Available online at www.sciencedirect.com

ScienceDirect

Ad Hoc Networks 6 (2008) 363–379

Ad Hoc Networks

www.elsevier.com/locate/adhoc

Defending against cache consistency attacks in wireless ad hoc networks

Wensheng Zhang [a,*], Guohong Cao [b]

[a] Department of Computer Science, Iowa State University, United States
[b] Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, United States

Received 18 November 2005; received in revised form 24 July 2006; accepted 6 February 2007
Available online 28 February 2007

Abstract

Caching techniques can be used to reduce bandwidth consumption and data access delay in wireless ad hoc networks. When cache is used, the issue of cache consistency must be addressed, and maintaining strong cache consistency is desired in some strategic scenarios (e.g., battlefields). In these situations, the invalidation-based approach is preferred due to its low overhead. However, this approach may suffer from some security attacks. For example, malicious nodes (also called *intruders*) may drop, insert or modify invalidation messages to mislead receivers to use stale data or unnecessarily invalidate data that are still valid. In this paper, we first propose to employ the Invalidation Report (IR) based cache invalidation strategy to prevent intruders from dropping or modifying invalidation messages. Although digital signatures can be used to protect IRs, this has significantly high overhead in terms of computational and bandwidth overhead. To address this problem, we further propose a family of randomized grouping-based schemes for intrusion detection, damage recovery and intruder identification. Extensive analysis and simulations are performed to evaluate the proposed schemes. The results show that our solution can achieve a satisfactory level of security with low overhead.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Ad hoc network; Caching; Authentication; Intrusion detection; Damage recovery

1. Introduction

Mobile ad hoc networks have been the focus of recent research due to their potential applications in civilian and military environments such as battlefield, disaster recovery, group conference, and wireless office. In ad hoc networks, mobile nodes communicate with each other using multi-hop wireless links. Due to lack of infrastructural 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 [1–4] that can efficiently find routes between two communicating nodes. Although routing is an important issue in ad hoc networks, other issue such as information (data) access is also very important since the ultimate goal of using ad hoc networks is to provide information access to mobile nodes.

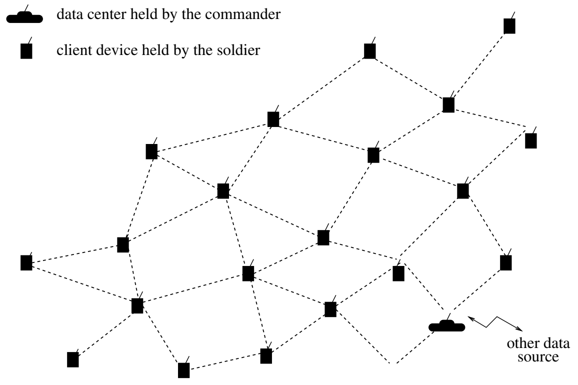
* Corresponding author. Tel.: +1 515 294 2821.
E-mail addresses: wzhang@cs.iastate.edu (W. Zhang), gcao@cse.psu.edu (G. Cao).

1570-8705/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2007.02.005

Caching frequently accessed data items at the client side is an effective technique to improve performance in mobile environments. With caching, both bandwidth consumption and data access delay can be 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. Fig. 1 shows an example in a battlefield, where the communication equipments held by a commander and a group of soldiers form an ad hoc network. The commander has the 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 some information from the data center, he or other soldiers near him may need to access it again and again. To save bandwidth and reduce access delay, the soldier can cache a copy of the data locally and serve requests from the local cache, as long as the cached copy is consistent with the copy 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, and database systems. Two widely used cache consistency models are the weak consistency model and the strong consistency model. In the weak consistency model, a 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 adversarial and strategic scenarios such as the battlefield (see Fig. 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, and then returns the valid copy to the user. However, in a large network, many clients may cache and frequently access some data items. Using the polling-based approach may generate significant network traffic [5], 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 this approach, the server keeps track of all the clients that cache the data item, and sends invalidation messages to the clients when the data is changed.


data center held by the commander
client device held by the soldier
other data source
Fig. 1. An ad hoc network in the battlefield.

In the adversarial 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. The attacks may prevent the systems from achieving the strong consistency. Basically, there are two kinds of attacks on the invalidation-based approach. First, the malicious or compromised node (intruder) may stop propagating invalidation messages, hence, mobile nodes far away from the data source may not be able to receive the invalidation messages and may use the stale cache (or replica) without realizing it. Second, an intruder may inject false invalidation messages, or modify passing invalidation messages, to mislead innocent receivers to use their stale caches or invalidate their caches which are still valid. If the data source authenticates each invalidation messages with a digital signature, receivers can verify messages and avoid the second kind of attack. However, digital signatures cannot be used to defend against the first kind of attack, since the mobile node may never receive the signed invalidation message.

In this paper, we propose solutions to deal with such attacks on cache consistency. To prevent malicious nodes from dropping the invalidation messages, we borrow ideas from the IR-based cache invalidation [6,7]. 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 measures can be taken to address the attack. To prevent malicious nodes from modifying the IRs and to authenticate the data source, 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, damage recovery and intruder identification. Extensive analysis and simulations are performed to evaluate the proposed schemes. The results show that our solution can achieve satisfactory level of security with low overhead.

The paper is organized as follows. In the next section, a brief review of the related work is presented. In Section 3, we propose solutions to defend against dropping invalidation messages. In Section 4, we propose randomized grouping techniques to reduce the authentication overhead. Security analysis of the proposed solution is presented in Section 5. Section 6 discusses the issues of reducing packet size and revoking intruders. Section 7 reports the simulation results and Section 8 concludes the paper.

2. Related work

Recently, many researchers have investigated the security issues in ad hoc networks. Hubaux et al. [8] addressed the issue 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 [9] proposed a solution based on threshold cryptography, in which an $(n, t+1)$ threshold cryptography scheme (with $n \geqslant 3t+1$) allows n parties to share the ability to perform a cryptography operation (e.g., creating a digital signature) so that any $t+1$ parties can perform this operation jointly, whereas it is infeasible for at most t parties to do so even by collusion.

Based on a trusted certificate authority, Sanzgiri et al. [10] proposed a solution to secure the routing protocol of ad hoc wireless networks. To address the high overhead associated with obtaining and verifying digital certificates, Hu et al. proposed a protocol [11] to secure on-demand routing protocols based on TESLA [12], an efficient broadcast authentication scheme that requires loose time synchronization. They also identified the wormhole attack [13], and provided a packet leash solution to solve the wormhole attack.

Even with secure routing protocols, a source and a destination may still set up a route which goes through a misbehaving (malicious) node agreeing to forward packets but failing to do so. In [14], the authors proposed to use a *watchdog* mechanism 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 deal with this problem, Buchegger and Le Boudec [15] designed protocols that are based on reputation systems. In this approach, the nodes observe the behaviors of each other and store the knowledge locally. Additionally, they distribute this information in reputation reports. The solution in [16] exploits collaboration among local nodes to protect the network without completely trusting any individual node. To motivate packet forwarding among mobile nodes, Buttyan and Hubaux [17] provided a solution based on a virtual currency, called *nuglets*: If a node wants to send its own packets, it has to pay for it, whereas if the node forwards a packet for the benefit of another node, it is rewarded.

Other researchers in this field also address the problem of intrusion detection [18] and MAC layer misbehavior [19]. 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. Defending against dropping invalidation messages

To prevent malicious nodes from dropping invalidation messages, we borrow ideas from the IR-based cache invalidation [6,7]. In this approach, the server periodically broadcasts an *invalidation report (IR)* which indicates the changed data items. The IR consists of the current timestamp T_i and a list of 2-tuples (d_x, t_x), where d_x is the data item *id*, t_x is the most recent update timestamp of d_x, w is the invalidation broadcast window size, and $t_x > (T_i - w \times L)$. 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 invalidation-based approach, clients can use the IR to invalidate their local cache. Different from the invalidation-based approach, the IRs are sent out regularly, and the clients expect IRs at regular time interval. Therefore, if a malicious node drops an IR, the nodes that are expecting the IR can detect it, and some measures can be taken to address the intrusion. The IR-based approach was originally designed for single-hop based mobile environment, and hence we have to find a way to distribute IRs to mobile nodes which may be multi-hops away from the data server.

The most reliable method to ensure that all nodes with cached data receive an IR is to use flooding. Although many protocols exist to reduce the flooding overhead, it is still high. Another option is to use multicast by setting up a multicast group (tree), where the data source is the root of the tree. A simple method for constructing and maintaining the multicast tree is as follows: Initially, the multicast tree has only one node which is the server. When a client N_i starts to cache data and needs to receive IR from the server, it sends a join request toward the server. When the request reaches a node (N_j) which is already in the tree, N_j becomes the parent of N_i. Each non-leaf node in the tree keeps a list of children id, which is a *soft-state*. Every time it receives an IR, it will forward the IR to each child in the list. When a child receives the forwarded IR, it should send an ACK to its parent. If a non-leaf node does not receive an ACK from one child for a certain time interval, it will remove the node from its children list and stop forwarding IRs to the node.

After joining the tree, N_i should receive IRs periodically, so it will be able to detect a missing IR. It may miss an IR because an intruder stops forwarding the IR or because the multicast tree is partitioned due to node movement. In either case, N_i will use the method described above to re-join the multicast tree and get the missed IR.

4. Reducing the authentication overhead by randomized grouping

To authenticate data source, digital signatures can be used. However, this approach has high overhead both in terms of computation and bandwidth [13]. To address this issue, researchers apply symmetric cryptographic techniques such as TESLA [12] to secure multicast and routing in ad hoc networks. 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 cannot be directly used for authenticating IRs. Next, we present a novel solution to reduce the authentication overhead by randomized grouping.

4.1. 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:

- (P_0^+, P_0^-): P_0^+ is its public key, and P_0^- is its private key.
- $\{K_i\}_{i=0,2,\ldots,M-1}$: 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 $(i = 1,2,\ldots)$ 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_0^+: 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 an authenticator issued by the server, i.e., $P_0^-(P_i^+)$.
- $K_{g(i)}$: 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 a 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 Fig. 2a, 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 $\langle IR,\ MAC_0, MAC_1\rangle$, where $MAC_i = C_{K_i}(IR)$, $i = 1,2$, 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.

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

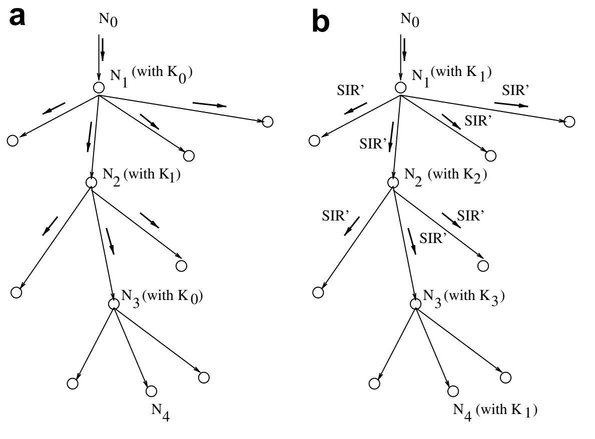


Fig. 2. Intrusion detection: (a) without collusion, (b) with collusion.

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.

4.2. Intrusion detection

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

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

where

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

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 an intruder. Otherwise, the node cannot 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 cannot detect it. Fig. 2b 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 are 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.

4.3. Intrusion recovery

When a node detects an intrusion, the detector sends the received SIR, the *id* of its parent, an accusation to its parent, and a request for the correct SIR, directly to the server. Since the message may go through a malicious node which can modify its content, the message should be signed using the private key 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.

On receiving a request for a correct SIR, the server replies with the correct SIR, which is encrypted using its private key so that other nodes cannot modify the message without being detected. 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 scheme, the victims (e.g., N_2 in Fig. 2) still cannot recover. To address the problem, the detector should send an *intrusion detection notification* to its parent. The notification message includes the SIR it received and the position of the incorrect MAC. The notification should also be signed by 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, responses as follows:

1. 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 directly to the server to accuse the child.

2. If the "incorrect MAC" claimed by the notification is actually correct, i.e., the detector maliciously generates a false alarm, the receiver sends a report directly to the server to accuse the detector, with the notification as the evidence.
3. If the notification passes the above tests, the receiver propagates the notification to its parent and other children, since they may also be victims. The receiver also sends the notification it has received, its parent *id*, and an accusation to its parent directly to the server 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. Fig. 3 illustrates an execution of the recovery scheme.

In the damage recovery scheme, 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 5) and the simulation results (Section 7), maintaining a satisfactory level of security only need a very short guarding delay, and hence the additional data access delay is very short.

Even with a long guarding delay, the damage recovery scheme cannot rescue all victims. Some victim 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 nodes detects it later; or even a descendant detects the intrusion, this node cannot 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.

Besides dropping a notification that it should forward, an intruder may attack the intrusion recovery scheme in other ways. Fig. 4a shows a case, where an intruder modifies a SIR it received, and sends out a false intrusion detection notification to its upstream nodes. According to rule (1) described above, the first innocent node on the path from the detector to the server, i.e., N_u, can detect the modification and identify its child (N_v, which may or may not be the malicious detector) must be malicious. Therefore, the attack can be limited in a small scope. Furthermore, according to the intruder identification rules described later, both N_u and N_v are identified as suspected intruders, since the server cannot distinguish this case from the case shown in Fig. 5a, where N_u is the intruder.

Fig. 4b shows another case, where the malicious detector does not modify a received SIR, but accuses MAC_i as incorrect. Accordingly to rule (2) described

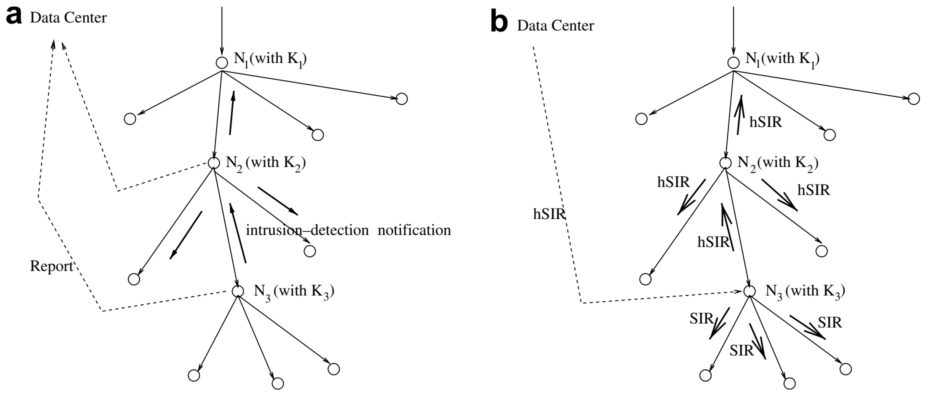


Fig. 3. Intrusion recovery: (a) the detector (N_3) reports to the data center, and sends intrusion detection notifications to the victims, (b) after receiving the hSIR from the server, N_3 sends the correct SIR to its children and forwards the hSIR to the victims.

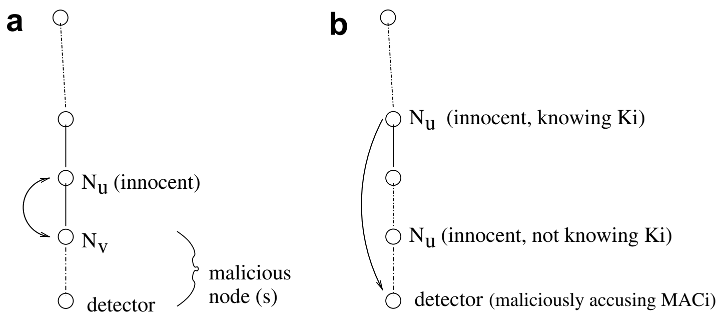

a
Nu (innocent)
Nv
malicious node (s)
detector
b
Nu (innocent, knowing Ki)
Nu (innocent, not knowing Ki)
detector (maliciously accusing MACi)

Fig. 4. Attacks on the intrusion recovery scheme.

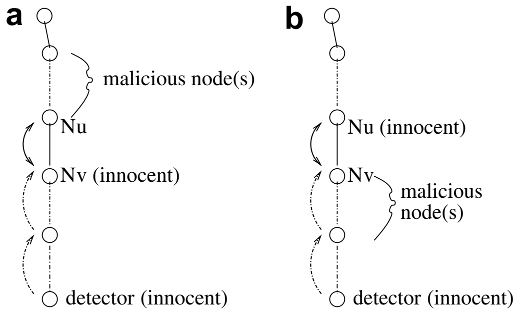


Fig. 5. Neighboring en-route nodes accusing each other.

above, N_u, which is first innocent node that knows K_i can detect the intrusion, and stop propagating the notification. Furthermore, according to rule (3), the first innocent node on the path from the malicious detector to the server, i.e., N_v, sends the notification to the server, and the server can identify the intruder, and send a hSIR to N_v to rescue the innocent nodes between the detector and N_u.

4.4. Intruder identification and isolation

After the server has collected the accusation reports from a series of nodes along a path, it can check the reports to identify the intruders or the suspected intruders. The rules are summarized as follows:

1. If a node reports an intrusion notification, and the SIR included in the notification is correct, the detector is identified as an intruder. If the node is not the detector, a hSIR is replied to the node for intrusion recovery. The principle behind this rule is already explained above.
2. If a pair of neighboring nodes send reports to accuse each other as the intruder, both of them are identified as suspects. Neighboring nodes may accuse each other due to several reasons. Fig. 4a shows one case which has been explained above. Fig. 5a and b shows other two cases. In (a), malicious node N_u (or its malicious upstream nodes) modified the SIR and sent the modified SIR to N_v, but it mis-accuses N_v as the intruder. In (b), after receiving a correct SIR from innocent node N_u, N_v or its malicious downstream nodes modified the SIR, but it maliciously accuses N_u as the intruder. In either case, they accuse each other, and the server cannot distinguish these two cases.
3. If an accused node does not accuse any other node within a certain time, the server informs the node of the accusation, and identifies both the accused node and the accusing node as suspects. Note that an accused node may fail to response due to network partition or the interference made by the accusing node. Also, the accused node may also be the true intruder and do not want to cooperate. Since the server cannot distinguish these cases from each other, it has to conservatively identify both of them as suspects.

Through the above isolation process, several suspects may be identified. For further isolation, the multicast tree should be adapted to separate the suspects which accuse each other. In order to identify the intruder, the server maintains a detection counter for each suspect, and the counter is initially set to one. The counter is increased by one whenever the suspect is accused of an intrusion. To prevent a malicious node from repeatedly accusing an innocent node, repeated accusations are counted only for once. A suspect will be identified as the intruder when the counter exceeds a certain threshold, e.g., 10% of the total number of clients in the network. After that, the *id* of the identified intruder is broadcast to all the nodes in the network, and the node will not be used as a parent node by any innocent node.

5. Security analysis

In this section, we show the level of security that our proposed schemes can achieve through analysis. Specifically, we use the following metrics: *misdetection probability* (i.e., the probability that intrusions are not detected by any node) and *cc-victim probability* (i.e., the probability that a node becomes a cc-victim). We also study the distance between a victim and a detector of the same intrusion (called *Victim-Detector distance*), since the distance is related to the guarding delay.

5.1. Preliminaries

5.1.1. Notations and assumptions

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

- N: the number of receivers (nodes).
- N_c: the number of compromised receivers (intruders).
- M: the number of groups. We assume that the receivers are uniformly distributed to these groups; i.e., the probability that a receiver is in group i ($i = 0, 1, \ldots, M-1$) is $\frac{1}{M}$.
- M_c: the number of compromised group keys.
- 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, $N = \sum_{i=1}^{H-1} d^i = d^H - 2$.

5.1.2. 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., $P(M_c = i|M, N_c)$ $(M > 0, N_c > 0)$ for $i = 1, 2, \ldots, \min(M, N_c)$.

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. Obviously, it is impossible to assign n_c nodes to i groups when n_c is smaller than i. Also, there is m options to assign all n_c nodes to one group. For other cases, the problem can be solved in two steps: First, the number of nodes (say j) assigned to the first group is decided; second, the number of options for assigning $n_c - j$ nodes to the remaining $i - 1$ of $m - 1$ groups are calculated. Thus,

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

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

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

and

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

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

5.2. The misdetection probability

5.2.1. The probability of being a potential detector (p-detector)

An innocent node has the potential to detect an intrusion only if it has an intruder ancestor. We call such a node *potential detector* (or *p-detector*). Let $P_u(l)$ be the probability that a l-level node is a p-detector.

- **Case I** ($l = 0$ **or** 1): Since the server cannot be an intruder, $P_u(0) = P_u(1) = 0$.
- **Case II** ($l > 1$): An innocent l-level node ($l > 1$) has at least one malicious ancestor if either of the following conditions is satisfied:
  1. Its parent is an intruder. The probability is: $\alpha = \frac{N_c}{N}$.
  2. Its parent is a p-detector. The probability is: $P_u(l-1)$. Therefore, $P_u(l) = (1-\alpha)\ [\alpha + P_u(l-1)]$.

In summary,

$$P_u(l) = \begin{cases} 0, & l = 0, 1; \\ (1-\alpha)[\alpha + P_u(l-1)], & 2 \leqslant l \leqslant H-1. \end{cases} \tag{4}$$

5.2.2. The probability that a p-detector misdetects an intrusion

A p-detector misdetects an intrusion only when the group key it holds has been compromised. The probability (denoted as $P_{u,m}$) is computed as follows:

$$\sum_{i=1}^{M}[P(\text{a p-detector misdetects}|M_c = i) \times P(M_c = i)]$$
$$= \sum_{i=1}^{M}[i/M \times P(M_c = i)] = E(M_c)/M.$$

Thus,

$$P_{u,m} = E(M_c)/M. \tag{5}$$

5.2.3. The misdetection probability

Let Z_l be the number of l-level p-detectors, and $p_m(l)$ be the probability that all l-level p-detectors mis-detect an intrusion. The probability that all p-detectors mis-detect an intrusion is:

$$E\left[\prod_{l=2}^{H-1} p_m(l)\right] = \prod_{l=2}^{H-1} E[p_m(l)], \tag{6}$$

where,

$$E[p_m(l)] = \sum_{i=0}^{d^l}[p_m(l|Z_l = i) \times P(Z_l = i)], \tag{7}$$

$$p_m(l|Z_l = i) = [E(M_c)/M]^i \tag{8}$$

and

$$P(Z_l = i) = (d^l_i)[P_u(l)]^i[1 - P_u(l)]^{d^l - i}. \tag{9}$$

Fig. 6 shows some analytic results of the misdetection probability. As shown in Fig. 6a and b, the misdetection probability is large when the number of groups (M) is small, and the probability drops rapidly as M increases. When M is very small (e.g., $M = 2$), the misdetection probability increases as the number of intruders increases. If M is large (e.g., $M \geqslant 10$), the misdetection probability drops as the number of intruders increases. When M is between 2 and 10, the misdetection probability first drops as the number of intruders (N_c) increases, and then increases when N_c becomes very large. Comparing Fig. 6a and b, we can see that the misdetection probability decreases as the number of receivers increases. Fig. 6c shows the impact of tree topology on the misdetection probability. With similar number of receivers (N), the misdetection probability increases as the tree degree (d) increases. In general, the misdetection probability is small as long as M is not very small (e.g., $M \geqslant 10$), especially when N is large.

5.3. The cc-victim probability

In this section, we show the sufficient and necessary condition for a node to be a cc-victim. Based on the condition, we derive the cc-victim probability.

5.3.1. Down hijacked, up hijacked and full hijacked

Before showing the sufficient and necessary condition, 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. The node is an intruder.
2. The node is a leaf node.
3. All the children of the node are down hijacked.

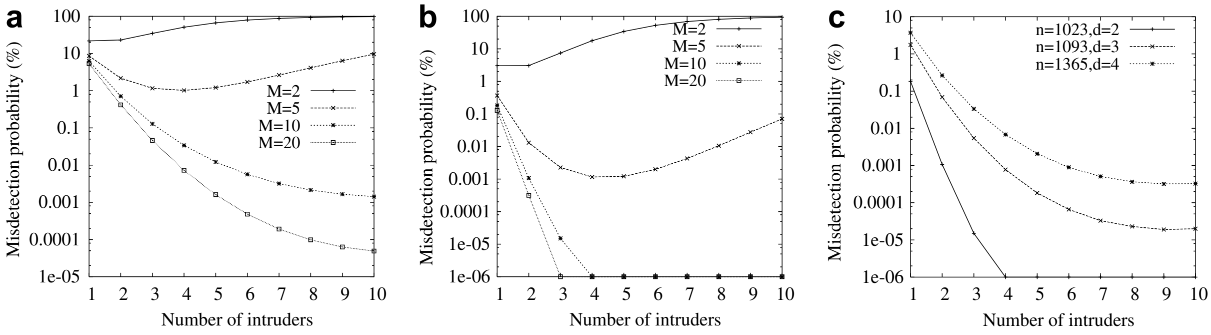


Fig. 6. Misdetection probability: (a) $N = 63$, $d = 2$; (b) $N = 1023$, $d = 2$; (c) $M = 10$.

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. The node is an intruder.
2. The parent of the node is an intruder.
3. The parent of the node is up hijacked and the other children of its parent are down hijacked.

Corollary 1. *An up hijacked node is either an intruder or a victim.*

Proof (*sketch*). Based on Definition 2, the corollary can be easily proved by induction on the level of an up hijacked node. □

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

Fig. 7 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 $(N_2, N_3, \ldots, N_6)$, N_2 and N_6 are down hijacked, N_2 and N_3 are up hijacked, and only N_2 is fully hijacked.

5.3.2. The sufficient and necessary condition for being a cc-victim

Lemma 1. *A node is a cc-victim only if it is down hijacked.*

Proof (*by induction*). Let us define a proposition as follows:

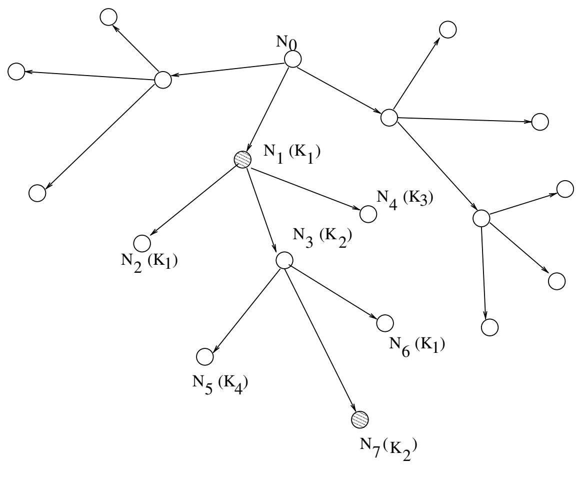


Fig. 7. Up hijacked, down hijacked and fully hijacked nodes.

Proposition 1. *For any l-level node N_i, if N_i is a cc-victim, it is down hijacked; or, if N_i is not down hijacked, it is not a cc-victim.*

Next, we will show by induction that Proposition 1 holds for $l \in \{1, 2, \ldots, H-1\}$.

1. **Base case**: $l = H-1$. Suppose a $(H-1)$-level (i.e., leaf) node N_i is a cc-victim. Its group key must have been known to an intruder. According to Definition 1, it should be down hijacked. Thus, Proposition 1 holds for $l = H-1$.
2. **Inductive step.** Assume that Proposition 1 holds for $l \in \{m, m+1, \ldots, H-1\}$, where $1 < m \leqslant H-1$. We now consider the case of $l = m-1$. Suppose a $(m-1)$-level node N_i is not down hijacked. According to Definition 1, it must have a child N_j which is not down hijacked (and hence not an intruder). According to the inductive assumption, m-level node N_j must not be a cc-victim. So N_j is a detector or a victim that can be rescued; i.e., if N_j receives a modified SIR, it can detect the intrusion or can receive a notification from a detector of the intrusion. Therefore, if N_i receives a modified SIR and does not detect it, N_i will pass the SIR to N_j and will receive an intrusion notification from N_j. So N_i is not a cc-victim. Proposition 1 holds for $l = m-1$.

   Based on steps (1) and (2), we conclude that Proposition 1 holds for any $l \in \{1, 2, \ldots, H-1\}$. □

Lemma 2. *A node is a cc-victim only if it is up hijacked.*

Proof (*sketch*). Similar to the proof of Lemma 1, 2 can be proved by induction on the level of a node. □

Lemma 3. *If a node is fully hijacked, it must be an intruder or a cc-victim.*

Proof ((*sketch*) *by contradiction*). Suppose there exist a fully hijacked node N_i, which is neither an intruder nor a cc-victim. According to Corollary 1, N_i is a victim. Since it is not a cc-victim, there must exist a detector N_j and a path connecting N_i and N_j. Also, the path should be free of intruders. Let the path be $N_j = N'_0 \to N'_1 \to \cdots \to N'_l = N_i$. Next, we will show that N_i is not fully hijacked. Specifically, we want to show (by induction) that the following proposition holds for $k = 0, 1, \ldots, l-1$. □

Proposition 2. *If N'_k is the parent of N'_{k+1}, N'_{k+1} is not up hijacked; if N'_k is a child of N'_{k+1}, neither N'_k nor N'_{k+1} is down hijacked.*

Based on Lemma 1–3, we can get the following theorem:

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

5.3.3. The probability of being a cc-victim

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

According to Definition 1,

$$P_{h,d}(l|M_c) = \begin{cases} \frac{M_c}{M}, & l = H-1; \\ \frac{M_c}{M}[\alpha + (1-\alpha)P_{h,d}(l+1|M_c)]^d, \\ \quad 0 \leqslant l < H-1. \end{cases} \tag{10}$$

According to Definition 2,

$$P_{h,u}(l|M_c) = \begin{cases} 0, & l = 0,1; \\ \frac{M_c}{M}[\alpha + (1-\alpha)P_{h,u}(l-1|M_c)] \\ \quad [\alpha + (1-\alpha)P_{h,d}(l \mid M_c)]^{d-1}, \\ \qquad 2 \leqslant l \leqslant H-1. \end{cases} \tag{11}$$

According to Definition 3,

$$P_{h,f}(l|M_c) = P_{h,d}(l|M_c) \times P_{h,u}(l|M_c). \tag{12}$$

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

$$\frac{1}{N}\sum_{i=1}^{M}\sum_{l=1}^{l=H-1}[P_{h,f}(l|M_c = i) \times P(M_c = i) \times d^l]. \tag{13}$$

According to Eqs. (10)–(13), the cc-victim probability can be calculated based on the system parameters such as the number of receivers (N), the number of intruders (N_c), the number of groups (M) and the tree degree (d). From these results, we can find the relationship between the cc-victim probability and these parameters. Fig. 8 shows that the cc-victim probability increases as N_c increases while the other parameters are the same. Fig. 8a and b shows that the cc-victim probability decreases as M increases. Comparing Figs. 8a and 8b, we find that increasing N can reduce the cc-victim probability. Fig. 8c 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., $M \geqslant 10$), the cc-victim probability is very small, especially when N is large.

5.4. Victim-detector distance

The victim-detector distance (D) is highly related to the guarding delay (T_g), and the relationship can be represented as $E[T_g] = E[T_r] \times E[D]$, where T_r is the round trip time between neighboring nodes. Next, we calculate the distribution of D, and study the relationship between d_{v_d}, T_g and some system parameter.

Suppose N_i is a victim and N_j detects this intrusion. The path connecting N_i and N_j is $N_i = N'_0 \to N'_1 \to \cdots \to N'_k = N_j$. According to the definition of victim and detector, N'_i $(i = 1, 2, \ldots, k-1)$ should all be victims. Therefore, the probability that N_i and N_j is k hops away can be calculated as:

$$P(d_{i,j} = k|M_c) = \left(\frac{M_c}{M}\right)^{k-1}\left(1 - \frac{M_c}{M}\right).$$

Thus, the average distance between any victim-detector pair, denoted as $E[D]$ can be calculated as:

$$E[D] = \sum_{m=1}^{M}\{E[D|M_c = m] \times P(M_c = m)\}, \tag{14}$$

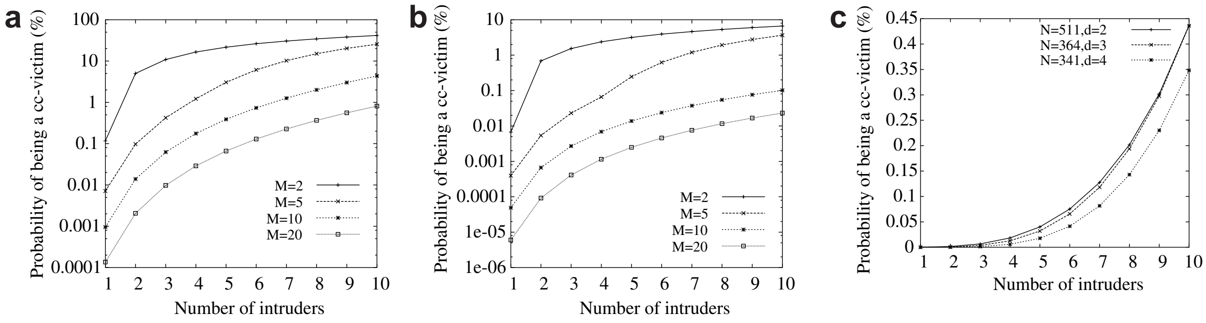


Fig. 8. cc-Victim probability: (a) $N = 63$, $d = 2$; (b) $N = 1023$, $d = 2$; (c) $M = 10$.

where

$$E[D|M_c] = \sum_{k=1}^{\infty}[k \times P(D = k \mid M_c)] \tag{15}$$

and

$$P(D = k|M_c) = \left(\frac{M_c}{M}\right)^{k-1}\left(1 - \frac{M_c}{M}\right). \tag{16}$$

Figs. 9 and 10 show some typical results of the average victim-detector distance and the distribution of the distance. As shown in Fig. 9, when the number of groups (M) is fixed, $E[D]$ increases as the number of intruders (N_c) increases. On the other hand, when N_c is fixed, $E[D]$ decreases as M increases. As shown in Fig. 10a, $P(D \leqslant 1) > 0.9$ when $M = 10$ and $N_c = 1$, which means that a node has 90% probability of not unknowingly using a compromised IR if it delays using the IR for 1-hop round trip time. Also, the delay increases as N_c increases. For example, when $N_c = 8$, the delay is as large as 5-hop round trip time in order to achieve the same level of security. Fig. 10b shows the impact of M on the distribution of D. As shown in the figure, when $M = 5$, a node should wait a 5-hop round trip time before using an IR in order to have 90% probability of not being cheated, and the delay decreases as M increases. From the figure, we can see that the delay is very short as M is not very small and N_c is not very large. Also, the delay is independent of the number of receivers.

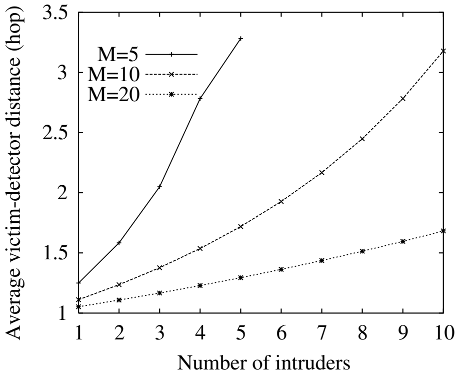


Fig. 9. Average victim-detector distance.

6. Discussion

6.1. Reducing message size

With the proposed scheme, several MACs should be attached to each message, and the MACs may occupy a large space. For example, if $M = 10$ and $MD5$ is used, the MACs occupy up to 1280 bits per packet, while each IR message typically has only tens of bytes. To reduce message size, the size of each MAC should be reduced, and the simplest way is to use a MAC with smaller size. This however will degrade the security level. To study the impact of smaller size MACs on the security level of the system, specifically, the probability of an innocent node being a cc-victim, we refine the formulae (10) and (11) to be as follows (note that v represents the size of each MAC):

$$P_{h,d}(l|M_c) = \begin{cases} \frac{M_c}{M} + \left(1 - \frac{M_c}{M}\right) \times 2^{-v}, & l = H - 1; \\ \left(\frac{M_c}{M} + \left(1 - \frac{M_c}{M}\right) \times 2^{-v}\right) \\ \quad \times[\alpha + (1-\alpha)P_{h,d}(l+1 \mid M_c)]^d, \\ \quad 0 \leqslant l < H - 1. \end{cases} \tag{17}$$

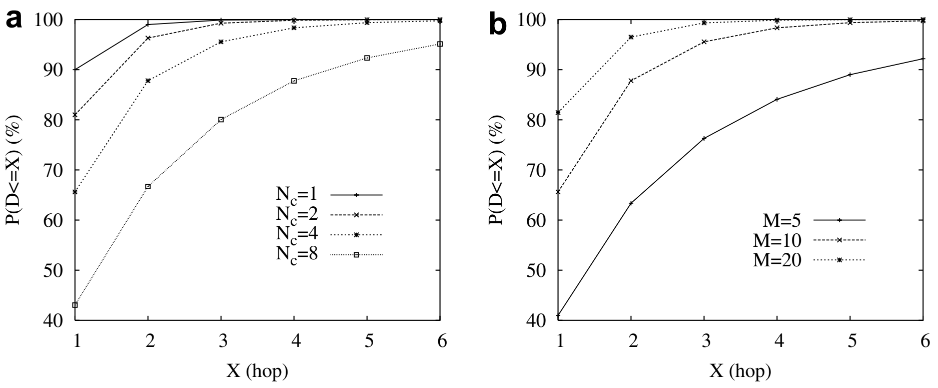


Fig. 10. The distribution of victim-detector distance (D): (a) impact of N_c ($M = 10$), (b) impact of M ($N_c = 4$).

$$P_{h,u}(l|M_c) = \begin{cases} 0, \quad l = 0,1; \\ \left(\frac{M_c}{M} + \left(1 - \frac{M_c}{M}\right) \times 2^{-v}\right) \\ \quad \times [\alpha + (1-\alpha) P_{h,u}(l-1|M_c)] \\ \quad \times [\alpha + (1-\alpha) P_{h,d}(l|M_c)]^{d-1}, \\ \quad 2 \leqslant l \leqslant H - 1. \end{cases} \tag{18}$$

Based on the above formulae, we calculate the probability of an innocent node being a cc-victim as the MAC size, the total number of nodes, the group size and the number of intruders (N_c) vary. Table 1 shows some numeric results. As shown in the table, if the MAC size is reduced too much from 128 bits, for example, 4 bits, the resulting probability is noticeably increased. However, as long as the size is not too small, in particular, greater than 8 bits, the reduction of the probability is generally slight. Based on the analytical results, the tradeoff between the security level and the MAC size (and hence the bandwidth consumption) can be made according to the required security level and the availability of resource in the network.

6.2. Group key update

After some nodes have been compromised by the adversary, the adversary may obtain the group keys held by these nodes, and give the keys to their newly compromised nodes. As previously mentioned, with more group keys, the newly compromised nodes can cheat more innocent nodes and become more difficult to be identified. Therefore, the compromised keys should be updated soon after they are identified.

The basic idea of a simple solution is as follows. Node N_i, which is randomly assigned to group $g(i)$, is preloaded with:

- p_0: a primary key.
- $f_{g(i)}(x)$: a function used to compute its group keys.
- $h(x)$: a hash function used to verify the received primary keys.

Initially, N_i computes $K_{g(i)} = f_{g(i)}(p_0)$, and uses it as its group key for verifying received IRs. After that, the server can initiate key update whenever

Table 1
The probability of an innocent node being a cc-victim when the MAC size varies (unit: %)

MAC size (bits)	Total number of nodes (N)	Group size (M)	$N_c = 2$	$N_c = 4$	$N_c = 6$	$N_c = 8$	$N_c = 10$
4	63	10	0.016232	0.153286	0.526376	1.221938	2.305920
8	63	10	0.013046	0.136286	0.483179	1.139149	2.169821
10	63	10	0.012888	0.135446	0.481055	1.135092	2.163165
16	63	10	0.012836	0.135171	0.480359	1.133763	2.160984
32	63	10	0.012835	0.135166	0.480348	1.133742	2.160950
128	63	10	0.012835	0.135166	0.480348	1.133742	2.160950
4	63	20	0.003124	0.032309	0.121980	0.305529	0.613749
8	63	20	0.002051	0.025525	0.103090	0.267121	0.547442
10	63	20	0.001998	0.025189	0.102157	0.265232	0.544192
16	63	20	0.001980	0.025079	0.101851	0.264613	0.543127
32	63	20	0.001980	0.025077	0.101846	0.264603	0.543110
128	63	20	0.001980	0.025077	0.101846	0.264603	0.543110
4	1023	10	0.000840	0.007774	0.025953	0.058580	0.107884
8	1023	10	0.000678	0.006947	0.023963	0.054957	0.102189
10	1023	10	0.000670	0.006906	0.023865	0.054779	0.101908
16	1023	10	0.000667	0.006893	0.023833	0.054720	0.101816
32	1023	10	0.000667	0.006892	0.023832	0.054719	0.101815
128	1023	10	0.000667	0.006892	0.023832	0.054719	0.101815
4	1023	20	0.000144	0.001471	0.005437	0.013263	0.025878
8	1023	20	0.000095	0.001170	0.004635	0.011716	0.023351
10	1023	20	0.000092	0.001155	0.004595	0.011639	0.023226
16	1023	20	0.000092	0.001150	0.004582	0.011614	0.023185
32	1023	20	0.000092	0.001150	0.004582	0.011613	0.023184
128	1023	20	0.000092	0.001150	0.004582	0.011613	0.023184

necessary. When the group keys are updated for the *i*th ($i = 1,\ldots$) time, the server disseminates primary key p_i, where $h(p_i) = p_{i-1}$, to all the trusted receivers.

To securely disseminate primary key p_i, the server first sends out key p_i to each trusted neighbor, respectively. p_i should be encrypted using the secret key shared by the sender and the receiver, such that other nodes cannot eavesdrop the message. When N_i receives a p_i and it has not received a correct p_i before, it checks the integrity of the message by comparing $h(p_i)$ and p_{i-1}. If they are different, the received p_i is incorrect and should be dropped. Otherwise, N_i updates its group key to be $f_{g(i)}(p_i)$, and sends the key to each of its trusted neighbors except the neighbor from which it has received p_i. The process continues, until each trusted node has updated its group key.

7. Performance evaluations

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

7.1. Simulation methodology

We develop a simulator based on *ns*2 (version 2.1b8a) [20]. In this simulator, the IEEE 802.11 MAC protocol and the two-ray ground propagation model are adopted. One hundred clients are distributed over a 1600 m × 400 m flat field, and a server is located at a corner of the field. The server is static and the clients move within the field according to the way-point model. The server maintains 100 data items, and each item is updated at an interval which is uniformly distributed with the mean value of 300 s. Each client access a data item at an interval which is uniformly distributed with the mean value of 30 s. 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 2.

In the simulations, we use the following metrics: the misdetection probability, the cc-victim probability, the average data access delay, and the extra bandwidth consumption caused 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 caused by the schemes.

Table 2
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
Maximum velocity of each client (m/s)	10
Pause time of each client (s)	60
IR interval (s)	20
IR window size	5
Size of an item in a IR (bit)	64
Size of MAC list in a SIR (bit)	$128 + 10 \times (M - 1)$

7.2. Simulation results

7.2.1. Misdetection probability

Fig. 11 shows the misdetection probability (both from simulations and analysis) 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., $M \geqslant 10$), 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. Generally speaking, the simulation and the analysis results shown in this figure are consistent with each other, though the misdetection probability resulted from simulations is greater than that resulted from analysis. This difference is mainly because the network topology in simulation is not regular. Specifically, some nodes have more than two children,

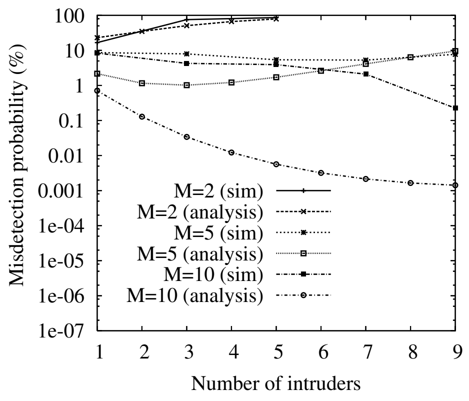


Fig. 11. Misdetection probability.

which is different from the assumption of the analysis. This implies that the height of the tree in simulations is generally shorter than the tree in analysis, and hence has smaller number of potential detectors. Consequently, the probability of misdetection is increased.

7.2.2. cc-Victim probability

Fig. 12 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 cannot detect a compromised SIR when it receives it. But its descendant with uncompromised 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.

As shown in Fig. 12a, 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 Fig. 11. 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 Fig. 12b. 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 5). The simulation and the analysis results shown in this figure are generally consistent with each other. Due to the similar reasons explained in the previous subsection, the cc-victim probability resulted from simulations is greater than that resulted from analysis.

7.2.3. Data access delay

Fig. 13 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 a SIR is sent out by the server to the time when a node really uses it.

As shown in Fig. 13a, 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 cannot be detected (as shown in Fig. 11), 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., $M \geqslant 5$), 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

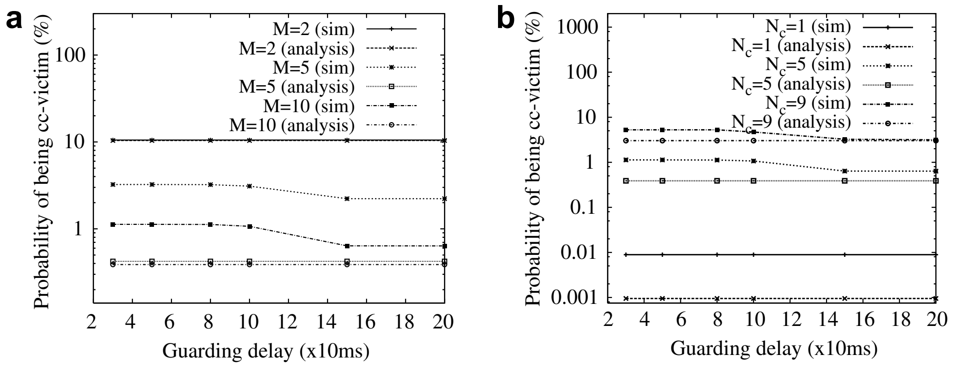


Fig. 12. cc-Victim probability: (a) impact of M ($N_c = 5$), (b) impact of N_c ($M = 10$).

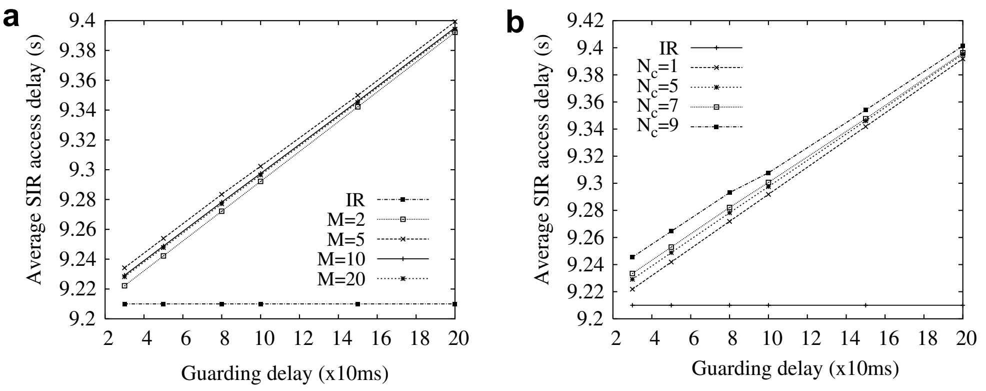


Fig. 13. Average data access delay: (a) impact of M ($N_c = 5$), (b) impact of N_c ($M = 10$).

the corresponding detectors (refer to Section 4.2), which increases their SIR access delay.

Fig. 13a also shows that, when $M \geqslant 5$, the SIR access delay decreases as M increases. This is due to the phenomenon illustrated in Fig. 9; i.e., 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. Fig. 13b shows that, the SIR access delay increases as the number of intruders (N_c) increases. Similarly, this is due to the other phenomenon illustrated in Fig. 9; i.e., 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 Figs. 12 and 13, 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 150 ms, very low cc-victim probability (i.e., less than 1.5% when $N_c \leqslant 7$) can be achieved, and the SIR access delay is not large (i.e., 640–680 ms).

7.3. Extra bandwidth consumption

Fig. 14 compares the extra bandwidth consumption caused by the original IR scheme and our proposed scheme. From the figure, we can see that our scheme has about 13–25% more extra bandwidth consumption. As the number of groups (M) increases, the extra bandwidth consumption also increases, since the MAC list has larger size. With fixed M, the extra bandwidth consumption increases as the number of intruders (N_c) increases. This is due to the reason that, some intrusion detection notification messages and accusation messages should be exchanged whenever an intrusion is detected. These messages have larger size and consume more bandwidth, since they are protected with digital signatures (1024 bit).

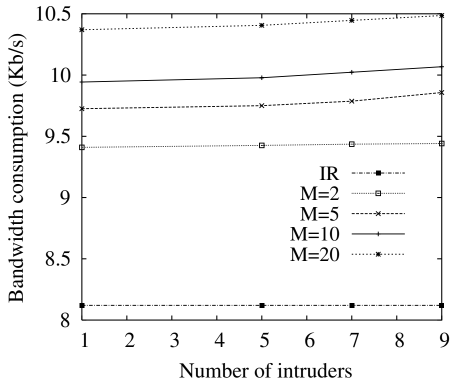


Fig. 14. Extra bandwidth consumption for different schemes.

8. Conclusions

When caching techniques are used, the issue of cache consistency 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, damage recovery and intruder identification. Analytical results and simulation results verify that the proposed solution can achieve satisfactory level of security with low overhead.

References

[1] S. Das, C. Perkins, E. Royer, Performance comparison of two on-demand routing protocols for ad hoc networks, IEEE Infocom (2000) 3–12.

[2] D. Johnson, D. Maltz, Dynamic source routing in ad hoc wireless networks, Mobile Computing, Kluwer (1996) 153–181.

[3] Y. Ko, N. Vaidya, Location-aided routing in mobile ad hoc networks, ACM Mobicom (1998) 66–75.

[4] S. Lee, W. Su, M. Gerla, On-demand multicast routing protocol in multihop wireless mobile networks, ACM/Kluwer Mobile Networks and Applications (MONET) 7 (6) (2002) 441–453.

[5] P. Cao, C. Liu, Maintaining strong cache consistency in the World Wide Web, IEEE Transactions on Computers (1998) 445–457.

[6] D. Barbara, T. Imielinski, Sleepers and workaholics: caching strategies for mobile environments, ACM SIGMOD (1994) 1–12.

[7] G. Cao, A scalable low-latency cache invalidation strategy for mobile environments, ACM Mobicom, 2000.

[8] H. Hubaux, L. Buttyan, S. Capkun, The quest for security in mobile ad hoc networks, ACM MobiHoc (2001) 146–155.

[9] L. Zhou, Z. Haas, Securing ad hoc networks, IEEE Network 13 (6) (1999) 24–30.

[10] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, E. Belding-Royer, A secure routing protocol for ad hoc networks, in: IEEE Int'l Conf. on Network Protocols (ICNP), November 2002.

[11] Y. Hu, A. Perrig, D. Johnson, Ariadne: a secure on-demand routing protocol for wireless ad hoc networks, ACM Mobicom, September 2002.

[12] A. Perrig, R. Canetti, J. Tygar, D. Song, Efficient authentication and signing of multicast streams over lossy channels, in: IEEE Symposium on Security and Privacy, May 2000.

[13] Y. Hu, A. Perrig, D. Johnson, Packet leashes: a defense against wormhole attacks in wireless ad hoc networks, in: IEEE Infocom, April 2003.

[14] S. Marti, T. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in: ACM MobiCom, August 2000.

[15] S. Buchegger, J-Y. Boudec, Performance analysis of the CONFIDANT Protocol: Cooperation Of Nodes Fairness In Dynamic Adhoc NeTworks, ACM Mobihoc (2002) 80–91.

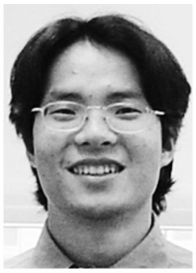
[16] H. Yang, X. Meng, S. Lu, Self-organized network-layer security in mobile ad hoc networks, in: ACM Workshop on Wireless Security, August 2002.

[17] L. Buttyan, J. Hubaux, Stimulating cooperation in self-organizing mobile ad hoc networks, ACM/Kluwer Mobile Networks and Applications (MONET) 8 (5) (2003).

[18] Y. Zhang, W. Lee, Intrusion detection in wireless ad-hoc networks, ACM Mobicom (Aug.) (2000) 275–283.

[19] P. Kyasanur, N. Vaidya, Detection and handling of MAC layer misbehavior in wireless networks, Technical Report, UIUC, August 2002.

[20] The CMU Monarch Project, The CMU Monarch Projects Wireless and Mobility Extensions to ns. <http://www.monarch.cs.cmu.edu/cmu-ns.html>, October 1999.



Wensheng Zhang received his BS degree from Tongji University, Shanghai, China, and his MS degree from Chinese Academy of Sciences. He received his Ph.D. degree in computer science from the Pennsylvania State University in 2005. Since then, he has been with the Department of Computer Science at Iowa State University as an assistant professor. His research interests are wireless networks and network security. He is an IEEE member.

Guohong Cao received his BS degree from Xian Jiaotong University, Xian, China. He received the MS degree and Ph.D. degree in computer science from the Ohio State University in 1997 and 1999 respectively. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently an Associate Professor. His research interests are wireless networks and mobile computing. He has published one hundred papers in the areas of sensor networks, data dissemination, resource management, wireless network security, and distributed fault-tolerant computing. He is an editor of the IEEE Transactions on Mobile Computing and IEEE Transactions on Wireless Communications, a co-guest editor of special issue on heterogeneous wireless networks in ACM/Kluwer Mobile Networking and Applications, and has served on the program committee of many conferences. He was a recipient of the Presidential Fellowship at the Ohio State University in 1999, and a recipient of the NSF CAREER award in 2001. He is a senior member of IEEE.