Defend Against Cache Consistency Attacks in Wireless Ad Hoc Networks *

Wensheng Zhang and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
Email: {wezhang, gcao}@cse.psu.edu

Abstract

Caching techniques can be used to reduce bandwidth consumption and data access delay in wireless ad hoc networks. When cache is used, cache consistency issues must be addressed. To maintain strong cache consistency in some strategic scenarios (e.g., battle fields), the invalidation-based approach is preferred due to its low overhead. However, this approach may suffer from some security attacks. For example, a malicious node (intruder) may drop, insert or modify invalidation messages to mislead the receivers to use stale data or unnecessarily invalidate the data that is still valid. In this paper, we propose a solution based on the IR-based cache invalidation strategy to prevent intruders from dropping or modifying the invalidation messages. Although digital signatures can be used to protect IRs, it has significantly high overhead in terms of computation and bandwidth consumption. To address this problem, we propose a family of randomized grouping based schemes for intrusion detection and damage recovery. Extensive analysis and simulations are used to evaluate the proposed schemes. The results show that our solution can achieve a good level of security with low overhead.

1 Introduction

In wireless ad hoc networks, nodes communicate with each other using multi-hop wireless links. Due to lack of infrastructure support, each node acts as a router, forwarding data packets for other nodes. Most of the previous research in ad hoc networks focuses on the development of dynamic routing protocols [6, 12, 15] that can efficiently find routes between two communicating nodes. Although routing is an important issue in ad hoc networks, other issues such as information (data) access are also very important since the ultimate goal of using ad hoc networks is to provide information access to mobile nodes.

Caching frequently accessed data items at the client side is an effective technique to improve performance in wireless networks. With caching, both bandwidth consumption and data access delay are reduced since some data access requests can be served from the local cache, thereby obviating the need for data transmission over the scarce wireless links. Figure 1 shows an example in a battlefield, where the communication equipment held by a commander and a group of soldiers form an ad hoc network. The commander has a data center, and the soldiers need to access the data center to get information about the enemy, the battlefield, and the attack plans. After a soldier obtained the information from the data center, other soldiers around him may also need to access the information. If the soldier can cache a copy of the data locally and use it to serve the requests of other soldiers. These soldiers will not need to send request and get reply from the faraway data center, and hence save bandwidth and reduce the access delay. Techniques to achieve this has been discussed in [5].

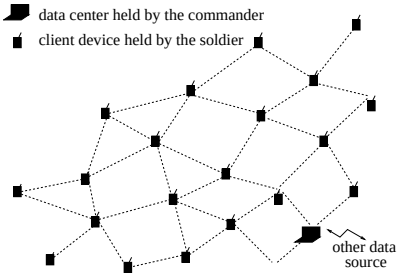


Figure 1. An ad hoc network in the battlefield

To use cache, the mobile nodes have to make sure that the cached data is consistent with the data at the data center. Problems related to cache consistency have been studied in many other systems such as multi-processor architectures, distributed file systems, distributed shared memory,

*This work was supported in part by the National Science Foundation (CAREER CCR-0092770 and ITR-0219711).

and database systems. Two widely used cache consistency models are the weak consistency model and the strong consistency model. In the weak consistency model, stale data might be returned to the client. In the strong consistency model, after a write completes, no stale copy of the modified data will be returned to the client. In some adversary and strategic scenarios such as in the battlefield (Figure 1), accessing stale data (e.g., outdated enemy information) may be *life threatening*, and hence we need to study how to achieve strong consistency.

For strong cache consistency, the *polling-based* approach can be used. In this approach, every time the user requests a data item and there is a cached copy, the cache first contacts the server to validate the cached copy, then returns the valid copy to the user. However, in a large network, using the polling-based approach may generate significant network traffic [4, 23], since a large number of clients need to frequently contact the server to validate their cached data items. To address the problem, the *invalidation-based* approach is widely used. In the *invalidation-based* approach, the server keeps track of the clients that cache the data, and sends invalidation messages to the clients when the data is changed.

In the hostile scenarios such as in the battlefield, some nodes in the network may be malicious; the adversary may also capture and compromise some nodes, and make use of the compromised nodes to launch various kinds of attacks on the invalidation-based approach. Basically, there are two kinds of attacks: First, the compromised nodes (intruders) may stop propagating the invalidation messages, and nodes far away from the data source may not be able to receive the invalidation message and may use the stale cache (or replica) without realizing it. Second, an intruder may inject a false invalidation message, or modify a passing invalidation message, to mislead the receivers to use their stale caches or invalidate their caches which are still valid.

In this paper, we propose solutions to deal with the above attacks on cache consistency. To prevent malicious nodes from dropping the invalidation messages, we borrow ideas from the IR-based cache invalidation [1, 3]. In this approach, the server periodically broadcasts an *invalidation report (IR)* in which the changed data items are indicated. Since IRs are sent out regularly, the clients expect the IR at regular time interval. If a client maliciously drops an IR, the nodes that are expecting the IR can detect it, and some measure can be taken to address the attack. To prevent malicious nodes from modifying the IRs, digital signature can be used. To reduce the high overhead associated with the digital signature approach, we propose a family of randomized grouping-based techniques for intrusion detection and intrusion recovery. Extensive analysis and simulations are used to evaluate the proposed schemes. The results show that our solution can achieve good level of security with low overhead.

The rest of the paper is organized as follows: Section 2 briefly presents the related work. Section 3 describes the system model. Section 4 proposes solutions to defend against dropping invalidation messages. Section 5 proposes randomized grouping techniques to reduce the authentication overhead. Security analysis of the proposed solution is presented in Section 6. Section 7 reports the simulation results and Section 8 concludes the paper.

2 Related Work

Recently, many researchers have studied security issues in ad hoc networks. Hubaux *et al.* [11] addressed the issues of distributing public keys in ad hoc networks, by proposing to let users issue certificates for each other based on their personal acquaintances. Zhou and Haas [26] proposed a solution based on threshold cryptography. Based on a trusted certificate authority, Sanzgiri *et al.* [21] proposed a solution to secure routing protocols in ad hoc wireless networks. To address the high overhead associated with obtaining and verifying the digital certificates, Hu *et al.* proposed a protocol [9] to secure on-demand routing protocols based on TESLA [18], an efficient broadcast authentication scheme that requires loose time synchronization. Our work also aims to reduce the high overhead of digital signatures. But our solution is different from TESLA in that it introduces very small verification delay.

Even with secure routing protocols, the source/destination may still setup a route which goes through a misbehaving (malicious) node that agrees to forward packets but fails to do so. Marti *et al.* [17] proposed to use a *watchdog* entity to identify misbehaving nodes and a *pathrater* mechanism to avoid routing packets through misbehaving nodes. However, misbehaving nodes are not punished, and thus there is no motivation for the nodes to cooperate. To overcome this problem, Buchegger and Le Boudec [2] defined protocols that are based on a reputation system. In our intrusion detection scheme, nodes do not need to use promiscuous mode to monitor the transmissions of their neighbors. Instead, the message authentication codes generated using keys are checked to identify intrusions.

Other researchers in this field also address the problem of intrusion detection [25] and MAC layer misbehavior [14]. There are also researches on securing sensor networks [7, 27, 16, 24]. To our knowledge, there is no existing work addressing the problem of attacks on cache consistency in ad hoc networks, which is the major goal of this paper.

3 System Model

We consider a wireless ad hoc network, as shown in Figure 1, which consists of a data center and many ordinary nodes. The data center (also called *server*) stores data that is updated now and then. Some ordinary nodes (called *clients*) frequently access the data, and cache some data locally to reduce network traffic and data access delay. We assume strong cache consistency is required, and the invalidation-based cache consistency model is used.

In the invalidation-based scheme, the server needs to send invalidation messages to clients. The most reliable method to ensure that all clients with cached data receive the invalidation messages is to use flooding, which has very high overhead. Another option is to use multicast by setting up a multicast group (tree) [20], where the server is the root of the tree. With this approach, a node can find out whether its local cache is valid or not. If not, the node has to send a request to the data center to ask for the updated data. If the data will be accessed by many nodes, this approach not only increases the access delay, but also creates a large amount of network traffic. To further improve performance, the multicast tree can also be used to push updates. Nodes receiving these updates do not need to send uplink requests and can reduce the access delay and bandwidth consumption. However, if no node is interested in this data update, pushing data down only consumes extra bandwidth. Techniques [20] exist to identify frequently accessed data whose updates should be pushed. It is easy to see that the multicast tree can be used for pushing cache invalidation and frequently accessed data to improve the system performance. In this paper, we assume that a multicast tree has been built due to the aforementioned reasons. Some existing techniques [8, 13, 15, 22] can be used to set up and maintain the multicast tree in ad hoc networks.

We consider two types of attacks that a malicious node can launch on the invalidation-based cache consistency scheme:

- The node may drop some invalidation messages, such that its descendants can not receive the message and hence may unknowingly use stale cached data.
- The node may modify some invalidation messages that it forwards, such that its descendants may receive wrong invalidation messages and hence may unknowingly use stale cached data or unnecessarily invalidate cached data.

4 Defending Against Dropping Invalidation Messages

To prevent malicious nodes from dropping invalidation messages, we borrow ideas from the IR-based cache invalidation [1, 3] scheme. In this approach, the server periodically broadcasts an *invalidation report (IR)* in which the changed data items are indicated. The IR consists of the current timestamp T_i and a list of tuples (d_x, t_x) such that $t_x > (T_i - w \times L)$, where d_x is the data item id, t_x is the most recent update timestamp of d_x, L is the length of the invalidation broadcast interval, and w is the invalidation broadcast window size. In other words, IR contains the update history of the past w broadcast intervals. Based on the value of w, clients can still validate their local cache even after missing $w-1$ IRs. Similar to the original invalidation-based approach, clients use the invalidation message (IR) to invalidate their local cache. Different from the original invalidation-based approach, the IR is sent out regularly, and the clients expect the IR at regular time interval. Therefore, if a malicious node drops an IR, its descendant nodes can detect it. A node may also miss an IR due to multicast tree partition caused by node movement or node failure. In either case, the node will re-join the multicast tree using some existing multicasting protocols [8, 13, 22], and then get the missed IR.

5 Reducing the Authentication Overhead by Randomized Grouping

To prevent a malicious node from modifying an IR, each IR should be authenticated, and digital signatures can be used for authentication. However, this approach has high overhead both in terms of computation and bandwidth [10]. To address this problem, symmetric cryptographic techniques such as TESLA [18] may be used. TESLA provides source authentication with *message authentication codes (MACs)* using only symmetric cryptography, based on delayed disclosure of keys by the sender. However, when applying TESLA to a large ad hoc network, the sender discloses a key to authenticate a previously sent packet, only after the furthest node has already received the packet. Due to the large authentication delay, TESLA is not suitable for authenticating IRs. In this section, we present a novel solution to reduce the authentication overhead by randomized grouping.

5.1 The Basic Idea of Randomized Grouping

In the proposed solution, the nodes (IR receivers) are randomly distributed into multiple groups, and the data center (server)[1] shares a unique group key with the receivers of each group. Before the nodes are deployed, the server (denoted as N_0) randomly picks the following keys:

[1] To simplify presentation, we assume that there is only one server. The solution can certainly be extended to the case of multiple servers.

- (P_0^+, P_0^-): P_0^+ is its public key, and P_0^- is its private key.
- $\{K_i\}_{i=0,1,\cdots,M-1}$: K_i is the group key shared by the server and the receivers in group i, where M is the number of groups.

For each trusted node N_i ($i = 1, 2, \cdots$) that is allowed to join the network, it is first randomly assigned to one of the M receiver groups. Then, it is preloaded with the following keys:

- P_0^+: the public key of the server.
- (P_i^+, P_i^-): P_i^+ and P_i^- are the public and private keys of N_i. When N_i disseminates its public key to others, it should show the *certificate* issued by the server, i.e., $P_0^-\{P_i^+\}$, which can be verified by other nodes using P_0^+.
- $K_{g(i)}$: the group key shared by N_i and the server, where $g(i)$ is the group id of N_i.

When the server sends out an IR, the IR is protected by several MACs, each with one group key. Since each receiver only knows one of the keys, if an intruder modifies the IR and the MAC using the group key it knows, the modification can be detected by a descendant in a different receiver group.

For example, as shown in Figure 2 (a), N_0 is the server and two group keys K_0 and K_1 are used. N_0 constructs a *secure IR (SIR)* by appending two MACs to the IR, i.e., $SIR = \langle IR, MAC_0, MAC_1\rangle$, where $MAC_i = C_{K_i}(IR), i = 0, 1$, and C is a MAC function. Then, it sends the SIR to N_1, which forwards it to N_2 and N_3. If N_1 and N_3 know K_0 while N_2 knows K_1, malicious modifications can be easily identified. For example, if N_2 is a malicious node and it modifies the IR, N_3 will be able to detect this modification by verifying MAC_0. After N_3 reports this modification to the server, the server can easily find that N_2 is the malicious node.

In the ideal case where neighbor nodes use different keys, the malicious node can be easily identified. However, if two neighbor nodes use the same group key, further isolation may be difficult. For example, if both N_1 and N_2 use K_0, it will be difficult for the server to find out who is the malicious node. Further, if the malicious nodes can collude, a malicious node may know more than one key; i.e., they can share their group keys with each other. In the following, we propose solutions to deal with these cases.

5.2 Intrusion Detection

Suppose M_c group keys $(K_1, K_2, \cdots, K_{M_c})$ have been compromised. When an intruder receives a SIR, it may modify the IR. Certainly, it also needs to adjust the associated MACs in order not to be detected. Since it only knows some group keys, it can only modify some MACs correctly. Specifically, the modified SIR (SIR') can be as follows:

$$\langle IR', MAC'_1, \cdots, MAC'_{M_c}, MAC_{M_c+1}, \cdots, MAC_M\rangle,$$

where

$$MAC'_i = C_{K_i}(IR'), i = 1, 2, \cdots, M_c.$$

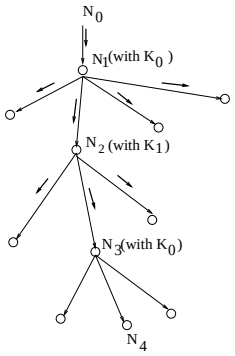
The intruder forwards the SIR' to its children. On receiving the message, each node (with group key K_j) checks the integrity of the message, by comparing MAC_j in the message to the MAC computed by itself (i.e., $C_{K_j}(IR)$). If they are different, the node immediately knows that one of its ancestors in the multicast tree is the intruder. Otherwise, the node can not detect the intrusion, and forwards the message to its children. We call a node *detector* if the node can detect a modified SIR it receives; and call a node *victim* if the node receives a compromised SIR but can not detect it. Figure 2 (b) shows an example, where group keys K_1 and K_2 are compromised, and intruder N_1 modifies the SIR with these keys. The modification is not detected by descendant N_2 whose group keys have been compromised. However, it can be detected by N_3 whose group key has not been compromised. Therefore, N_2 is a victim and N_3 is a detector.

5.3 Intrusion Recovery

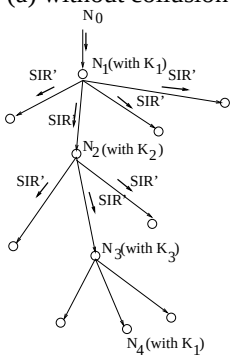
In this subsection, we first present the intrusion recovery scheme under the assumption that the intrusion detector is innocent. Then, we consider the cases where the intrusion detector is malicious, and show that the intrusion recovery scheme can also deal with these cases. Finally, the limitation of the scheme is discussed.

5.3.1 The Intrusion Recovery Scheme

When an innocent node detects an intrusion, the detector sends to the server a *SIR_Req* message which includes the following information: the received SIR, an accusation to its parent, and a request for the correct SIR. Since the message may go through a malicious node which can modify its content, the message should be signed using the private key (i.e., P_i^-, where i is the ID of the detector) of the detector. Certainly, the malicious node may drop the message. In this case, the detector has to find other routes and send the message again, until it receives a reply from the server. If going through the malicious node is the only way to reach the server, it will be similar to the problem of network partition, and the node must be aware that the cached data may be stale.



(a) without collusion



(b) with collusion

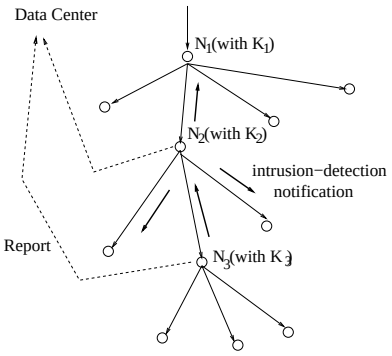
Figure 2. Intrusion detection

On receiving a request for a correct SIR, the server replies the SIR which is encrypted using its private key (P_0^-) so that other nodes cannot modify the messages. Such a special SIR is referred to as *heavy-SIR (hSIR)*. When the detector receives the hSIR, it decrypts the message, reconstructs a correct SIR, and forwards the SIR to its children.

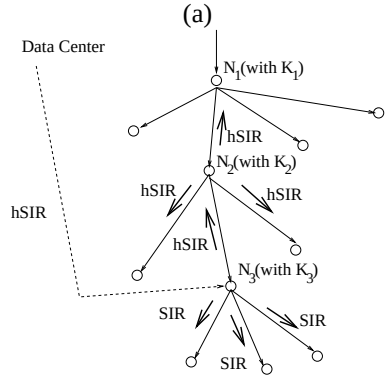
With the above process, the victims (e.g., N_2 in Figure 2 (b)) still cannot recover. To address the problem, the detector should send an *intrusion detection notification* to its parent. The notification messages includes the SIR it received and the position of the incorrect MAC. The notification should also be signed using the private key of the detector to avoid an intruder from impersonating other nodes. On receiving a notification from its child, the receiver, if it is innocent, responds as follows:

(R1) If the SIR included in the notification is different from the SIR it previously sent to the child, the receiver immediately realizes that the child changed the SIR and is an intruder. So, it sends a report to the server to accuse the child.

(R2) If the "incorrect MAC" claimed in the notification is actually correct, i.e., the detector maliciously generates a false alarm, the receiver sends a report to the server to accuse the detector, with the received notification as an evidence.



(a)



(b)

Figure 3. Intrusion recovery. (a) The detector (N_3) reports to the data center, and sends intrusion detection notifications to the victims. (b) After receiving the hSIR from the server, N_3 sends the correct SIR to its children and forwards the hSIR to the victims.

(R3) If the notification passes the above tests, the receiver propagates the notification to its parent and other children, since they may also be victims. Also, the receiver sends to the server the following information: the received notification, and an accusation to its parent. The above information may be used for the intruder identification purpose. The recursive process continues until it reaches an intruder, which may not want to propagate the notification, or a node that has already known the intrusion from other detectors.

After the detector receives a hSIR from the server, the hSIR is forwarded to the victims in the same way as the intrusion detection notification. Figure 3 illustrates an execution of the recovery scheme.

For the purpose of damage recovery, a client should wait for some time (called *guarding delay*) before using the received SIR. With this delay, if a descendant of the client finds that the SIR has been compromised, and its intrusion detection notification reaches the client before the guarding

delay expires, the client can avoid using the compromised SIR. Certainly, using the guarding delay increases the data access delay. However, based on the analytical results (Section 6) and the simulation results (Section 7), maintaining a good level of security only needs a *very short guarding delay*, and hence the additional data access delay is very small.

5.3.2 Dealing with Malicious Detectors

If a node maliciously initiates an intrusion recovery process, it must be one of the following two cases:

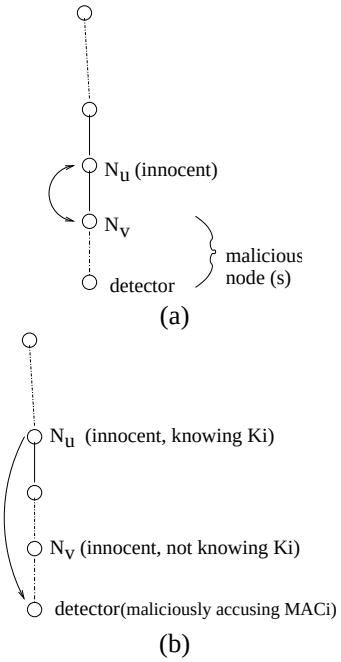


Figure 4. Dealing with Malicious Detectors

Case 1: As shown in Figure 4 (a), the malicious node modifies a received SIR, and sends out a false intrusion detection notification to its upstream nodes. According to (R1), the first innocent node on the path from the detector to the server, i.e., N_u, can detect the modification and stop propagating the intrusion detection notification. Therefore, no innocent node is affected by the attack.

Case 2: As shown in Figure 4(b), the malicious node does not modify an SIR, but claims that MAC_i in the SIR is modified by its upstream nodes. According to (R2), N_u, which is the first innocent node on the path from the malicious node to the server that knows K_i, can detect the attack. N_u will stop propagating the notification. Furthermore, according to (R3), the first innocent node on the path from the malicious detector to the server, i.e., N_v, sends the notification (issued by the malicious detector) to the server. The server can thus identify the malicious detector, and sends a hSIR to N_v to rescue the innocent nodes between N_v and N_u.

5.3.3 Limitation of the Scheme

Even with a guarding delay, the damage recovery scheme can not rescue all victims. Some victims may still use a modified SIR without knowing it. This happens when the modified SIR has not been detected before this node receives it, and no other node detects it later; or even a descendant detects the intrusion, this node can not receive the intrusion detection notification from the detector, since there is another intruder on the path connecting them. We call such a victim *cache consistency victim (cc-victim)*. Later, we will evaluate the probability for a node to be a cc-victim through analysis and simulations. The results show that the probability is low when the number of groups is large or the number of intrusion is small.

5.4 Other Security Issues

The randomized grouping-based schemes are designed to defend against attacks on cache consistency, and do not aim to address all types of attacks a compromised node may launch, such as continuously injecting packets to jam the channel or arbitrarily dropping passing packets. Many existing techniques [26, 14, 11, 17, 2] can be used to defend against these attacks.

To attack cache consistency, an intruder may also modify the updated data sent by the server, and mislead the receivers to use and cache false data. We can use the randomized grouping-based schemes to deal with these attacks. If the data is not frequently updated, we can also use digital signatures to achieve higher level of security, without significantly increasing the overhead.

6 Security Analysis

In this section, we use the metric of *cc-victim probability* (i.e., the probability that a node becomes a cc-victim) to show the security level of the proposed schemes.

6.1 Preliminaries

6.1.1 Notations and Assumptions

In our analysis, we use the following notations and assumptions:

- N, N_c, M_c: N is the number of nodes on the tree; N_c is the number of compromised nodes; M_c is the number of compromised groups.

- M: the number of groups. We assume that the receivers are uniformly distributed to these groups; i.e., the probability for a receiver to be in group i ($i = 0, 1, \cdots, M-1$) is $\frac{1}{M}$.
- d, H: For simplicity, we assume that the sender and receivers form a d-ary tree; i.e., each non-leaf node has exactly d children. Also, the tree is full and the height of the tree is H. Thus, $N = \sum_{i=1}^{H-1} d^i = d^H - 2$.

6.1.2 Distribution of M_c

Given the number of groups (M) and the number of intruders (N_c), how many group keys may be compromised is highly related to the level of security achieved. We now compute the distribution of the number of compromised keys (M_c); i.e., $P(M_c = i \mid M, N_c)$ ($M > 0, N_c > 0$) for $i = 1, 2, \cdots, min(M, N_c)$.

Let $A(m, n_c, i)$ be the total number of different options that assign n_c nodes to i groups, where the number of groups is m. So,

$$A(m, n_c, i) = \begin{cases} 0, & n_c < i; \\ m, & n_c \geq i \wedge i = 1; \\ \sum_{j=1}^{n_c} [\binom{n_c}{j} A(m-1, n_c - j, i-1)] \\ +A(m-1, n_c, i), & \text{otherwise.} \end{cases} \quad (1)$$

The total number of options for assigning n_c nodes to at most m groups is m^{n_c}, so

$$P(M_c = i \mid M, N_c) = \frac{A(M, N_c, i)}{M^{N_c}} \quad (2)$$

and

$$E(M_c \mid M, N_c) = \sum_{i=1}^{M} [i \times P(M_c = i \mid M, N_c)] \quad (3)$$

Since M and N_c are constant in the rest of this section, we use $P(M_c)$ (or $E(M_c)$) instead of $P(M_c \mid M, N)$ (or $E(M_c \mid M, N)$).

6.2 The cc-victim Probability

In this section, we introduce the sufficient and necessary condition for a node to be a cc-victim. Based on the condition, we derive the cc-victim probability. Before this, we need to introduce several definitions as follows:

Definition 1 (down hijacked) *A node is down hijacked if the group key it holds is known to an intruder and it satisfies at least one of the following conditions: (1) It is an intruder; (2) It is a leaf node; (3) All its children are down hijacked.*

Definition 2 (up hijacked) *A node is up hijacked if the group key it holds is known to an intruder and it satisfies at least one of the following conditions: (1) It is an intruder; (2) Its parent is an intruder; (3) Its parent is up hijacked and the other children of its parent are down hijacked.*

Definition 3 (fully hijacked) *A node is fully hijacked if it is both down hijacked and up hijacked.*

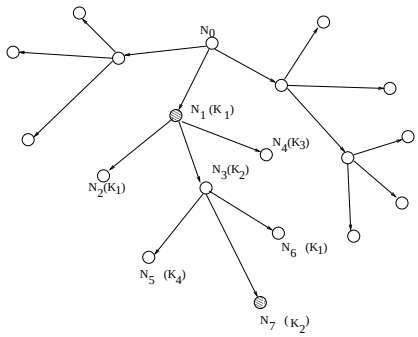


Figure 5. Up hijacked, down hijacked and fully hijacked nodes

Figure 5 shows a multicast tree. In this tree, node N_1 and N_7 are the intruders, and their group keys K_1 and K_2 are the compromised keys. Among the innocent nodes ($N_2, N_3, \cdots, N_6$), N_2 and N_6 are down hijacked, N_2 and N_3 are up hijacked, and only N_2 is fully hijacked.

Based on the previous definitions, we can obtain Theorem 1, which is presented as follows. Due to space limit, we do not show the proof.

Theorem 1 *A node is a cc-victim if and only if it is fully hijacked.*

For a l-level innocent node, let the probability that it is down hijacked, up hijacked and fully hijacked be $P_{h,d}(l \mid M_c)$, $P_{h,u}(l \mid M_c)$ and $P_{h,f}(l \mid M_c)$, respectively.

According to Definition 1,

$$P_{h,d}(l \mid M_c) = \begin{cases} \frac{M_c}{M}, & l = H-1; \\ \frac{M_c}{M}[\alpha + (1-\alpha)P_{h,d}(l+1 \mid M_c)]^d, \\ & 0 \leq l < H-1. \end{cases} \quad (4)$$

According to Definition 2,

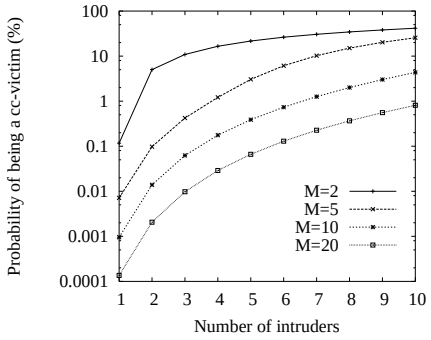
$$P_{h,u}(l \mid M_c) = \begin{cases} 0, & l = 0, 1; \\ \frac{M_c}{M}[\alpha + (1-\alpha)P_{h,u}(l-1 \mid M_c)] \\ [\alpha + (1-\alpha)P_{h,d}(l \mid M_c)]^{d-1}, \\ & 2 \leq l \leq H-1. \end{cases} \quad (5)$$

According to Definition 3,

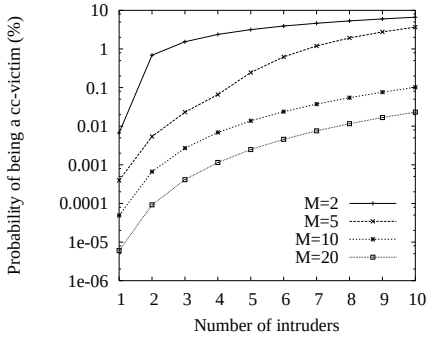
$$P_{h,f}(l \mid M_c) = P_{h,d}(l \mid M_c) \times P_{h,u}(l \mid M_c) \quad (6)$$

Thus, the probability that a node is a cc-victim is:

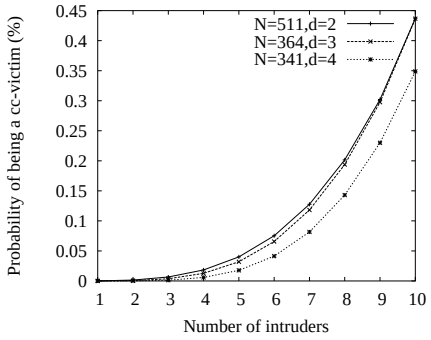
$$\frac{1}{N} \sum_{i=1}^{M} \sum_{l=1}^{l=H-1} [P_{h,f}(l \mid M_c = i) \times P(M_c = i) \times d^l] \quad (7)$$



(a) $N = 63, d = 2$



(b) $N = 1023, d = 2$



(c) $M = 10$

Figure 6. cc-victim probability

According to Eq. (4)-(7), the cc-victim probability can be calculated. Figure 6 shows that the cc-victim probability increases as N_c increases while the other parameters are the same. Figure 6 (a) and (b) shows that the cc-victim probability decreases as M increases. Comparing Figure 6 (a) and Figure 6 (b), we find that increasing N can reduce the cc-victim probability. Figure 6 (c) shows the impact of d. As shown in this figure, when N is similar, increasing d can reduce the cc-victim probability. In general, when M is not very small (e.g., $M \geq 10$), the cc-victim probability is very small, especially when N is large.

7 Performance Evaluations

In this section, we evaluate the proposed intrusion detection and damage recovery schemes in more practical scenarios. We first describe the simulation methodology, and then present and analyze the simulation results.

7.1 Simulation Methodology

We develop a simulator based on $ns2$ (version 2.1b8a) [19]. In this simulator, the IEEE 802.11 MAC protocol and the tworay ground propagation model are adopted. 100 clients are randomly distributed over a $1600 \times 400m^2$ flat field, and a server is located at a corner of the field. All the nodes are stationary and form a tree rooted at the server. The server maintains 100 data items, and each item is updated at an interval which is uniformly distributed with the mean value of $300s$. Each client access a data item at an interval which is uniformly distributed with the mean value of $30s$. Since a client may concentrate on a specific set of data items, we assume that 80% queries issued by a specific client are for a set of 10 data items, and the other 20% queries issued by the client are for other data items. Each client has a cache that can store 20 data items. Most simulation parameters are shown in Table 1.

Table 1. Simulation Parameters

Parameter	Values
Field size (m^2)	1600×400
Communication range of each node (m)	250
Bandwidth (Mb/s)	2
Number of data items	100
Cache size of each client (data items)	20
Average data update interval (s)	300
Average data access rate each client (s)	30
Client number	100
Group number: M	2-20
Intruder number	1-10
Guarding delay (s)	0.03-0.2
Simulation time for each scenario (s)	3000
IR interval (s)	20
IR window size	5

In the simulations, we use the following metrics: the misdetection probability (i.e., the probability that intrusions

are not detected by any node), the cc-victim probability, the average data access delay, and the extra bandwidth consumed by the proposed schemes. Note that the first two metrics are used for measuring the security level achieved by the schemes, and the last two metrics are for measuring the overhead brought by the schemes.

7.2 Simulation Results

7.2.1 Misdetection Probability

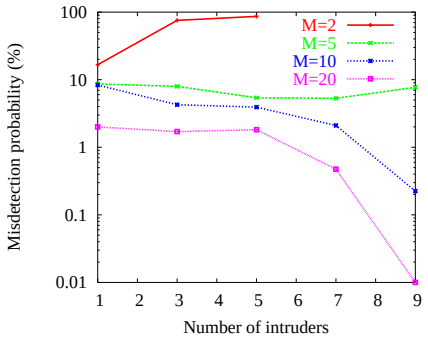
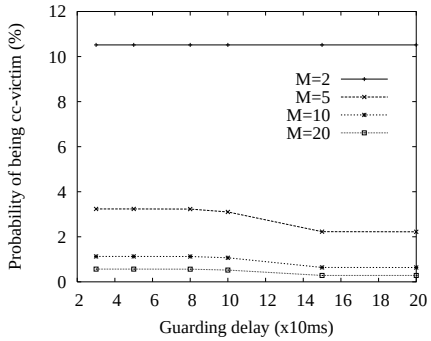


Figure 7. Misdetection probability

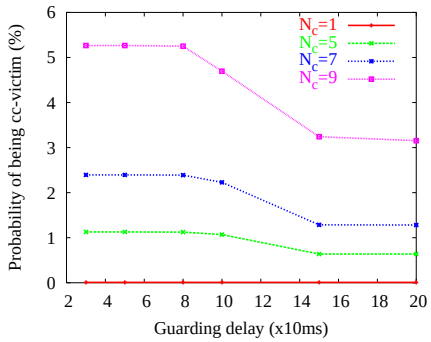
Figure 7 shows the misdetection probability as the number of intruders (N_c) and the number of groups (M) vary. When M is very small (e.g., $M = 2$), even a small number of intruders may compromise most group keys, which makes it difficult or impossible for an innocent node to detect the intrusion. Therefore, as shown in the figure, the misdetection probability is large and increases rapidly as N_c increases. However, as M increases, the number of group keys that are not compromised also increases, which reduces the misdetection probability. Also, when M is large (e.g., $M \geq 10$), more innocent nodes become the descendants of an intruder as N_c increases. Meanwhile, increasing N_c does not rapidly increase the number of compromised keys. Consequently, the misdetection probability decreases as N_c increases.

7.2.2 The cc-victim Probability

Figure 8 shows the cc-victim probability. As we can see, the cc-victim probability increases as the guarding delay increases. This can be explained as follows: A victim can not detect a compromised SIR when it receives it, but its descendant with non-compromised key may detect the intrusion later and send it an intrusion detection notification. In this case, increasing the guarding delay also increases the probability that a victim receives an intrusion detection notification before using the modified SIR.



(a) Impact of M ($N_c = 5$)



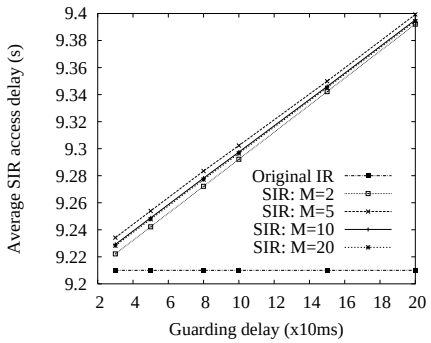
(b) Impact of N_c ($M = 10$)

Figure 8. cc-victim probability

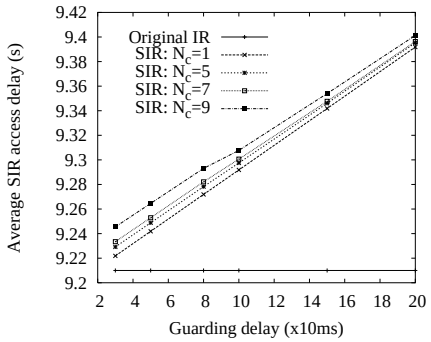
As shown in Figure 8 (a), when the number of intruders (N_c) is fixed, the cc-victim probability decreases as the number of groups (M) increases. This is mainly due to the reason illustrated in Figure 7. The intrusion misdetection probability decreases as M increases; i.e., the probability of detecting an intrusion and recovering from it increases as M increases. The impact of N_c on the cc-victim probability is shown in Figure 8 (b). When M is fixed, the cc-victim probability increases as N_c increases. This is because increasing N_c also increases the number of fully hijacked nodes (as shown in Section 6). Note that the trends shown in Figure 8 (a) and (b) are consistent with the analytic results shown in Figure 6.

7.2.3 Data Access Delay

Figure 9 shows the average data access delay as the system parameters vary. From the figure, we can find that using the proposed schemes increases the average data access delay, compared to the original IR scheme. We call the difference as *SIR access delay*, which is the average time elapsed from the time when an SIR is sent out by the server to the time when a node really uses it. From the figure, we can see that the data access delay is only *slightly increased* when using our scheme. For example, when $M = 10$ and $N_c = 9$, the delay is increased about 2%.



(a) Impact of M ($N_c = 5$)



(b) Impact of N_c ($M = 10$)

Figure 9. Average data access delay

As shown in Figure 9 (a), when the number of groups (M) is very small (e.g., $M = 2$), the average SIR access delay is also very small. This is because most intrusions can not be detected (as shown in Figure 7), and hence the SIR access delay for most nodes is minimized (i.e., equal to the sum of the SIR propagation delay and the guarding delay). However, when M is not very small (e.g., $M \geq 5$), most intrusions can be detected, and most intrusion victims can receive an intrusion detection notification before using the SIRs they have received. Therefore, they also need to wait for the hSIRs from the corresponding detectors (refer to Section 5.2), which increases their SIR access delay.

Figure 9 (a) also shows that, when $M \geq 5$, the SIR access delay decreases as M increases. This is due to the fact that the average victim-detector distance decreases as M increases. As the distance decreases, the time for a victim to receive a hSIR from the detector is reduced. Figure 9 (b) shows that, the SIR access delay increases as the number of intruders (N_c) increases. Similarly, this is due to the fact that the average victim-detector distance increases as N_c increases. As the distance increases, the time for a victim to receive a hSIR from the detector also increases.

Based on the results shown in Figure 8 and Figure 9, we can balance the cc-victim probability and the SIR access delay by selecting an appropriate guarding delay. For example, if $M = 10$ and the guarding delay is $150ms$, very low cc-victim probability (i.e., less than 1.5% when $N_c \leq 7$) can be achieved, and the SIR access delay is not large (i.e., $640ms$-$680ms$).

8 Conclusion

When caching techniques are used, cache consistency issues must be addressed. The invalidation-based approach is widely used to maintain strong cache consistency. However, this approach may suffer from some security attacks. For example, the invalidation messages may be dropped or modified by malicious nodes. To defend against such attacks, we proposed a solution based on the IR-based cache invalidation strategy. Since using digital signatures to protect the IR has significantly high overhead, we proposed a family of randomized grouping based techniques for intrusion detection and damage recovery. Analytical results and simulation results showed that the proposed solution can achieve good level of security with low overhead.

References

[1] D. Barbara and T. Imielinski. Sleepers and Workaholics: Caching Strategies for Mobile Environments. *ACM SIGMOD*, pages 1–12, 1994.

[2] S. Buchegger and J.-Y. Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes Fairness In Dynamic Adhoc NeTworks. *ACM Mobihoc*, pages 80–91, June 2002.

[3] G. Cao. A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments. *ACM Mobicom*, 2000.

[4] P. Cao and C. Liu. Cooperative Cache-Based Data Access in Ad Hoc Networks. *IEEE Transactions on Computers*, pages 445–457, April 1998.

[5] G. Cao, L. Yin, and C. Das. Cooperative Cache-Based Data Access in Ad Hoc Networks. *IEEE Computer*, pages 32–39, Feburary 2004.

[6] S. Das, C. Perkins, and E. Royer. Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks. *IEEE Infocom*, pages 3–12, 2000.

[7] W. Du and J. Deng. A Pairwise Key Pre-distribution Schemes for Wireless Sensor Networks. *The 10th ACM Conference on Computer and Communications Security*, 2003.

[8] E. Royer and C. Perkins. Multicast Operations of the Adhoc On-Demand Distance Vectore Routing Protocol. *Proceedings of ACM/IEEE Mobicom'99*, Aug. 1999.

[9] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Wireless Ad Hoc Networks. *ACM Mobicom*, Sep. 2002.

[10] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. *IEEE Infocom*, April 2003.

[11] H. Hubaux, L. Buttyan, and S. Capkun. The Quest for Security in Mobile Ad Hoc Networks. *ACM MobiHoc*, pages 146–155, 2001.

[12] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing, Kluwer*, pages 153–181, 1996.

[13] K. Chen and K. Nahrstedt. Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in MANET. *Proceedings of IEEE Infocom'02*, June 2002.

[14] P. Kyasanur and N. Vaidya. Detection and Handling of MAC Layer Misbehavior in Wireless Networks. *Technical Report, UIUC*, Aug. 2002.

[15] S. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 7(6):441–453, December 2002.

[16] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. *The 10th ACM Conference on Computer and Communications Security*, 2003.

[17] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. *ACM MobiCom*, Aug. 2000.

[18] A. Perrig, R. Canetti, J. Tygar, and D. Song. Effi cient Authentication and Signing of Multicast Streams over Lossy Channels. *IEEE Symposium on Security and Privacy*, May 2000.

[19] T. C. M. Project. The CMU Monarch Projects Wireless and Mobility Extensions to ns. *http://www.monarch.cs.cmu.edu/cmu-ns.html*, October 1999.

[20] M. Roussopoulos and M. Baker. CUP: Controlled update propagation in peer-to-peer networks. *Proceedings of the 2003 USENIX Annual Technical Conference*, 2003.

[21] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. *IEEE Int'l Conf. on Network Protocols (ICNP)*, Nov. 2002.

[22] L. Xiao, L. Ni, and A. Esfahanian. Prioritized Overlay Multicast in Mobile Ad Hoc Environments. *IEEE Computer*, Feb. 2004.

[23] L. Yin and G. Cao. Supporting Cooperative Caching in Ad Hoc networks. *IEEE INFOCOM'04*, 2004.

[24] W. Zhang and G. Cao. Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration Based Approach. *IEEE INFOCOM'05*, 2005.

[25] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad-Hoc Networks. *ACM Mobicom*, pages 275–283, Aug. 2000.

[26] L. Zhou and Z. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6):24–30, November/December 1999.

[27] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks. *IEEE Symposium on Security and Privacy*, 2004.