             if the information in $table_{v_1}(u)$ and $table_{v_2}(u)$ are consistent with each other then
9:                 Return 0
10:            else
11:                Return 1
12:            end if
13:        else
14:            Return N/A    ▷ The information that $v_1$ and $v_2$ have recorded for $u$ is insufficient to launch the local witness check.
15:        end if
16:    else
17:        Return N/A   ▷ $u$ is not picked by $v_1$ and $v_2$ to launch the local witness check.
18:    end if
19: end function
```

Example: As shown in Fig.2(a), u and u_r have different locus caused by mobility. Assume u meets more number of nodes than u_r does. Therefore, at time t_i the challenge issued by u has a larger index in u's challenge chain, while at time t_j the challenge issued by u_r has a smaller index in the challenge chain, where $t_i < t_j$. Suppose v_1 (v_2) is a witness node of u (u_r), where v_1 receives $ch_{k_i}^u$ from u at time t_i, v_2 receives $ch_{k_j}^u$ from u_r at time t_j. When v_1 and v_2 meet each other, as shown in Fig.2(b), they pick u to launch the SDD-LWC check. Note that the information $(t_i, ch_{k_i}^u)$ and $(t_j, ch_{k_j}^u)$ contradict with Lemma 3.1, which indicates a node replication attack on



(a) v_1 meets u at t_i, v_2 meets u_r at t_j



(b) At time t_k, v_1 and v_2 meet each other

Fig. 2. Nodes v_1 and v_2 perform SDD-LWC check on identity u.

identity u.

IV. Analysis

A. Security Analysis

Our TDD and SDD schemes are based on the detection of the paradoxes a node u and its replicas generate in the network. In the following, we first argue that u and its replicas keep on generating *challenge paradoxes* when they move and communicate. Based on this observation, we then perform analysis on the detection accuracies of TDD and SDD.

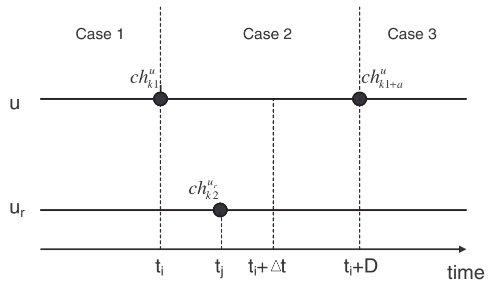


Fig. 3. Illustration of the proof of Theorem 4.1

1) Challenge Paradox: Suppose u issues $ch^u_{k_1}$ at t_i and its replica u_r issues $ch^{u_r}_{k_2}$ at t_j, where $t_i < t_j$ and $|t_j - t_i| \le D$. Denote the sets of challenges issued by u and u_r within $[t_i, t_i+D]$ and $[t_j-D, t_j]$ by $S_D(ch^u) = \{ch^u_{k_1}, ch^u_{k_1+1}, \cdots, ch^u_{k_1+a}\}$ and $S_D(ch^{u_r}) = \{ch^{u_r}_{k_2-b}, \cdots, ch^{u_r}_{k_2-1}, ch^{u_r}_{k_2}\}$, respectively. Then $a+1$ and $b+1$ represent the numbers of nodes u and u_r met in $[t_i, t_i+D]$ and $[t_j-D, t_j]$, respectively. Let Δt ($\Delta t'$) be the minimum amount of time for u (u_r) to issue the $a+1$ ($b+1$) challenges in the network. Set $\Delta = \max(\Delta t, \Delta t')$.

Theorem 4.1: Given u and its replica u_r as defined above, they always generate challenge paradoxes if $|t_j - t_i| < \Delta$.

Proof: We first prove that the claim holds if $|t_j - t_i| < \Delta t$. Given u and its challenges $\{ch^u_{k_1}, ch^u_{k_1+1}, \cdots, ch^u_{k_1+a}\}$ issued during $[t_i, t_i+D]$, there are three different cases when u_r issues its challenge $ch^{u_r}_{k_2}$ at time t_j, as shown in Fig. 3.

Case 1: $k_2 < k_1$**.**

Since $ch^{u_r}_{k_2}$ has a smaller index than $ch^u_{k_1}$ in u's challenge chain, $ch^{u_r}_{k_2}$ is supposed to be issued earlier than $ch^u_{k_1}$. However, $ch^{u_r}_{k_2}$ is issued at a later time than $ch^u_{k_1}$. According to Lemma 3.1, there must be a paradox between the two challenges $ch^{u_r}_{k_2}$ and $ch^u_{k_1}$. Here we call $ch^{u_r}_{k_2}$ an *outdated challenge*.

Case 2: $k_1 \le k_2 \le k_1 + a$**.**

In this case $ch^{u_r}_{k_2} \in S_D(ch^u)$. In other words, u and u_r issue a *duplicate challenge* within the time duration D.

However, every challenge in u's challenge chain is unique and cannot be issued twice according to the local information exchange protocol in Section III-A. Namely the challenges issued by u and u_r cannot be the same within D. Therefore, u and u_r must generate a paradox within D that contradicts Lemma 3.1.

Case 3: $k_2 > k_1 + a$**.**

Based on the definition of Δt, we can see that u cannot issue more than $a+1$ challenges within Δt time. According to our local information exchange protocol, the challenges in $S_D(ch^u)$ issued by u have to be in consecutive order in u's challenge chain. Therefore the challenge with the maximum index that u can issue during $[t_i, t_i+\Delta t]$ is $ch^u_{k_1+a}$.

Since $t_j \in (t_i, t_i+\Delta t]$, at time t_j the index of the issued challenge in u's challenge chain cannot be larger than k_1+a. However, we have $k_2 > k_1 + a$. Therefore, u and u_r must generate a challenge paradox that contradicts Lemma 3.3.

A similar discussion holds for the case when $|t_j - t_i| < \Delta t'$. Therefore, if $|t_j - t_i| < \Delta$, no matter how u and u_r issue their challenges, they always generate challenge paradoxes. ■

Corollary 4.1: Given a sufficiently large[4] interval D, there must exist paradox node pairs (v_i,v_j) when $|t_j - t_i| < \Delta$.

According to our approaches proposed in Section III-C and Section III-D, given u and its replica u_r, the detection accuracy of TDD depends on whether u and u_r generate a paradox node pair during the time $t \pm \Delta$, while the detection accuracy of SDD depends on whether a generated paradox node pair meet each other within D and launch the SDD check on u.

To facilitate our analysis, we let $\theta_u(t)$ denote the number of nodes a node u meets at the time $t \pm \Delta$, and $\lambda_1(t)$, $\lambda_2(t)$, and $\lambda_3(t)$ denote the number of challenge paradoxes incurred by u and u_r in t under Case 1, Case 2, and Case 3, respectively. According to Theorem 4.1, we have,

$$\lambda_1(t) \quad = \quad \lambda_2(t) = \theta_u(t) \qquad (4)$$

[4] D is required to be sufficiently large to guarantee that $|t_j - t_i| < \Delta$ can be easily satisfied since Δ is determined by a ad b, which are determined by D.

$$\lambda_3(t) \quad = \quad \theta_u(t)(\sum_{t_j \in [t, t+\Delta t)} \theta_{u_r}(t_j)) \tag{5}$$

Therefore, $\lambda(t)$, the total number of challenge paradox node pairs generated by u and u_r at $t \pm \Delta$ is

$$\lambda(t) = \begin{cases} \lambda_1(t) & \text{Case 1,} \\ \lambda_2(t) & \text{Case 2,} \\ \lambda_3(t) & \text{Case 3.} \end{cases} \tag{6}$$

2) Detection Accuracy of TDD: Given node u issuing a challenge at time $t \pm \Delta$, under Case 1 and Case 2, if the outdated or duplicated challenges are issued by u_r within $t \pm \Delta$, TDD can certainly reveal the paradoxes and detect the replica attacks. If not, we expect that SDD detect these replica attacks. In the following, we analyze the detection accuracy of TDD under Case 3.

At each time interval T_i of the trusted node TN, the probability that TDD detects a paradox is

$$P_{TDD}(u, u_r, T_i) = \min(\lambda_3(t), 1),$$

where $t \in T_i$. Suppose the time duration D consists of K time intervals, denoted by $T_{i_1}, T_{i_2}, \cdots, T_{i_K}$, we have

$$P_{TDD}(u, u_r) \quad = \quad 1 - \prod_{j=1}^{K}(1 - P_{TDD}(u, u_r, T_{i_j}))$$

where $P_{TDD}(u, u_r)$ is the probability that TDD detects a paradox generated by u and u_r within D.

If there are m replicas of u, the detection accuracy of TDD will be

$$\begin{aligned} P_{TDD}(u) \quad = \quad & 1 - \prod_{1 \le j_1, j_2 \le m, j_1 \ne j_2} (1 - P_{TDD}(u_{j_1}, u_{j_2})) \\ & \cdot \prod_{i=1}^{m}(1 - P_{TDD}(u, u_i)) \end{aligned} \tag{7}$$

Remark: Given u, its replica u_r, and a sufficiently large duration D, there always exist challenge paradoxes generated by u and u_r in $t \pm \Delta$. Therefore, our TDD scheme under Case 3 detects the replicas at 100% accuracy.

3) Detection Accuracy of SDD: Suppose v and w are a paradox node pair of u. When v and w met each other within D, they launch the SDD-LC check on u if $u \in \{v, w\}$, or randomly pick d identities from the other $N-2$ nodes in the network for the SDD-LWC check if $u \notin \{v, w\}$. For the first case, the probability $p(u \in \{v, w\})$ that u is checked by SDD-LC is $\frac{2}{N}$. For the second case, the probability that u is checked by the SDD-LWC procedure is $p(u \text{ is selected}, u \notin \{v, w\})$, which can be computed by the conditional probability

$$\begin{aligned} & p(u \text{ is selected}, u \notin \{v, w\}) \\ = \quad & p(u \text{ is selected} | u \notin \{v, w\}) \times p(u \notin \{v, w\}) \\ = \quad & \frac{d}{N-2} \times \frac{N-2}{N} = \frac{d}{N} \end{aligned} \tag{8}$$

Therefore, the probability that u is checked by SDD is $\frac{2}{N} + \frac{d}{N} = \frac{d+2}{N}$.

Let α denote the probability that two nodes meet each other within D. Then the probability $p_{(v,w)}(u, u_r)$ that the paradox node pair (v, w) meet each other within D and launch the SDD check on u becomes

$$p_{(v,w)}(u, u_r) = \frac{\alpha(d+2)}{N}. \tag{9}$$

Let q denote the total number of paradox node pairs generated by u and u_r within D. According to Eq. (6), we have

$$q = \sum_{t \in D} \lambda(t). \tag{10}$$

Then Eq. (9) and Eq. (10) yield

$$P_{SDD}(u, u_r) \quad = \quad 1 - \prod_{i=1}^{q}(1 - p_{(v_i, w_i)}(u, u_r)), \tag{11}$$

where $P_{SDD}(u, u_r)$ denotes the probability that SDD detects a paradox generated by u and u_r during D.

If there are m replicas of u, the detection accuracy of SDD will be

$$\begin{aligned} P_{SDD}(u) \quad = \quad & 1 - \prod_{1 \le j_1, j_2 \le m, j_1 \ne j_2} (1 - P_{SDD}(u_{j_1}, u_{j_2})) \\ & \cdot \prod_{i=1}^{m}(1 - P_{SDD}(u, u_i)). \end{aligned} \tag{12}$$

Remark: According to Theorem 4.1 and Corollary 4.1, given a sufficiently large D, q, the number of paradox node pairs generated by u and u_r, could be large enough such that the detection accuracy $P_{SDD}(u, u_r)$ of SDD approaches to 100%.

B. Efficiency Analysis

In this section, we evaluate the performance of our detection schemes in terms of communication/storage/computation overhead, and conduct a brief comparison study with the most related work.

When checking identity u, TDD simply asks the witness nodes to report to some location $H_{loc}(u, t)$ the messages received from u during the time interval $t \pm \Delta$. Since the number of nodes reporting to location $H_{loc}(u, t)$ can be characterized by $\sum_{t_i \in t \pm \Delta} \theta_u(t_i)$, the computation overhead of launching TDD check on u is $O(\sum_{t_i \in t \pm \Delta} \theta_u(t_i))$. As $H_{loc}(u, t)$ is a random location in the network, the communication cost of TDD is $O(\sqrt{N})$. SDD-LC is a pure localized procedure that incurs communication and computation overheads of $O(1)$. In SDD-LWC, d number of nodes are randomly selected for paradox check and therefore the communication and computation overheads of SDD-LWC are $O(d)$. The storage overhead of TDD and SDD is determined by Table I created by the local information exchange protocol. Since D is usually large, in the worst case a node could meet all others nodes in the network within $2D$. Therefore, the storage overheads of TDD and SDD are $O(N)$.

A comparison study with XED [18] and SPRT [19], the two existing replication detection schemes proposed for mobile networks in literature, is reported in Table II. We conclude that

Schemes	Communication Cost	Storage Cost	Computation Cost
XED [18]	$O(1)$	$O(N)$	$O(1)$
SPRT [19]	$O(\sqrt{N})$	$O(1)$	$O(1)$
TDD	$O(\sqrt{N})$	$O(N)$	$O(\sum_{t_i \in t \pm \Delta} \theta_u(t_i))$
SDD-LC	$O(1)$	$O(N)$	$O(1)$
SDD-LWC	$O(d)$	$O(N)$	$O(d)$

TABLE II
COMMUNICATION COST, STORAGE COST, AND COMPUTATION COST OF DETECTING A REPLICA DURING A DETECTION PROCESS.

TDD and SDD maintain a comparable amount of communication/storage/computation overhead while achieving superior performance in detection accuracy even when clones collude.

V. CONCLUSION AND FUTURE WORK

Previous node replication detection schemes depend primarily on a stationary network model, and/or on the assumption of limited replicas/compromised nodes without collusion. To address these fundamental limitations, we provide a novel approach to detecting node replication attacks in MANETs. Specifically, we propose two novel schemes, Time-Domain Detection and Space-Domain Detection, based on a one-way hash function. Our analysis indicates that these schemes provide a high detection accuracy disregarding node collusion and the number/distribution of replicas/compromised nodes. Moreover, these schemes are naturally extensible to other classes of mobile networks in which nodes can be physically captured and replicated by adversaries. For future work, we are going to perform further study on the overhead analysis and the impact of the parameter settings on the performance of the proposed schemes.

ACKNOWLEDGMENTS

This research was supported by the US National Science Foundation under grant CNS-0721669 and the National Natural Science Foundation of China under grant 2006CB303006.

REFERENCES

[1] V. Gligor, "Handling new adversaries in secure mobile ad-hoc networks," in *ESNS*, 2007.
[2] Y. chun Hu, A. Perrig, and D. B. Johnson, "Wormhole detection in wireless ad hoc networks," Tech. Rep., 2002.
[3] N. Song, L. Qian, and X. Li, "Wormhole attacks detection in wireless ad hoc networks: A statistical analysis approach," in *IEEE IPDPS'05 - Workshop 17*, 2005, p. 289.1.
[4] F. Liu, X. Cheng, and D. Chen, "Insider attacker detection in wireless sensor networks," in *IEEE INFOCOM'07*, 2007, pp. 1937–1945.
[5] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," in *ACM SASN'04*, 2004, pp. 66 – 77.
[6] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *INFOCOM*, 2003, pp. 1976 – 1986.
[7] B. Przydatek, D. Song, and A. Perrig, "Sia: secure information aggregation in sensor networks," in *SenSys'03*, 2003, pp. 255–265.
[8] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Sdap: a secure hop-by-hop data aggregation protocol for sensor networks," in *MobiHoc'06*, 2006, pp. 356–367.
[9] S. Capkun and J.-P. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *IEEE INFOCOM'05*, 2005, pp. 1917 – 1928.
[10] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *IEEE SP'05*, 2005, pp. 49–63.
[11] H. Choi, S. Zhu, and T. Laporta, "Set: Detecting node clones in sensor networks," in *SecureComm'07*, 2007, pp. 341–350.
[12] B. Zhu, V. G. K. Addada, S. Setia, S. Jajodia, and S. Roy, "Efficient distributed detection of node replication attacks in sensor networks." in *ACSAC*, 2007, pp. 257–267.
[13] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei, "A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks," in *MobiHoc'07*, 2007, pp. 80–89.
[14] K. Xing, F. Liu, X. Cheng, and D. H. C. Du, "Real-time detection of clone attacks in wireless sensor networks," in *IEEE ICDCS'08*, 2008, pp. 3–10.
[15] R. R. Brooks, P. Y. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M. T. Kandemir, "On the detection of clones in sensor networks using random key predistribution," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 37, no. 6, pp. 1246–1258, 2007.
[16] P. Kyasanur and N. H. Vaidya, "Detection and handling of mac layer misbehavior in wireless networks," in *IEEE DSN*, 2002, pp. 173 – 182.
[17] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis defenses james newsome," in *IPSN'04*, 2004, pp. 259–268.
[18] C.-M. Yu, C.-S. Lu, and S.-Y. Kuo, "Mobile sensor network resilient against node replication attacks," in *IEEE SECON'08*, vol. 16, 2008, pp. 597 – 599.
[19] J.-W. Ho, M. Wright, and S. Das, "Fast detection of replica node attacks in mobile sensor networks using sequential analysis," in *IEEE INFOCOM*, 2009, pp. 1773 – 1781.
[20] D. Shaw and W. Kinsner, "Multifractal modelling of radio transmitter transients for classification," in *WESCANEX 97: IEEE Conference on Communications, Power and Computing*, 1997, pp. 306–312.
[21] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *IEEE INFOCOM*, vol. 2, 2005, pp. 902– 913.
[22] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, 2002.
[23] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *IEEE SECON'04*, 2004, pp. 71–80.
[24] G. Gaubatz, J. peter Kaps, and B. Sunar, "Public key cryptography in sensor networks - revisited," in *ESAS'04*, 2004, pp. 2–18.
[25] B. Hoffman-Wellenhof, H. Lichteneeger, and J. Collins, *Global Positioning System: Theory and Practice (4th ed.)*. New York: Springer-Verlag, 1991.
[26] J. Yi, S. Yang, and H. Cha, "Multi-hop-based monte carlo localization for mobile sensor networks," in *SECON'07*, 2007, pp. 162–171.
[27] J. Koo, J. Yi, and H. Cha, "Localization in mobile ad hoc networks using cumulative route information," in *UbiComp'08*, 2008, pp. 124–133.
[28] L. Hu and D. Evans, "Localization for mobile sensor networks," in *MobiCom'04*, 2004, pp. 45–57.
[29] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *MobiCom'01*, 2001, pp. 166–179.
[30] X. Cheng, A. Thaeler, G. Xue, and D. Chen, "Tps: A time-based positioning scheme for outdoor sensor networks," in *IEEE INFOCOM*, vol. 4, 2004, pp. 2685– 2696.
[31] F. Liu, X. Cheng, D. Hua, , and D. Chen, "Tpss: A time-based positioning scheme for sensor networks with short range beacons," in *ICCNMC'05*, 2005, pp. 33–42.
[32] A. Thaeler, M. Ding, and X. Cheng, "itps: An improved location discovery scheme for sensor networks with long range beacons," *Journal of Parallel and Distributed Computing*, vol. 65, no. 2, pp. 98–106, 2005.
[33] K. Sun, P. Ning, and C. Wang, "Tinysersync: secure and resilient time synchronization in wireless sensor networks," in *ACM CCS'06*, 2006, pp. 264–277.
[34] L. Chen and J. Leneutre, "A secure and scalable time synchronization protocol in ieee 802.11 ad hoc networks," in *ICPPW'06*, 2006, pp. 207–214.
[35] K. Römer, "Time synchronization in ad hoc networks," in *MobiHoc'01*, 2001, pp. 173–182.
[36] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *MobiCom'00*, 2000, pp. 255–265.