Byzantine-Tolerant, Point-To-Point Information Propagation in Untrustworthy and Unreliable Networks

Kai Han[1], Binoy Ravindran[1], and E. Douglas Jensen[2]

[1] ECE Dept., Virginia Tech, Blacksburg, VA 24061, USA, khan05@vt.edu, binoy@vt.edu
[2] The MITRE Corporation, Bedford, MA 01730, USA, jensen@mitre.org

Abstract. In a decentralized network system, an authenticated node is referred to as a Byzantine node, if it is fully controlled by a traitor or an adversary, and can perform destructive behavior to disrupt the system. Typically, Byzantine nodes together or individually attack point-to-point information propagation by denying or faking messages. In this paper, we assume that Byzantine nodes are of some intelligence — that is, they can protect themselves from being identified by authentication mechanisms, including encryption mechanisms. Therefore, a node cannot trust any other node in the system. We present an authentication-free, gossip-based application-level propagation mechanism called LASIRC, in which "healthy" nodes utilize Byzantine features to defend against Byzantine attacks. We show that LASIRC is robust against message-denying and message-faking attacks. Our experimental studies verify LASIRC's effectiveness.

1 Introduction

In a decentralized, network-based information system, nodes communicate with each other through point-to-point information propagation (i.e., directly sending messages to destination nodes without receiving help from a server). The propagation may suffer attacks from malicious nodes hiding in the system. Attacks where a traitor or an adversary has full control of an authenticated device, and can perform destructive behaviors to disrupt the system are referred to as Byzantine attacks [1]. A node showing Byzantine behaviors is called a Byzantine node. Byzantine nodes are more difficult to deal with than other attackers [2, 3].

We consider network systems where authentication mechanisms (including any kind of encryption) are unable to defend against Byzantine attacks. Since Byzantine nodes are authenticated, in any authentication process, they act just like other nodes. Therefore, a "healthy" node cannot trust its peers — it does not know whether another node is a friend, a traitor, or an adversary. Further, we assume that Byzantine nodes are intelligent—i.e., if a Byzantine node cannot protect itself from being identified by others, it will not attack. For instance, it is less possible for an attacker to fake a reply message during a request message propagation process, because such an attack will finally be identified by the real reply node (we discuss this attack scenario in Section 2.3). In the rest of the paper, we use the terms "Byzantine nodes" and "Byzantine attackers", interchangeably.

We focus on application-level Byzantine attacks in *request-reply* message propagation. A node initiates such a process by sending REQUEST (REQ) messages to others, trying to get a YES/NO answer from them (e.g., requesting a service that it cannot provide [4]). A knowing receiver responds by sending back REPLY (REP) messages, indicating its answer to the sender. Byzantine nodes may attack both processes by denying or faking messages (e.g., faking requested service ID in a REQ message, faking NO for an YES answer in a REP message, or directly faking REP messages). Furthermore, we consider information systems that use *unreliable networks* (e.g., those without a fixed network infrastructure, including mobile, ad hoc and wireless networks) with dynamically uncertain properties. These uncertain properties, which are application- and network-induced, include arbitrary node failures, transient and permanent network failures, and varying packet drop behaviors. Example such systems that motivate our work include US DoD's Network-Centric Warfare system [5].

In this paper, we present LASIRC, a Byzantine-tolerant, gossip-based, point-to-point application-level information propagation model/mechanism. Gossip mechanisms are well-known for their robustness to propagation uncertainties in unreliable networks [2]. A node initiates a gossip process by starting a series of synchronous gossip rounds. During each round, nodes holding REQ messages randomly select a set of neighbors and send REQ messages to them. The number of gossip rounds (or R), and the number of selected neighbors (i.e., the "fan-out" number, or F) are determined by the original sender [6]. If a receiver

knows the answer, it gossips REP messages back. Message losses and node failures may happen during a gossip process. However, gossip "fights" non-determinism (i.e., unpredictable message losses and node failures) with non-determinism (i.e., randomly selecting sending targets) — duplicated REQ and REP messages guarantee the normal information propagation. Our gossip-based LASIRC model/mechanism also features this same robustness, and makes LASIRC robust against Byzantine message-denying attacks, as these attacks can be regarded as message losses/node failures (Byzantine attackers receive, but do not forward messages). In addition to message-denying attacks, LASIRC also provides a set of mechanisms to detect and defend Byzantine message-faking attacks.

The rest of the paper is organized as follows: In Section 2, we discuss possible Byzantine attacks in the LASIRC model. We then present our Byzantine attacker detectors in Section 3. Sections 4 and 5 describe and analyze the LASIRC model and mechanism, respectively. In Section 6, we report on our experimental (simulation) studies. We conclude the paper and identify future work in Section 7.

2 Byzantine Attacks

We describe the REQ and REP message structures, discuss Byzantine attack types (denying and faking messages), and possible Byzantine attacks in REQ and REP message propagation under gossip protocols.

2.1 Message Structures

A REQ message contains the original sender's (the request node) IP, the intermediate node's (message-transferring node in gossip protocol) IP, the requested service ID, the gossip fan-out number (F), and the number of gossip rounds (R). A REP message contains the original sender (the reply node)'s IP, the intermediate node's IP, the requested service ID, the answer ("YES" or "NO"), F, and R.

2.2 Byzantine Attack Types

A message-denying attack issued by an individual attacker is called a *Black Hole* attack. In addition, two or more attackers may collude together to form a larger "*Black Hole*". We call these two attacks as *Black Hole Class (BHC)* attacks. BHC attacks become more serious if attackers participate in message propagation at early gossip rounds, or a large number of attackers collude together. Under such conditions, BHC attacks largely reduce the message number, and thus seriously slow or even cease message propagation.

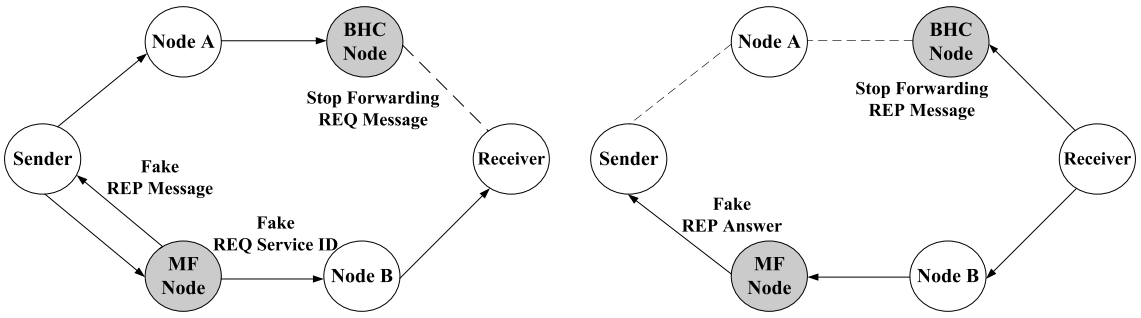
A *Message-Faking (MF)* attack is more harmful than a BHC attack. Unlike BHC attackers that only deny messages, an MF attacker directly propagates incorrect information, misleading other nodes to make a wrong decision. For instance, a node may send a REQ message requesting a remote service. If an MF attacker replies with a "NO" instead of the correct answer "YES", the sender will incorrectly abort a waiting task. If the attacker replies with a "YES" for a "NO", the sender has to keep the ineligible task and hold all resources that the task needs, thereby delaying the execution of eligible tasks, which is undesirable if tasks have time constraints.

2.3 Byzantine Attacks in REQ Message Propagation

BHC attackers try to slow or even cease REQ message propagation, while MF attackers try to spread incorrect service ID or fake REP messages. Figure 1(a) shows BHC and MF attacks in REQ message propagation.

If an MF attacker spreads a fake requested service ID, it can activate an irrelevant receiver to gossip back, and thus increase the message overhead in the network. If the attacker directly spreads a fake REP message, it may tempt the sender to give wrong responses.

If a Byzantine node is "intelligent" as discussed in Section 1, it will not execute the second MF attack. An important feature of gossip is that every node will finally receive the (attacker's fake) messages, with a high probability [7]. If the original REQ sender receives a fake REQ message with an incorrect requested service ID, it cannot identify the attacker by comparing it with the real requested service ID. This is because the intermediate node IP in a REP message might only be a transferring node's IP, and not the original MF attacker's IP. However, if the real receiver gets a fake REP message, it may easily identify the attacker (by comparing it with the real answer). Therefore, "intelligent" Byzantine nodes will not execute such an MF attack in REQ message propagation.



(a) Attacks in REQ Message Propagation (b) Attacks in REP Message Propagation

Fig. 1. Byzantine Attacks in REQ and REP Message Propagation

2.4 Byzantine Attacks in REP Message Propagation

Figure 1(b) shows BHC and MF attacks in REP message propagation. BHC attacks stop forwarding REP messages, while MF attacks spread REP messages with fake answer (NO for a YES or vice versa).

Unlike in REQ message propagation, an "intelligent" attacker can execute MF attacks in REP message propagation without being identified by other nodes. Although the real receiver can finally get fake REP messages and can identify this attack by comparing the REP message answer with its own, it cannot identify the fake message initiator. This is because, it is quite possible that the sender in a REP message is only an intermediate node, and this node can be a victim cheated by a real MF attacker.

Note that MF attackers cannot mislead the real receiver by faking requested service ID in REQ message propagation. However, they can directly disturb REP message propagation. Therefore, MF attack on REP message propagation is more dangerous than that on REQ message propagation. Thus, we focus on how to guarantee a correct REP message propagation.

3 Byzantine Attacker Detection

In this section, we present BHC Attacker Detector (BHCAD) and MF Attacker Detectors (MFADs).

BHCAD. BHC attackers behave like failed nodes — they receive, but do not forward messages. BHCAD acts as a failure detector.

Algorithm 1: BHC Attacker Detector

```
1  Each node i maintains a BHCAD list;
2  for each known node j do
3      combine i.heartbeat_counter with i's address;
4  for each gossip round do
5      i.heartbeat_counter++;
6      combine i's list with gossip messages;
7      GOSSIP(messages);
8  Upon receiving a message from node k:
9    for each member j in BHCAD list do
10       j.heartbeat_counter ← max{i's, k's};
11   if j.heartbeat_counter does not change after gossip finishes then
12       i considers j as failed;
```

BHCAD operates as follows: Each node maintains a BHCAD list, which contains a heartbeat counter for each known node. When gossiping to others, each node increases its own heartbeat counter and sends the list with gossip messages. If the node does not receive any change in another node's heartbeat counter

Algorithm 2: Gossip Emission (GOSSIP() Function)

```
1 On gossiping a message msg:
2   while R != 0 do
3       Every gossip round Γ, randomly select F target nodes;
4       for each m ∈ [1, ..., F ] do
5           SEND(target_i, msg);
6       R = R – 1;
```

during the entire gossip period, it will consider that node as failed. Algorithm 1 describes BHCAD. The GOSSIP() procedure invoked in line 7 of Algorithm 1 is shown in Algorithm 2.

MFAD. MFAD utilizes MF attackers' feature of faking requested service ID in REQ messages, or faking the answer in REP messages. To execute MF attacker detection, the first node that is not an MF attacker must initiate its own MFAD before others. (We call a node that is not an MF attacker as a "healthy node.")

Algorithm 3: Initiating MF Attacker Detector

```
1 Node i puts its own service ID in "REQ" messages;
2 Node i sets R = 1;
3 At the first gossip round: Broadcast "REQ" messages in the system;
4 After the second gossip round: Check service ID in every received "REQ" message;
5 if the service ID is changed then
6     Identify this message sender as an MF attacker;
```

To detect MF attackers faking requested service ID, the initiating MFAD broadcasts "REQ" messages in the system, but includes its own service ID in them and sets the gossip round R to 1. MF attackers will change this service ID. Since $R = 1$, an attacker must broadcast fake "REQ" messages to all other nodes in one gossip round. Therefore, the initiating MFAD can easily identify an MF attacker by checking the requested service ID in every received "REQ" message. For those activated MFADs located on other "healthy" nodes, since they receive the original "REQ" message from the initiating "healthy" node, they can also identify MF attackers by comparing every received "REQ" message with the original one. MFADs are described in Algorithms 3 and 4.

Algorithm 4: Activated MF Attacker Detector

```
1 At the first gossip round: Receive "REQ" messages from the initiating MFAD;
2 At the second gossip round: Broadcast "REQ" messages once;
3 After the second gossip round: Compare service ID in every received "REQ" message with the one in
  the original message;
4 if the service ID is changed then
5     Identify this message sender as an MF attacker;
```

To detect MF attackers faking the answer in REP messages, the initiating MFAD broadcasts "REP" messages in the system, includes its preset answer (e.g. "YES") in them, and sets $R = 1$. MF attackers will reverse the answer (e.g. "NO"). Since $R = 1$, an attacker must broadcast fake "REP" messages to all other nodes in one gossip round. Similar to MFAD in REQ message propagation, both initiating and activated MFAD can easily identify an MF attacker. Since MFADs in REP message propagation behave similar to those in REQ message propagation, we do not separately show their algorithm descriptions.

If an activated MFAD does not receive the original "REQ" or "REP" message from the initiating MFAD, it cannot identify any MF attacker later. If an MFAD does not receive the one-time broadcast

"REQ" or "REP" message from a node, it cannot regard that node as an MF attacker. Therefore, the effectiveness of an MFAD depends on successful point-to-point message propagation.

4 LASIRC Model and Mechanism

BHCAD and MFADs cannot exhaust attackers if message loss rate is larger than zero, which is common in unreliable networks. Since gossip is robust to message losses and node failures, it is relatively easy to deal with hiding BHC attackers. However, gossip cannot handle hiding MF attackers, which is more dangerous. Therefore, it is necessary to design a gossip-based propagation model/mechanism to defend against MF attacks.

4.1 LASIRC Model

As stated in Section 2.3, due to the REQ message structure, it is difficult to identify an MF attacker that fakes service ID in REQ message propagation. In addition, though such an attack increases communication overhead, it does little harm to the real REQ message propagation process. Therefore, we focus on modeling node behaviors in REP message propagation.

We introduce new definitions for nodes participating in REP message propagation:

Definition 1 (Healthy Node) *A node that is not an MF attacker.*

Definition 2 (Host (H)) *A node that has received one or more REP messages.*

Definition 3 (Launcher (L)) *The initiating sender of the REP messages.*

Definition 4 (Attacker (A)) *An MF attacker that tries to spread a fake answer.*

Definition 5 (Susceptible (S)) *A healthy node that has not received any REP message.*

Definition 6 (Infective (I)) *A healthy host that has a fake answer.*

Definition 7 (Removed (R)) *A healthy host that knows its answer is correct, or always sends correct REP messages.*

Definition 8 (Consumer (C)) *The initiating sender of REQ messages.*

We introduce definitions for REP messages:

Definition 9 (Agent) *A fake REP message from an attacker.*

Definition 10 (Virus) *A fake REP message that possibly turns a susceptible into an infective.*

Definition 11 (Antibiotic) *An agent that turns a susceptible/infective into a removed.*

Definition 12 (Berry) *A correct REP message.*

Definition 13 (Vaccine) *A berry that possibly turns a susceptible into a removed.*

We refer to this model as the LASIRC model. Note that we assume that every healthy node only checks the first received REP message, discarding all of the following messages with the same message ID, unless the message is an antibiotic, or contains a change defined in the REP message definitions.

A host is said to be immune to an attacker, if its MFAD has identified that attacker. A host gets immunized when its first received message is a vaccine.

4.2 LASIRC Mechanism

We now describe the LASIRC mechanism. Algorithm 5 shows how a launcher works. A launcher is the initiating sender of REP messages. It holds the correct answer, so it cannot be infected by an MF attacker.

Algorithm 5: Launcher

```
1 Initialize REP message rep;
2 GOSSIP(rep);
```

When an MF attacker receives a REP message, it is activated. If the sender is another attacker, it follows the sender's answer. Otherwise, it reverses the answer ("NO" for a "YES"). Algorithm 6 describes the MF attacker.

Algorithm 6: MF Attacker

```
1 On receiving a REP message rep:
2   if the sender is not an MF attacker then reverse the answer in rep;
3   GOSSIP(rep);
```

Algorithm 7 shows healthy node behaviors in the LASIRC mechanism. A susceptible turns into a removed if its first received message is a vaccine (from a removed), or turns into an infective if that message is a virus (from an infective or an MF attacker).

Consumer behavior in the LASIRC mechanism is shown in Algorithm 8. A consumer acts as other healthy nodes during gossip rounds. After gossip finishes, if it still cannot identify itself as a removed (knowing the correct answer), it will count the number of the same answers in received REP messages. If the consumer is optimistic, it believes MF attackers occupy less than half of the total number of nodes. Then, it will select the answer in most received REP messages. Otherwise, the consumer is pessimistic, and it will select the answer in less received REP messages.

Algorithm 7: Susceptible, Infective and Removed

```
1  On receiving the first REP message rep:
2    GOSSIP(rep);
3  On receiving another REP message with the same message ID:
4    if the sender has sent REP message before then
5      if the answer changes then
6        adopt answer in new REP message; //Identify sender has changed from an infective to a
           removed
7        reverse the answer in rep;
8        GOSSIP(rep); //Change from an infective to a removed
9    if the sender is an identified MF attacker then
10     if the answer in the first rep is the same as the one in this attacker's message then
11       reverse the answer in rep;
12       GOSSIP(rep); //Change from an infective to a removed
```

Algorithm 8: Consumer

```
In gossip: Act as other healthy nodes do in Algorithm 7
After gossip finishes:
  if the consumer has not identified itself as a removed then
      select the answer in most(less) received REP messages; //Optimistic (Pessimistic) consumer
```

5 LASIRC Analysis

Lemma 1 *A REP message from an infective to a susceptible is a virus.*

Proof. According to Definition 6, an infective is a healthy node, not an attacker. As previously stated, a susceptible is only immune to a part of attackers. It cannot distinguish fake REP messages from healthy nodes. If the first REP message it receives is a fake one from an infective, it will adopt the wrong answer. According to Definition 10, that first received message is a virus.

Lemma 2 *An agent turns into a virus if the receiving healthy host is not immune to the attacker, or an antibiotic if it is immune to that attacker.*

Proof. The former holds if a host is not immune to the attacker that sends the agent. Otherwise, if an attacker has been identified, the answer in its agent will be reversed by the receiving susceptible/infective. According to Definitions 7 and 11, the latter holds.

Theorem 3 *A berry to a susceptible is a vaccine.*

Proof. If the first received message is a berry, a susceptible will adopt this correct answer and ignore all following berries. According to Lemmas 2 and 3, it may receive viruses from attackers and infectives, but they won't infect it since the berry arrives first. According to Lemma 3, it may receive antibiotics. That makes it believe that its answer is correct and does not change the answer. According to Definition 7, the berry is a vaccine.

Theorem 4 *If a susceptible/infective changes its status, this change is irreversible.*

Proof. According to Definition 2, a susceptible will become a host. Suppose this change is reversible. That means a host does not receive any REP message. This contradicts Definition 2.

According to Definitions 3–7 and Lemma 2, the only possible status change of an infective is to become a removed. Suppose this change is reversible. This means that a removed does not know if its answer is correct, or may send fake REPLY messages. This contradicts Definition 7.

6 Experimental Studies

We simulated the LASIRC mechanism in a 100-node system, in which every node is reachable to others. We considered a scenario where BHC attacks resulted in 15% of message loss during propagation, and compared LASIRC with the Susceptible-Infective (SI) mechanism, which is the gossip protocol without any MF attacker detection and protection methods. The number of gossip rounds (R) and the fan-out number (F) was 10.

Figure 2 shows the Detection Ratio (DR; the ratio of number of MF attackers detected by a node to the total number of MF attackers) along with the message loss ratio (MLR). We observe that DR on both the detection initiating detector

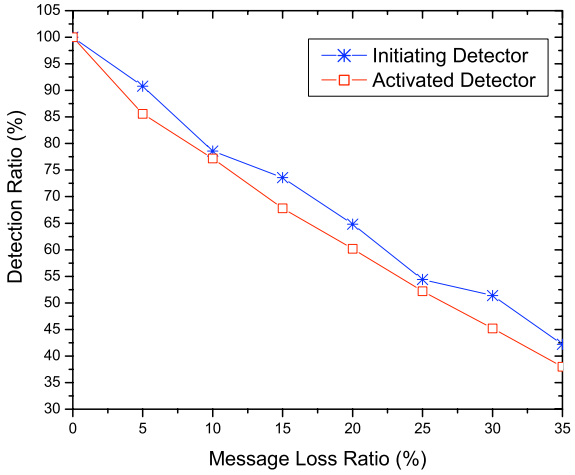
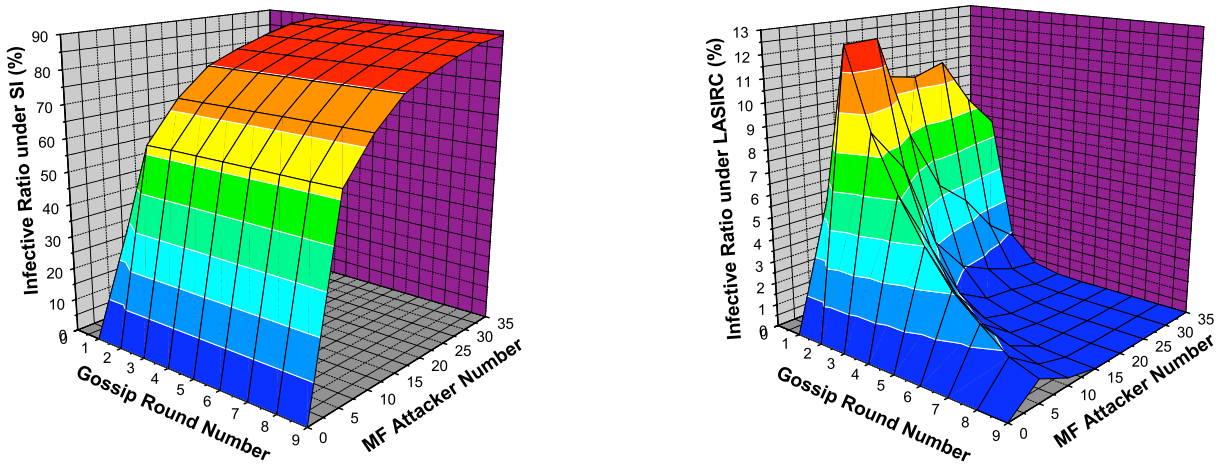


Fig. 2. LASIRC MF Attacker Detection

and activated detectors decrease when MLR increases, because MFAD depends on received messages; thus its performance degrades when MLR increases. In addition, we observe that the initiating detector's DR is higher than that of activated detectors. This is because, the initiating detector knows the correct answer before the detection begins, and activated detectors need to receive messages (message transmissions may suffer message losses) from the initiating one to know the correct answer. We also observe that MFAD cannot detect all MF attackers if MLR is larger than 0. Therefore, we need LASIRC mechanism to deal with hiding MF attackers that survive this detection.



(a) Infective Ratio under SI

(b) Infective Ratio under LASIRC

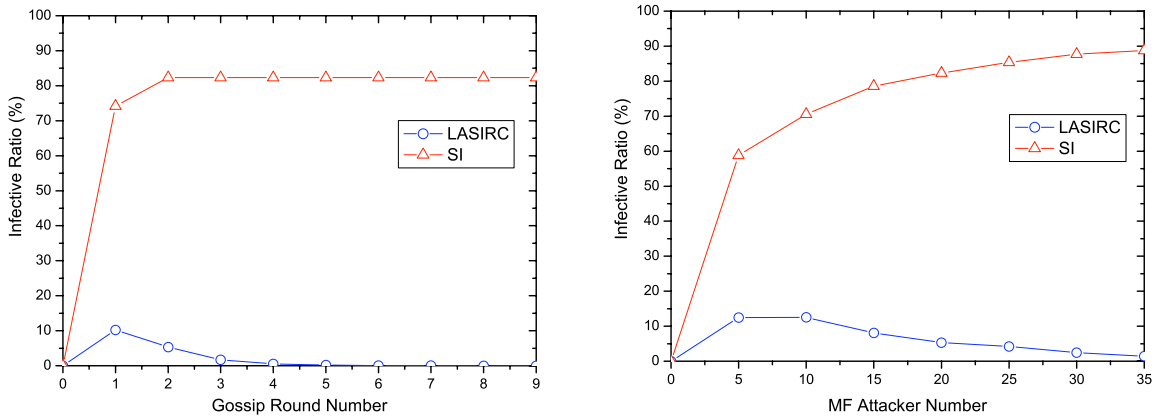
Fig. 3. Infective Ratio under LASIRC/SI Model

Figure 3 shows Infective Ratio (IR; the ratio of number of infectives to the number of healthy nodes) along with the MF attacker number (N) and R. From Figure 3(a), we observe that as R and N increase, IR sharply increases to nearly 90%. However, in Figure 3(b), IR decreases to nearly zero when R increases to 10, and generally decreases as N increases.

We can make a clearer observation in Figure 4(a) that shows the cross-section of Figure 3. We observe that as R increases, at the first round, LASIRC's IR moderately increases from 0 to 12.84%, while SI's IR dramatically increases from 0 to 74.52%. This is because LASIRC MFADs make nodes identify a number of MF attackers and help them turn viruses into antibiotics. In comparison, SI does not have MFADs, and nodes immediately get infected if they receive viruses. In the later rounds, we observe that LASIRC's IR quickly decreases to near zero ($IR = 0.01\%$ when $R = 9$), while SI's IR almost remains unchanged ($IR = 82.35\%$ when $R = 9$). With Algorithms 7 and 8, the LASIRC mechanism automatically turns infectives into removeds during message propagation. Since SI does not have these methods, it cannot convert an infective into a removed, and its number of infective nodes keeps increasing till there is no susceptible.

Figure 4(b) shows IR along with (N). We observe that SI's IR dramatically increases as N increases ($IR = 88.76\%$ when $N = 35$). However, LASIRC's IR slightly increases and then decreases to almost zero ($IR = 1.07\%$ when $N = 35$). This shows the effectiveness of LASIRC MFADs — if there are more MF nodes, MFADs identify more MF attackers, and then more viruses are turned into antibiotics during message propagation. This is why LASIRC's IR decreases when the attacker number increases.

Figure 5 shows the consumer Correctness Ratio (CR; the number of times a consumer gets the correct answer after gossip finishes over the total simulation time) with an optimistic consumer. We also show CRs of SI, LASIRC without Consumer Decision Mechanism (CDM, Algorithm 8), LASIRC without CDM and Sender Identification Area Mechanism (SIM, part of Algorithm 7), as comparisons.



(a) Infective Ratio with Increasing Rounds (20 Attackers) (b) Infective Ratio with Increasing Attackers (2nd Round)

Fig. 4. Infective Ratio under LASIRC/SI Model (cross section)

We observe that LASIRC gets a satisfactory result. With N increases, LASIRC's CR remains almost 100%, while SI's CR dramatically decreases to 4.97% ($N = 35$). With MFADs, CDM and SIM, LASIRC mechanism guarantees that the consumer can finally get the correct answer after gossip finishes. The other two curves indicate the function of each LASIRC method. We observe that the CR of LASIRC without CDM and SIM is much better than that of SI.

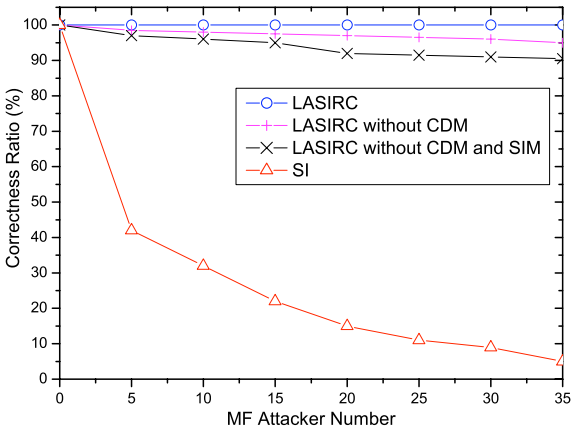


Fig. 5. Consumer Correctness Ratio

This comparison illustrates the importance of MFADs to LASIRC. MFADs help turn a virus message into an antibiotic message, which largely reduces the number of infectives. We also observe that LASIRC without CDM is better than LASIRC without CDM and SIM — this demonstrates that SIM helps to enhance the CR. Also, LASIRC is better than all the others. If other methods cannot help, CDM (Algorithm 8) finally determines the answer. To determine the final answer, we need to set a consumer to be optimistic or pessimistic. This setting depends on the number of MF attackers in the system. We can conclude that MFADs provide a major contribution to CR, and CDM and SIM improve CR to almost 100%.

7 Conclusions and Future Work

We presented a Byzantine-tolerant, gossip-based point-to-point information propagation model/mechanism called LASIRC. With BHC and MF attacker detectors, LASIRC can detect Byzantine attackers before information propagation begins. LASIRC is robust to BHC attacks because of its gossip feature. In addition, CDM and SIM methods help LASIRC largely enhance its performance under MF attacks. Our experimental studies verify the effectiveness of the LASIRC mechanism.

Directions for future work include extending LASIRC to deal with MF attack that fakes requested service ID in REQ message propagation, and considering propagating information under time constraints.

References

1. Awerbuch, B., Curtmola, R., et al.: Mitigating byzantine attacks in ad hoc wireless networks. Technical Report I, JHU CS Dept. (March 2004)
2. Harry C. Li, A.C., et al.: Bar gossip. In: USENIX OSDI. (November 2006) 191–204
3. Minsky, Y.M., Schneider, F.B.: Tolerating malicious gossip. Distributed Computing **16**(1) (February 2003) 49–68
4. Han, K., et al.: Exploiting slack for scheduling dependent, distributable real-time threads in mobile ad hoc networks. In: 15th International Conference on Real-Time and Network Systems (RTNS'07). (2007)
5. CCRP: Network centric warfare. `http://www.dodccrp.org/html2/research_ncw.html` Last accessed, May 2006.
6. Pittel, B.: On spreading a rumor. SIAM Journal on Applied Mathematics **47**(1) (February 1987) 213 – 223
7. Birman, K.P., Hayden, M., et al.: Bimodal multicast. ACM TOCS **17**(2) (1999) 41–88