Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 3, March 2014, pg.1149 – 1160

RESEARCH ARTICLE

Defending against Flood Attacks in Disruption Tolerant Networks using Blowfish Algorithm

S. Pushpa[1], B. Mani Megalai[2], A. Wisy Shantha[3]

[1]PG Scholar of CSE Department& Anna University, India

[2]Mater of Computer Applications & Anna Universities, India

[3]PG Scholar of CSE Department &Department & University, India

Pushpaj85@gmail.com; mani17mca@gmail.com; Wisyabraham22@gmail.com

Abstract— Disruption Tolerant Networks (DTNs) utilize the mobility of nodes and the opportunistic contacts among nodes for data communications. Due to the limitation in network resources such as contact opportunity and buffer space, DTNs are vulnerable to flood attacks in which attackers send as many packets or packet replicas as possible to the network, in order to deplete or overuse the limited network resources. In this paper, we employ rate limiting to defend against flood attacks in DTNs, such that each node has a limit over the number of packets that it can generate in each time interval and a limit over the number of replicas that it can generate for each packet. We propose a distributed scheme to detect if a node has violated its rate limits. To address the challenge that it is difficult to count all the packets or replicas sent by a node due to lack of communication infrastructure, our detection adopts claim-carry-and-check: each node itself counts the number of packets or replicas that it has sent and claims the count to other nodes; the receiving nodes carry the claims when they move, and cross-check if their carried claims are inconsistent when they contact. The claim structure uses the pigeonhole principle to guarantee that an attacker will make inconsistent claims which may lead to detection. We provide rigorous analysis on the probability of detection, and evaluate the effectiveness and efficiency of our scheme with extensive trace-driven simulations.

Keywords— DTN; security; flood attack; detection

I. INTRODUCTION

Disruption Tolerant Networks (DTNs) [1] consist of mobile nodes carried by human beings [2], [3], vehicles [4], [5], etc. DTNs enable data transfer when mobile nodes are only intermittently connected, making them appropriate for applications where no communication infrastructure is available such as military scenarios and rural areas. Due to lack of consistent connectivity, two nodes can only exchange data when they move into the transmission range of each other (which is called a contact between them). DTNs employ such contact opportunity for data forwarding with “store-carry-and-forward”; i.e., when a node receives some packets, it stores these packets in its buffer, carries them around until it contacts another node, and then forwards them. Since the contacts between nodes are opportunistic and the duration of a contact may be short because of mobility, the usable bandwidth which is only available during the opportunistic contacts is a limited resource. Also, mobile nodes may have limited buffer space.

Due to the limitation in bandwidth and buffer space, DTNs are vulnerable to flood attacks. In flood attacks, maliciously or selfishly motivated attackers inject as many packets as possible into the network, or instead of injecting different packets the attackers forward replicas of the same packet to as many nodes as possible. For convenience, we call the two types of attack packet flood attack and replica flood attack, respectively. Flooded packets and replicas can waste the precious bandwidth and buffer resources, prevent benign packets from being forwarded and thus degrade the network service provided to good nodes. Moreover, mobile nodes spend much energy on transmitting/receiving flooded packets and replicas which may shorten their battery life. Therefore, it is urgent to secure DTNs against flood attacks.

Although many schemes have been proposed to defend against flood attacks on the Internet [6] and in wireless sensor networks [7], they assume persistent connectivity and cannot be directly applied to DTNs that have intermittent connectivity. In DTNs, little work has been done on flood attacks, despite the many works on routing [8], [4], [36], data dissemination [9], [37], black hole attack [10], wormhole attack [11], and selfish dropping behavior [12], [13]. We noted that the packets flooded by outsider attackers (i.e., the attackers without valid cryptographic credentials) can be easily filtered with authentication techniques (e.g., [14]). However, authentication alone does not work when insider attackers (i.e., the attackers with valid cryptographic credentials) flood packets and replicas with valid signatures. Thus, it is still an open problem is to address flood attacks in DTNs.

In this paper, we employ rate limiting [15] to defend against flood attacks in DTNs. In our approach, each node has a limit over the number of packets that it, as a source node, can send to the network in each time interval. Each node also has a limit over the number of replicas that it can generate for each packet (i.e., the number of nodes that it can forward each packet to). The two limits are used to mitigate packet flood and replica flood attacks, respectively. If a node violates its rate limits, it will be detected and its data traffic will be filtered. In this way, the amount of flooded traffic can be controlled.

Our main contribution is a technique to detect if a node has violated its rate limits. Although it is easy to detect the violation of rate limit on the Internet and in telecommunications networks where the egress router and base station can account each user’s traffic, it is challenging in DTNs due to lack of communication infrastructure and consistent connectivity. Since a node moves around and may send data to any contacted node, it is very difficult to count the number of packets or replicas sent out by this node. Our basic idea of detection is claim-carry-and-check. Each node itself counts the number of packets or replicas that it has sent out, and claims the count to other nodes; the receiving nodes carry the claims around when they move, exchange some claims when they contact, and cross-check if these claims are inconsistent. If an attacker floods more packets or replicas than its limit, it has to use the same count in more than one claim according to the pigeonhole principle,1 and this inconsistency may lead to detection. Based on this idea, we use different cryptographic constructions to detect packet flood and replica flood attacks.

Because the contacts in DTNs are opportunistic in nature, our approach provides probabilistic detection. The more traffic an attacker floods, the more likely it will be detected. The detection probability can be flexibly adjusted by system parameters that control the amount of claims exchanged in a contact. We provide a lower and upper bound of detection probability and investigate the problem of parameter selection to maximize detection probability under a certain amount of exchanged claims. The effective-ness and efficiency of our scheme are evaluated with extensive trace-driven simulations.

This paper is structured as follows. Section 2 motivates our work. Section 3 presents our models and basic ideas. Sections 4 and 5 present our scheme. Section 6 presents security and cost analysis. Section 7 presents simulation results. The last two sections present related work and conclusions, respectively.

II. Motivation

A. The Potential Prevalence of Flood Attacks

Many nodes may launch flood attacks for malicious or selfish purposes. Malicious nodes, which can be the nodes deliberately deployed by the adversary or subverted by the adversary via mobile phone worms [16], launch attacks to congest the network and waste the resources of other nodes.

Selfish nodes may also exploit flood attacks to increase their communication throughput. In DTNs, a single packet usually can only be delivered to the destination with a probability smaller than 1 due to the opportunistic connectivity. If a selfish node floods many replicas of its own packet, it can increase the likelihood of its packet being delivered, since the delivery of any replica means successful delivery of the packet. With packet flood attacks, selfish nodes can also increase their throughput, albeit in a subtler manner. For example, suppose Alice wants to send a packet to Bob. Alice can construct 100 variants of the original packet which only differ in one unimportant padding byte, and send the 100 variants to Bob independently. When Bob receives any one of the 100 variants, he throws away the padding byte and gets the original packet.

B. The Effect of Flood Attacks

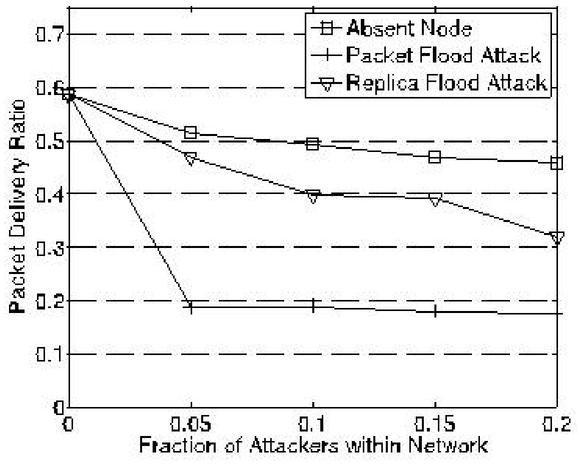
To study the effect of flood attacks on DTN routing and motivate our work, we run simulations on the MIT Reality trace [17] (see more details about this trace in Section 7).

We consider three general routing strategies in DTNs. 1) Single-copy routing (e.g., [18], [8]): after forwarding a packet out, a node deletes its own copy of the packet. Thus, each packet only has one copy in the network. 2) Multicopy routing (e.g., [19]): the source node of a packet sprays a certain number of copies of the packet to other nodes and each copy is individually routed using the single-copy strategy. The maximum number of copies that each packet can have is fixed. 3) Propagation routing (e.g., [17], [20], [21]): when a node finds it appropriate (according to the routing algorithm) to forward a packet to another encountered node, it replicates that packet to the encountered node and keeps its own copy. There is no preset limit over the number of copies a packet can have. In our simulations, SimBet [8], Spray-and-Focus [19] (three copies allowed for each packet) and Propagation are used as representatives of the three routing strategies, respectively. In Propagation, a node replicates a packet to another encountered node if the latter has more frequent contacts with the destination of the packet.

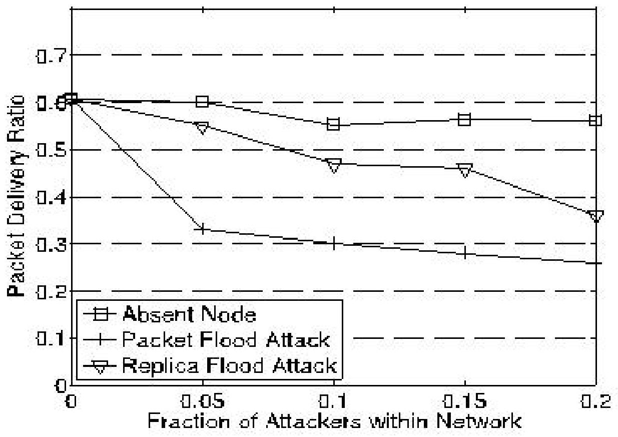
Two metrics are used; the first metric is packet delivery ratio, which is defined as the fraction of packets delivered to their destinations out of all the unique packets generated. The second metric is the fraction of wasted transmissions (i.e., the transmissions made by good nodes for flooded packets). The higher fractions of wasted transmissions, the more network resources are wasted. We noticed that the effect of packet flood attacks on packet delivery ratio has been studied by Burgess et al. [22] using a different trace [4]. Their simulations show that packet flood attacks significantly reduce the packet delivery ratio of single-copy routing but do not affect propagation routing much. However, they do not study replica flood attacks and the effect of packet flood attacks on wasted transmissions.

In our simulations, a packet flood attacker floods packets destined to random good nodes in each contact until the contact ends or the contacted node's buffer is full. A replica flood attacker replicates the packets it has generated to every encountered node that does not have a copy. Each good node generates thirty packets on the 121st day of the Reality trace, and each attacker does the same in replica flood attacks. Each packet expires in 60 days. The buffer size of each node is 5 MB, bandwidth is 2 Mbps and packet size is 10 KB.

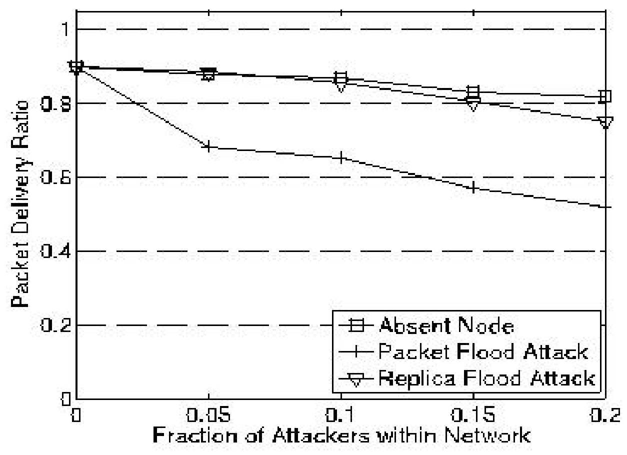
Fig. 1 shows the effect of flood attacks on packet delivery ratio. Packet flood attack can dramatically reduce the packet delivery ratio of all three types of routing. When the fraction of attackers is high, replica flood attack can significantly decrease the packet delivery ratio of single-copy and multicopy routing, but it does not have much effect on propagation routing.



(a) Single-copy Routing



(b) Multi-copy Routing



(c) Propagation Routing

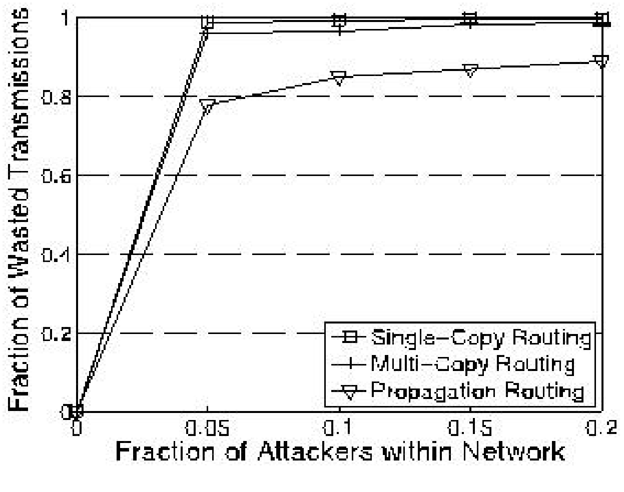
Fig. 2 shows the effect of flood attacks on wasted transmission. Packet flood attack can waste more than 80 percent of the transmissions made by good nodes in all routing strategies when the fraction of attackers is higher than 5 percent. When 20 percent of nodes are attackers, replica flood attack can waste 68 and 44 percent of transmissions in single-copy and multicopy routing, respectively. However, replica flood attack only wastes 17 percent of transmissions in propagation routing. This is because each good packet is also replicated many times.

III. OVERVIEW

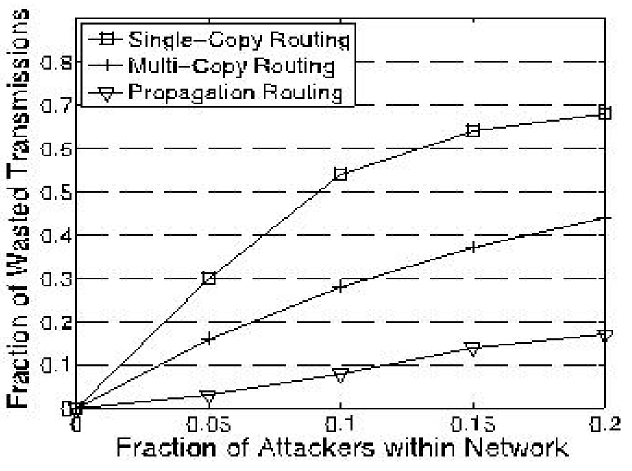
A. Problem Definition

1 Defense against Packet Flood Attacks

We consider a scenario where each node has a rate limit L on the number of unique packets that it as a source can generate and send into the network within each time interval T . The time intervals start from time 0, T , 2T , etc. The packets generated within the rate limit are deemed legitimate, but the packets generated beyond the limit are deemed flooded by this node. To defend against packet flood attacks, our goal is to detect if a node as a source has generated and sent more unique packets into the network than its rate limit L per time interval.



(a) Packet Flood Attack



(b) Replica Flood Attack

Fig. 2. The effect of flood attacks on th the fraction of wasted transmission. Attackers are randomly deployed.

A node's rate limit L does not depend on any specific routing protocol, but it can be determined by a service contract between the node and the network operator as discussed in Section 3.1.3. Different nodes can have different rate limits and their rate limits can be dynamically adjusted.

The length of time interval should be set appropriately. If the interval is too long, rate limiting may not be very effective against packet flood attacks. If the interval is too short, the number of contacts that each node has during one interval may be too nondeterministic and thus it is difficult to set an appropriate rate limit. Generally speaking, the interval should be short under the condition that most nodes can have a significant number of contacts with other nodes within one interval, but the appropriate length depends on the contact patterns between nodes in the specific deployment scenario.

2 *Defense against Replica Flood Attacks*

As motivated in Section 2, the defense against replica flood considers single-copy and multicopy routing protocols. These protocols require that, for each packet that a node buffers no matter if this packet has been generated by the node or forwarded to it, there is a limit l on the number of times that the node can forward this packet to other nodes. The values of l may be different for different buffered packets. Our goal is to detect if a node has violated the routing protocol and forwarded a packet more times than its limit l for the packet.

A node's limit l for a buffered packet is determined by the routing protocol. In multicopy routing, l ¼ L0 (where L0 is a parameter of routing) if the node is the source of the packet, and l ¼ 1 if the node is an intermediate hop (i.e., it received the packet from another node). In single-copy routing, l ¼ 1 no matter if the node is the source or an intermediate hop. Note that the two limits L and l do not depend on each other.

We discuss how to defend against replica flood attacks for quota-based routing [23], [19], [24] in Section 4.9.

3 *Setting the Rate Limit L*

One possible method is to set L in a request-approve style. When a user joins the network, she requests for a rate limit from a trusted authority which acts as the network operator. In the request, this user specifies an appropriate value of L based on prediction of her traffic demand. If the trusted authority approves this request, it issues a rate limit certificate to this user, which can be used by the user to prove to other nodes the legitimacy of her rate limit. To prevent users from requesting unreasonably large rate limits, a user pays an appropriate amount of money or virtual currency (e.g., the credits that she earns by forwarding data for other users [25]) for her rate limit. When a user predicts an increase (decrease) of her demand, she can request for a higher (lower) rate limit. The request and approval of rate limit may be done offline. The flexibility of rate limit leaves legitimate users' usage of the network unhindered. This process can be similar to signing a contract between a smartphone user and a 3G service provider: the user selects a data plan (e.g., 200 MB/month) and pays for it; she can upgrade or downgrade the plan when needed.

IV. **OUR SCHEME**

Our scheme uses two different cryptographic construc-tions to detect packet flood and replica flood attacks independently. When our scheme is deployed to propaga-tion routing protocols, the detection of replica flood attacks is deactivated.

The detection of packet flood attacks works indepen-dently for each time interval. Without loss of generality, we only consider one time interval when describing our scheme. For convenience, we first describe our scheme assuming that all nodes have the same rate limit L, and relax this assumption in Section 4.8. In the following, we use SIGið_Þ to denote node i's signature over the content in the brackets.

A. Inconsistency Caused by Attack

In a dishonest P-claim, an attacker uses a smaller packet count than the real value. (We do not consider the case where the attacker uses a larger packet count than the real value, since it makes no sense for the attacker.) However, this packet count must have been used in another P-claim generated earlier. This causes an inconsistency called count reuse, which means the use of the same count in two different P-claims generated by the same node. For example in Fig. 3a the count value 3 is reused in the P-claims of packet m3 and m4. Similarly, count reuse is also caused by dishonest T-claims.

B. Protocol

Suppose two nodes contact and they have a number of packets to forward to each other. Then our protocol is sketched in Algorithm 1.

Algorithm 1. The protocol run by each node in a contact

```
1. Metadata (P-claim and T-claim) exchange and attack detection
2. if Have packets to send then
3. For each new packet, generate a P-claim;
4. For all packets, generate their T-claims and sign them with a hash tree;
5. Send every packet with the P-claim and T-claim attached;
6. end if
7. if Receive a packet then
8. if Signature verification fails or the count value in its P-claim or T-claim is invalid then
9. Discard this packet;
10. end if
11. Check the P-claim against those locally collected and generated in the same time interval to detect
    inconsistency;
12. Check the T-claim against those locally collected for inconsistency;
13. if Inconsistency is detected then
14. Tag the signer of the P-claim (T-claim, respec- denotes the size of a hash (e.g., 256 for SHA-256). W
    keeps tively) as an attacker and add it into a blacklist;
15. Disseminate an alarm against the attacker to the network;
16. else
17. Store the new P-claim (T-claim, respectively);
18. end if
19. end if
```

When a node forwards a packet, it attaches a T-claim to the packet. Since many packets may be forwarded in a contact and it is expensive to sign each T-claim separately, an efficient signature construction is proposed in Section 4.7. The node also attaches a P-claim to the packets that are generated by itself and have not been sent to other nodes before (called new packet in line 3, Algorithm 1).

When a node receives a packet, it gets the P-claim and T-claim included in the packet. It checks them against the claims that it has already collected to detect if there is any inconsistency (see Section 4.5). Only the P-claims generated in the same time interval (which can be determined by the time tag) are cross-checked. If no inconsistency is detected, this node stores the P-claim and T-claim locally (see Section 4.4).

To better detect flood attacks, the two nodes also exchange a small number of the recently collected P-claims and T-claims and check them for inconsistency. This metadata exchange process is separately presented in Section 5.

When a node detects an attacker, it adds the attacker into a blacklist and will not accept packets originated from or forwarded by the attacker. The node also disseminates an alarm against the attacker to other nodes (see Section 4.6).

V. METADATA EXCHANGE

When two nodes contact they exchange their collected P-claims and T-claims to detect flood attacks. If all claims are exchanged, the communication cost will be too high. Thus, our scheme uses sampling techniques to keep the communication cost low. To increase the probability of attack detection, one node also stores a small portion of claims exchanged from its contacted node, and exchanges them to its own future contacts. This is called redirection.

A. Sampling

Since P-claims and T-claims are sampled together (i.e., when a P-claim is sampled the T-claim of the same packet is also sampled), in the following we only consider P-claims.

A node may receive a number of packets (each with a P-claim) in a contact. It randomly samples Z (a system parameter) of the received P-claims, and exchanges the sampled P-claims to the next K (a system parameter) different nodes it will contact, excluding the sources of the P-claims and the previous hop from which these P-claims are received.

However, a vulnerability to tailgating attack should be addressed. In tailgating attack, one or more attackers tailgate a good node to create a large number (say, d) of frequent contacts with this node, and send Z packets (not necessarily generated by the attackers) to this node in each created contact. If this good node sends the Zd P-claims of these contacts to the next K good nodes it contacts, much effective bandwidth between these good nodes will be wasted, especially in a large network where K is not small.

To address this attack, the node uses an inter-contact sampling technique to determine which P-claims sampled in historical contacts should be exchanged in the current contact. Let SK denote a set of contacts. This set includes the minimum number of most recent contacts between this node and at least K other different nodes. Within this set, all the contacts with the same node are taken as one single contact and a total of Z P-claims are sampled out of these contacts. This technique is not vulnerable to the tailgating attack since the number of claims exchanged in each contact is bounded by a constant.

B. Redirection

There is a stealthy attack to flood attack detection. For replica flood attacks, the condition of detection is that at least two nodes carrying inconsistent T-claims can contact. However, suppose the attacker knows that two nodes A and B never contact. Then, it can send some packets to A, and invalidly replicate these packets to B. In this scenario, this attacker cannot be detected since A and B never contact. Similarly, the stealthy attack is also harmful for some routing protocols like Spray-and-Wait [19] in which each packet is forwarded from the source to a relay and then directly delivered from the relay to the destination.

To address the stealthy attack, our idea is to add one level of indirection. A node redirects the Z P-claims and T-claims sampled in the current contact to one of the next K nodes it will contact, and this contacted node will exchange (but not redirect again) these redirected claims in its own subsequent contacts. Look at the example in Fig. 6. Suppose attacker S sends mutually inconsistent packets to two nodes A and B which will never contact. Suppose A and B redirect their sampled P-claims to node C and D, respectively. Then so long as C and B or D and A or C and D can contact, the attack has a chance to be detected. Thus, the successful chance of stealthy attack is significantly reduced.

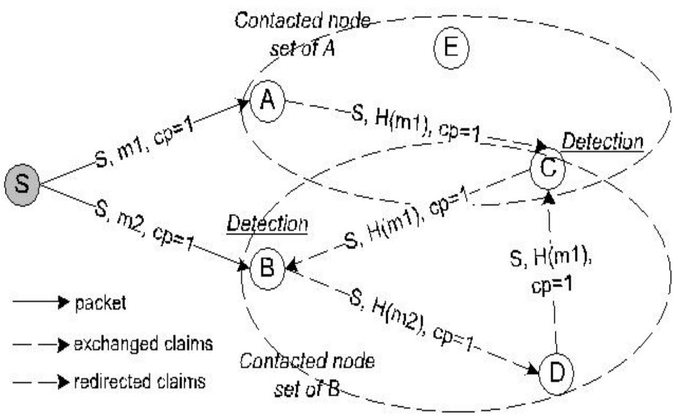


Fig. 6. The idea of redirection which is used to mitigate the stealthy attack

C. The Exchange Process

Each node maintains two separate sets of P-claims (T-claims, respectively in the following) for metadata exchange, a sampled set which includes the P-claims sampled from the most recent contacts with K different nodes (i.e., SK in Section 5.1), and a redirected set which includes the P-claims redirected from those contacts. Both sets include Z P-claims obtained in each of those contacts.

When two nodes A and B contact, they first select KZ P-claims from each set with the inter-contact sampling technique (see Section 5.1), and then send these P-claims to each other. When A receives a P-claim, it checks if this P-claim is inconsistent with any of its collected P-claims using the method described in Section 4.5. If the received P-claim is inconsistent with a locally collected one and the signature of the received P-claim is valid, A detects that the issuer (or signer) of the received P-claim is an attacker.

Out of all the P-claims received from B, A randomly selects Z of the P-claims from the sampled set of B, and stores them to A's redirected set. All other P-claims received from B are discarded after inconsistency check.

D. Metadata Deletion

A node stores the P-claims and T-claims collected from received data packets for a certain time denoted by and delete them afterward. It deletes the claims redirected from other nodes immediately after it has exchanged them to K different nodes.

VI. ANALYSIS

This section presents rigorous analysis over the security and cost of our scheme, and discusses the optimal parameter to maximize the effectiveness of flood attack detection under a certain amount of exchanged metadata per contact.

A. Detection Probability

The following analysis assumes uniform and independent contacts between nodes, i.e., at any time each node's next contacted node can be any other node with the same probability. This assumption holds for mobility models such as Random Waypoint (RWP) where the contacts between all node pairs can be modeled as i.i.d. Poisson processes [30]. When analyzing the detection probability, we assume that each attacker acts alone. The case of collusion is analyzed separately in Section 6.4.

B. The Basic Attack

First we consider a basic attack (see Fig. 7a) in which an attacker S floods two sets of mutually inconsistent packets to two good nodes A and B, respectively. Each flooded packet received by A is inconsistent with one of the flooded packets received by B. In the contacts with A and B, S also forwards some normal, not flooded, packets to A and B to make the attack harder to detect. Let y denote the proportion of flooded packets among those sent by S. For simplicity, we assume y is the same in both contacts. Suppose A and B redirect the claims sampled in the contact with S to C and D, respectively.

To consider the worst case performance, suppose the flooded packets are not forwarded from A and B to other nodes (which is the case in Spray-and-Wait [19]), i.e., only A and B have the inconsistent claims. Note that the analysis also applies to the detection of replica flood attacks.

For convenience, we define node A's (or B's) detection window as from the time it receives the flooded packets to the time it exchanges the sampled claims to K nodes, and node C's (or D's) detection window as from the time it receives the redirected claims to the time it exchanges them to K nodes. The attacker has a chance to be detected if node pairs hA; Bi, hA; Di, hC; Bi and hC; Di can contact within their detection windows. Table 1 shows the variables used in the analysis.

Lower bound. The lower bound of detection probability is obtained in the following scenario (see Fig. 7b): when B receives the packets from S, both A and C have finished their detection window. Due to the effect of sampling, the attacker can be detected

1. by A if A 2 SB and eB ¼ TRUE;
2. by A if D is a good node, A 2 SD and eB ¼ TRUE;
3. by C if C is a good node, C 2 SB and e^AB ¼ TRUE; or
4. by C if both C and D are good nodes, C 2 SD and

e^AB ¼ TRUE.

Upper bound. The upper bound of detection probability is obtained in the following scenario (see Fig. 7c): D receives the redirected claims from B not later than the time C receives the redirected claims from A, and they are the first node that A and B encounter after the contact with S. Besides the four cases that we discussed when deriving the lower bound, the attacker can also be detected

1. by B if B 2 SA and eA ¼ TRUE;
2. by B if C is a good node, B 2 SC and eA ¼ TRUE;
3. by D if D is a good node, D 2 SA and e^AB ¼ TRUE; or
4. by D if both C and D are good nodes, D 2 SC and

e^AB ¼ TRUE.

Similarly as in the lower bound case, we can obtain that the detection probability is Pd _ 2K 1Nþr Ps. Since Ps 1, an upper bound of the detection probability is

We use the number of times that a count value is reused to represent the strength of an attack. Intuitively, if each count value is used many times, the attacker floods a lot of packets or replicas. Let X denote the number of times a count value is reused. We want to derive the relation between X and the probability that the attacker will be detected.

C. Cost Analysis

1. Communication

The communication cost mainly has two parts. One part is the P-claim and T-claim transmitted with each packet, and the other part is the partial claims transmitted during metadata exchange. As to the latter, at most 4ZK P-claims and 4ZK T-claims are exchanged in each contact, with one half for sampled and the other half for redirected claims.

2. Computation

As to signature generation, a node generates one signature for each newly generated packet. It also generates one signature for all its T-claims as a whole sent in a contact. As to signature verification, a node verifies the signature of each received packet. It also verifies one signature for all the T-claims as a whole received in one contact.

3. Storage

Most P-claims and T-claims are compacted when the packets are forwarded. The Z sampled P-claims and T-claims are stored in full until the packets are forwarded or have been exchanged to K nodes, whichever is later, and then compacted. For each received packet, less than 20 bytes of compact claims are stored for time duration _.

D. Parameter Selection with Fixed Cost of Metadata Exchange

We study the following question: if we fix the communica-tion cost of metadata exchange (note that little can be done with the transmission of claims within packets), then how should we set parameter K and Z to maximize the detection probability? Note that the communication cost of metadata exchange is proportional to ZK. As to detection probability, we consider the lower-bound detection probability for the basic attack which can be written as Pd ¼ cKð1 _ ð1 _ yÞZ Þ.

VII. RELATED WORK

Our scheme bears some similarity with previous approaches (e.g., [33]) that detect node clone attacks in sensor networks. Both rely on the identification of some kind of inconsistency to detect the attacker. However, their approaches assume consistent connectivity between nodes which is unavailable in DTNs. Also, they do not handle the long delays of detection.

A few recent works [10], [25], [12], [11], [13] also address security issues in DTNs. Li et al. [10] studied the blackhole attack in which malicious nodes forge routing metrics to attract packets and drop all received packets. Their approach uses a primitive called encounter ticket to prove the existence of contacts and prevent the forgery of routing metrics, but encounter ticket cannot be used to address flood attacks. Li and Cao [13] also proposed a distributed scheme to mitigate packet drop attacks, which works no matter if the attackers forge routing metrics or not. Ren et al. [11] studied wormhole attacks in DTNs. Chen and Choon [25] proposed a credit-based approach and Shevade et al. proposed a gaming-based approach [12] to provide incentives for packet forwarding. Privacy issues have also be addressed [38], [39], However, these work do not address flood attacks. Other works (e.g., Sprite [34]) deter abuse by correlating the amount of network resources that a node can use with the node's contributions to the network in terms of forwarding. This approach has been proposed for mobile ad hoc networks, but it is still not clear how the approach can be applied to DTNs, where nodes are disconnected most of the time. Another recent work [14] proposed a batch authentication protocol for DTNs, which verifies multiple packet signatures in an aggregated way to save the computation cost. This work is complementary to ours, and their protocol can also be used in our scheme to further reduce the computation cost of authentication.

Parallel to our work, Natarajan et al. [35] also proposed a scheme to detect resource misuse in DTNs. In their scheme, the gateway of a DTN monitors the activities of nodes and detects an attack if there is deviation from expected behavior. Different from their work that requires a special gateway for counting, our scheme works in a totally distributed manner and requires no special nodes.

VIII. **CONCLUSIONS**

In this paper, we employed rate limiting to mitigate flood attacks in DTNs, and proposed a scheme which exploits claim-carry-and-check to probabilistically detect the violation of rate limit in DTN environments. Our scheme uses efficient constructions to keep the computation, communications and storage cost low. Also, we analyzed the lower bound and upper bound of detection probability. Extensive trace-driven simulations showed that our scheme is effective to detect flood attacks and it achieves such effectiveness in an efficient way. Our scheme works in a distributed manner, not relying on any online central authority or infrastructure, which well fits the environment of DTNs. Besides, it can tolerate a small number of attackers to collude.

ACKNOWLEDGMENT

This work was supported by R.V.S Faculty of Engineering in Computer Science Engineering Department of Coimbatore.

REFERENCES

1. K. Fall, “A Delay-Tolerant Network Architecture for Challenged Internets,” Proc. ACM SIGCOMM, pp. 27-34, 2003.
2. P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, “Pocket Switched Networks and Human Mobility in Conference Environments,” Proc. ACM SIGCOMM, 2005.
3. M. Motani, V. Srinivasan, and P. Nuggehalli, “PeopleNet: Engineering a Wireless Virtual Social Network,” Proc. MobiCom, pp. 243-257, 2005.
4. J. Burgess, B. Gallagher, D. Jensen, and B. Levine, “Maxprop: Routing for Vehicle-Based Disruption-Tolerant Networks,” Proc. IEEE INFOCOM, 2006.
5. S.J.T.U.Grid Computing Center, “Shanghai Taxi Trace Data,” http://wirelesslab.sjtu.edu.cn/, 2012.

6. J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, Internet Denial of Service: Attack and Defense Mechanisms. Prentice Hall, 2005.
7. C. Karlof and D. Wagner, “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures,” Proc. IEEE First Int’l Workshop Sensor Network Protocols and Applications, 2003.
8. E. Daly and M. Haahr, “Social Network Analysis for Routing in Disconnected Delay-Tolerant MANETs,” Proc. MobiHoc, pp. 32-40, 2007.
9. W. Gao, Q. Li, B. Zhao, and G. Cao, “Multicasting in Delay Tolerant Networks: A Social Network Perspective,” Proc. ACM MobiHoc, 2009.
10. F. Li, A. Srinivasan, and J. Wu, “Thwarting Blackhole Attacks in Distruption-Tolerant Networks Using Encounter Tickets,” Proc. IEEE INFOCOM, 2009.
11. Y. Ren, M.C. Chuah, J. Yang, and Y. Chen, “Detecting Wormhole Attacks in Delay Tolerant Networks,” IEEE Wireless Comm. Magazine, vol. 17, no. 5, pp. 36-42, Oct. 2010.
12. U. Shevade, H. Song, L. Qiu, and Y. Zhang, “Incentive-Aware Routing in DTNS,” Proc. IEEE Int’l Conf. Network Protocols (ICNP ’08), 2008.
13. Q. Li and G. Cao, “Mitigating Routing Misbehavior in Disruption Tolerant Networks,” IEEE Trans. Information Forensics and Security, vol. 7, no. 2, pp. 664-675, Apr. 2012. 182 IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 10, NO. 3, MAY/JUNE 2013
14. H. Zhu, X. Lin, R. Lu, X.S. Shen, D. Xing, and Z. Cao, “An Opportunistic Batch Bundle Authentication Scheme for Energy Constrained DTNS,” Proc. IEEE INFOCOM, 2010.
15. B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. Snoeren, “Cloud Control with Distributed Rate Limiting,” Proc. ACM SIGCOMM, 2007.