SMURFEN: A Knowledge Sharing Intrusion Detection Network

Carol Fung[1], Quanyan Zhu[2], Raouf Boutaba[1], and Tamer Başar[2]
[1]David R. Cheriton School of Computer Science,
University of Waterloo, Ontario, Canada, {j22fung,rboutaba}@uwaterloo.ca
[2]Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign, USA. {zhu31,basar1}@illinois.edu

Technical Report CS-2011-06

Abstract—**The problem of Internet intrusions has become a world-wide security concern. To protect computer users from malicious attacks, Intrusion Detection Systems (IDSs) are designed to monitor network traffic and computer activities in order to alert users about suspicious intrusions. Collaboration among IDSs allows users to benefit from the collective knowledge and information from their collaborators and achieve more accurate intrusion detection. However, most existing collaborative intrusion detection networks rely on the exchange of intrusion data which raises the privacy concern of participants. To overcome this problem, we propose SMURFEN: a knowledge-based intrusion detection network, which provides a platform for IDS users to effectively share their customized detection knowledge in an IDS community. An automatic knowledge propagation mechanism is proposed based on a decentralized two-level optimization problem formulation, leading to a Nash equilibrium solution which is proved to be scalable, incentive compatible, fair, efficient and robust. We evaluate our rule sharing mechanism through simulations and compare our results to existing knowledge sharing methods such as random gossiping and fixed neighbors sharing schemes.**

I. Introduction

In recent years, Internet intrusions have become more sophisticated and difficult to detect. With the increasing complexity of software and systems, thousands of vulnerabilities are being discovered and exposed for exploitation every year. Attacks usually appear before security vendors release their defense technology and software vendors release their corresponding patches (e.g., zero-day attacks). Attacks from the Internet are usually accomplished with the assistance of malicious code (a.k.a. malware), including worms, viruses, Trojan horses, and Spyware. An example is the Conflicker worm which infected more than 3 million servers from year 2008 to 2009, with an estimated economic loss of $9.1 billion [1]. Recent intrusion attacks compromise a large number of nodes to form botnets [2]. Hackers not only harvest private data and identify information from compromised nodes [3], but also use those compromised nodes to launch distributed attacks such as distributed-denial-of-service (DDoS) attacks, distribution of spam messages, or organized attacks such as Fast-Flux service networks [4].

To protect computer users from malicious intrusions, Intrusion Detection Systems (IDSs) are designed to monitor network traffic and computer activities by raising intrusion alerts to network administrators or security officers. IDSs can be categorized into host-based (HIDS) or network-based (NIDS) according to their targets, and signature-based or anomaly-based according to their detection methodologies. A NIDS monitors the network traffic from/to one or a group of computers and compare the data with known intrusion patterns. A HIDS monitors the activities of one computer but has a deeper insight by tracking the system files and system logs of the computer. A signature-based IDS identifies malicious codes if the a match is found with a pattern in the attack signature database. An anomaly-based IDS, on the other hand, monitors the traffic volume or behavior of the computer and raise alerts when they are out of a predefined normal scope. Compared to HIDS, an NIDS has a broader view of the status of the network it monitors, but may miss some intrusions which are hard to detect by observing network traffic only. A signature-based IDS can accurately identify intrusions and the false positive rate is low compared to anomaly-based detection. However, it is not effective for zero-day attacks, polymorphic, and metamorphic malware [5]. An anomaly-based IDS may detect zero-day attacks by analyzing their abnormal behaviors. However, an anomaly-based detection usually generates a high false positive rate.

Traditional IDSs work independently from each other and rely on downloading new signatures or detection rules from the corresponding security vendor's signature/rule base to remain synchronized with new detection knowledge. However, the increasing number and diversity of intrusions render it not effective to rely on the detection knowledge from a single vendor, since not a single vendor can cover all the possible intrusions due to limited labor and available technology. Indeed, vendors usually choose to cover high priority intrusions which may have large influence among their clients or have high risk levels. Collaborative intrusion detection networks (CIDNs) provide a platform for IDSs to take advantage of the collective knowledge from collaborators to improve the overall detection capability and accuracy. However, most existing CIDNs, such as [6], [7], [8], [9], and [10], rely on the sharing of intrusion data with others, which raise privacy concerns from the participants. The other way, sharing detection knowledge such as malware signatures and intrusion detection rules, causes less privacy concern.

In reality, expert IDS users, including security analysts, network administrators, and security system programmers, create their own detection rules or customize existing ones to improve detection accuracy specifically for their individual environment [11]. A new detection rule created by one user may be adopted directly by another user if they have similar network/computer configurations. For example, detection rules created for an academic computing environment may be easily adopted by another similar institution; a new intrusion detection rule created to minimize vulnerability of a software can be adopted by others using the same software. An expert user who creates new rules for newly revealed vulnerabilities may share their rules with others who are subject to similar vulnerabilities. Sharing rules among a large group of users can be an effective way to improve the overall security among all users.

In this paper, we leverage the benefit of intrusion detection knowledge sharing and propose SMURFEN, a knowledge-based collaborative intrusion detection network, where intrusion detection knowledge is shared among users who have similar interests in the community. Accordingly, an automatic knowledge dissemination mechanism is proposed to allow users effectively share detection rules with other users without overwhelming their receiving capacities.

The major contributions of this paper are as follows: 1) We propose a novel intrusion detection rule sharing system, called SMURFEN[1], which is based on a peer-to-peer overlay and uses the collective social intelligence for intrusion detection. 2) We develop a rule dissemination protocol based on a decentralized two-level optimization framework, which determines the information propagation rates to each recipient. We set an optimal rule sharing policy for each node and show the existence of a Nash equilibrium in the intrusion detection network. 3) We employ Bayesian learning for each node to estimate the compatibility ratio of others based on the empirical data collected by the node. 4) We design distributed dynamic algorithms to find the Nash equilibrium and perform comprehensive simulations to demonstrate the efficiency, incentive-compatibility, fairness, robustness and scalability of the rule sharing mechanism.

The rest of the paper is organized as follows. In Section II, we give an overview of collaborative intrusion detection systems and information sharing paradigms. In Section III, we describe the SMURFEN framework. The knowledge sharing system modeling and analysis are elaborated in Section IV. We discuss the Bayesian learning of IDS compatibility in Section V and evaluate the proposed system using simulation in Section VI. Finally, we conclude the paper in Section VII.

II. Related Work

Traditional IDS collaboration utilizes the collective intrusion information and knowledge from other IDSs to improve accuracy in intrusion detection. Existing CIDNs can be categorized into information-based and expertise-based. In an information-based CIDN, IDSs collect intrusion data such as intrusion alerts or firewall logs from other nodes to perform overall intrusion detection for the whole network. Most works proposed in the last few years are information-based CIDNs, such as [6], [7], and [8]. They are especially effective in detecting epidemic worms or attacks, and most of them require homogeneous participant IDSs. In an expertise-based CIDN, suspicious data samples are sent to expert collaborators for diagnosis. Feedbacks from the collaborators are then aggregated to help the sender IDS detect intrusions. Examples of such CIDNs include those given in [12], [9], [13], and [10]. Expertise-based CIDNs may involve heterogeneous IDSs and are effective in detecting many intrusion types including malware, scannings, and vulnerability exploitations.

However, both types of CIDNs rely on the sharing of intrusion data, which raises the concern of privacy and information breaching. Therefore, it greatly discourages users from collaborating with unknown parties. In contrast, sharing detection knowledge, such as detection rules, policies, or malware signatures, does not involve the sharing of sensitive data. Hence, it can effectively eliminate the privacy concern in IDS collaboration. In fact, some open source IDSs, such as Snort, use mailing lists to allow users to contribute and share their own detection rules. However, mailing lists do not provide customized filtering and they do not scale well either, making it inefficient for frequent knowledge exchange within large communities such as social networks and peer-to-peer networks.

Information and knowledge propagation in a community can be realized through gossiping. Gossiping is a communication paradigm where information is propagated through multi-hop pair-wise communication. Gossiping has been used to exchange information in distributed collaborative intrusion detection, such as local gossiping [14], and global gossiping [15]. Sharing observations from distributed nodes is useful to detect and throttle fast spreading computer worms. It is effective for communications in ad hoc or random networks, where a structured communication topology is hard to establish. However, traditional gossiping relies on random pairs-wise communication and information flooding. Therefore, it is not suitable when the network is large and the messages are only intended to be delivered to a small set of nodes. Mailing list broadcasting can be seen as a special type of gossiping where one node communicates with every other node in the network to deliver messages.

Publish-subscribe systems also manage the information delivery from publishers to subscribers among collaborative networks. Compared to gossiping, publish-subscribe systems allow customized information delivery. They can be either topic-based, such as [16], or content-based, such as [17]. In a topic-based system, publishers and subscribers are connected together by predefined topics; content is published on well-advertised topics to which users can subscribe based on their interests. In a content-based system, users' interests are expressed through queries, and a content filtering technique is used to match the publishers' content to the subscriber. However, publish-subscribe systems do not take the quality of

[1]SMURFEN is an abbreviation for "SMURFs deFENse".

the information into consideration. Hence, they are not suitable for the intrusion detection rule sharing, for which the rule quality is critical and should be taken into account.

III. SMURFEN: A Knowledge Sharing Intrusion Detection Network

An intrusion detection rule is a detection policy which specifies the pattern of suspicious attacks. Each rule can trigger an alert once the pattern is matched. Detection rules can be vulnerability-based or exploit-based. A vulnerability is a software defect or system misconfiguration that allows attackers to gain access or interfere with system operations. Common examples of software vulnerabilities are software buffer overflows and HTTP header injection. A vulnerability-based detection rule specifies the pattern of attacks on a specific vulnerability. The patterns can be the IP address, port number, protocol flags, and context of the data payload. An exploit-based detection rule specifies the common patterns of general attacks. Comparatively the exploit-based detection causes higher false positives than vulnerability-based detection, but is effective when the vulnerabilities are unknown.

Defense against attackers is a challenging problem since a defender needs to know all possible attacks to ensure network security, whereas an attacker only needs to know a few attack techniques to succeed. It is often impossible for one person or a small group of defenders to know all attack techniques but is common to have knowledge about some attacks. As a result, the attackers have a significant advantage over the defenders. This motivates defenders to share knowledge with others to overcome their weakness. In fact, some open source intrusion detection systems, such as Snort [18] and OSSEC [19], allow users to create and edit detection rules, which provides an opportunity for users to contribute and exchange intrusion detection rules. The purpose of SMURFEN is to provide such a platform for users to share their detection rules with others effectively. In this paper, we use Snort detection rules as examples to demonstrate our rule-sharing system. We focus on an efficient rule sharing mechanism design and compare it with other possible solutions such as random gossiping and fixed neighbors sharing mechanisms.

A. The SMURFEN Framework

The SMURFEN framework is built on a Chord [20] peer-to-peer (p2p) communication overlay as illustrated in Figure 1. Each node is assigned a key and maintains a finger table which contains a list of other nodes for key search (e.g. routing) in the Chord ring. Other than that, each node also maintains a list of neighbors to communicate and exchange intrusion detection rules with. We call such a list the *acquaintance list*. In the rest of this paper, we use the terms *acquaintance* and *neighbor* interchangeably. Note that the acquaintance relationship is symmetric, i.e., if node i is in node j's acquaintance list, then node j is in node i's acquaintance list. Each node may have a long list of acquaintances and each acquaintance j has a certain probability $p_{ij} \in (0, 1]$ to be chosen to receive rules from the sender node i. A user on the receiver side

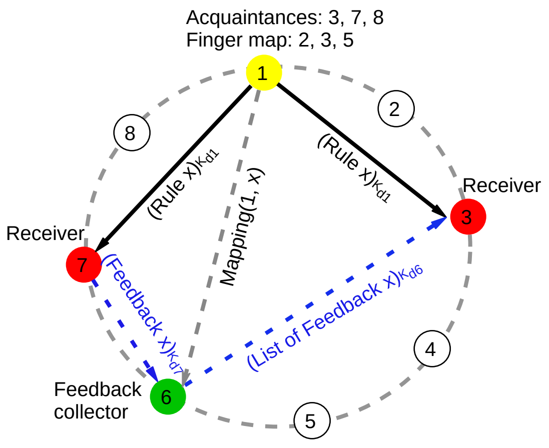
Acquaintances: 3, 7, 8
Finger map: 2, 3, 5
1
2
3
4
5
6
7
8
Receiver
Receiver
Feedback collector
$(\text{Rule } x)_{K_{d1}}$
$(\text{Rule } x)_{K_{d1}}$
Mapping(1, x)
$(\text{Feedback } x)_{K_{d7}}$
$(\text{List of Feedback } x)_{K_{d6}}$

Fig. 1. SMURFEN design of 8 nodes on a Chord ring: nodes 3 and 7 receive a rule from node 1. The feedbacks are collected by node 6.

evaluates rules sent from its neighbors and may choose to "*accept*" or "*reject*" the rule. The decision is then recorded by a Bayesian learning algorithm to update the *compatibility ratio* of the sender. The *compatibility ratio* from i to j is the probability that the rules from the sender i are accepted by the receiver j. The higher a collaborator's compatibility, the more helpful it is in collaboration. The decision is also sent to a corresponding rule feedback collector. The feedback collector is a random node in the p2p network, determined by a key mapping function of the rule ID and the sender ID. The corresponding node holding the key will host the feedback of the rule. Inexperienced users can check feedback from others before they make their own decision whether to accept the rule or not. Users can also report false positives and true positives about the rule, so that the rule creator can collect feedback and make updates accordingly. More details about the feedback collector are provided in section III-D.

B. Snort Rules

Many intrusion detection systems, such as Snort, allow users to create and edit their own detection rules in their rule base. Snort rules are certified by the Vulnerability Research Team (VRT) [21], after being tested by security experts. Snort rules are vulnerability-based and written in plain text; hence can be easily interpreted and edited by users. Snort rules obtained from third parties can be adopted directly or indirectly with some changes. Snort rules can be independent or can be grouped together into rule units. The basic rule structure includes two logical sections: the header section and the option section. The rule header contains the rule's action, protocol, source and destination IP addresses and network masks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine whether the rule action should be taken [22]. Figure 2 illustrates a simple Snort rule. When a TCP packet with the destination IP and port number matching the specified pattern and data payload containing the specified binary content is detected, a "mounted access" alert is raised.

```
alert tcp any any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"mountd access";)
```

Fig. 2. An Example of Snort Rule (adapted from [22])

C. Join or Leave SMURFEN

To prevent the man-in-the-middle attack, the communication between each pair of nodes is signed by the private key of the sender. When a new node joins the network, it creates a key pair (K_e, K_d), and registers a new ID into the p2p network. Note that the complexity of searching for a node with a specific ID in Chord is $O(\log(n))$. If a conflict is detected, a new key pair is generated and the process is repeated until the registration is successful. After that, the new node sends connection requests to random nodes in the network and acquaintance relationships are established when the requests are accepted. New intrusion detection rules are exchanged among acquaintances.

When a node leaves a network, it is not required to send a notification to other nodes. When a collaborator sends rule retrieving requests and receives no response, it automatically sets the acquaintance connection status to be inactive and select new acquaintances. Note that the frequency of acquaintance seeking requests should be limited for each node. Excessive acquaintance seeking requests from a node are dropped by routing nodes to prevent from DoS attacks.

D. Feedback Collector

When a user receives new rules from the community, she/he may evaluate the rules and determine whether or not to adopt the rule. A SMURFEN system includes feedback collectors to record the feedback on the rules from users. Less experienced users may check the feedback from others before making their decisions. As shown in Figure 3, rule author "A" propagates a new rule i to its acquaintances R_1 and R_2. Both rule receivers can retrieve and send feedback from/to the feedback collector C, which is a random node in the p2p network determined by the key mapping of the creator and the rule ID. Replicas collectors can be used to improve the availability of feedback collector service. All feedbacks are signed by their authors to prevent from malicious tampering.

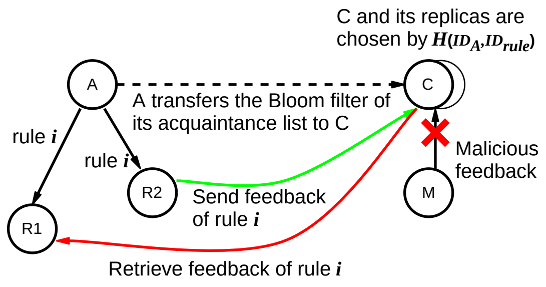
C and its replicas are chosen by $H(ID_A, ID_{rule})$
A
C
A transfers the Bloom filter of its acquaintance list to C
rule i
rule i
Malicious feedback
R2
Send feedback of rule i
M
R1
Retrieve feedback of rule i

Fig. 3. Feedback Collection in SMURFEN. The malicious node M attempts to leave fraudulent feedback but was blocked since it does not match the Bloom filter on the feedback collector.

Moreover, to avoid feedback fraudulence, each feedback collector maintains a Bloom filter [23] of the authorized nodes list. The rule author hashes all of its acquaintances into a Bloom filter and passes it to the feedback collector. Only nodes with hashed IDs matching the Bloom filter are allowed to leave feedback on the collector. The use of Bloom filter not only reduces the communication overhead to transfer long acquaintance lists, it also avoids unnecessary information leaking from the rule author.

E. An example

For a better understanding of the rule sharing framework, we illustrate the mechanism with an example (see Figure 1). Assume that user 1 (on node 1) detects a new software vulnerability and creates a new Snort rule x to protect the system before the official release from the VRT. User 1 is part of the rule sharing network. The new rule is automatically propagated to its acquaintances through a propagation process (to be described in Section IV). User 3 and user 7 receive rule x from user 1. The user 7 finds rule x to be useful to her/his network and can choose to *accept* or *reject* it. The decision is then notified to a feedback collector on node 6. If the rule is adopted and alerts are triggered by rule x, the decision whether it is a true or false alarm is also forwarded to node 6. Users can reject a formally accepted rule any time when it causes large false positives or does not detect any attack after a certain amount of time. Rule x is also propagated to node 3. The user 3 finds that the rule covers vulnerabilities but does not have enough experience to judge the quality of the rule, she/he chooses to inspect the feedback from other users about the rule from the feedback collector. The decision of acceptance or rejection can be delayed to allow enough time for observation.

IV. SMURFEN Knowledge Propagation Modeling

Knowledge propagation is an essential part of the SMURFEN system. In this section, we describe a system model for a collaborative network comprising a set of n IDSs, denoted by $\mathcal{N}$. In the network, users are allowed to contribute and share rules with others using peer-to-peer communication substrate. A user i propagates new rules to its neighbors, denoted by $\mathcal{N}_i$, with a probability $p_{ij}, j \in \mathcal{N}_i$, to achieve an optimal impact. We let $n_i = |\mathcal{N}_i|$ be the number of neighbors of node i. The communication in the collaboration network is bi-directional, i.e., if node i propagates rules to node j, then node j also propagates rules to node i. We use a matrix $\mathbf{r} = [r_{ij}]_{i,j \in \mathcal{N}}$ to represent the rule propagation rate between nodes in the network and $r_{ij} \in [0, \bar{r}_i], \forall i, j \in \mathcal{N}$, is the rule propagation rate from node i to node j. To make the design robust to DoS attacks, nodes specify maximum sending rate from their neighbors. Denoted by $\mathbf{R} = [R_{ij}]_{i,j \in \mathcal{N}}$ the *requested sending rate* from i to j. Note that R_{ij} is controlled by node j and informed to node i. SMURFEN requires nodes to control their sending rate under the requested rate, i.e., $r_{ij} \leq R_{ij}, \forall i, j \in \mathcal{N}$. To control the communication overhead, an IDS i can set the upper-bound $M_i \in \mathbb{R}_{++}$ on the total outbound communication rate, i.e, $\sum_{j\in\mathcal{N}_i} r_{ij} \leq M_i$. Denote by $\bar{r}_i$ the *rule contribution rate* from node i. The rule propagation rate from node i to other nodes can not exceed the rule contribution rate $\bar{r}_i$ of node i. Let $p_{ij} \in [0,1]$ denote the probability that node i send a rule to node j when such a new rule occurs. Then the probability can be derived from the rule sending and contribution rates, i.e., $p_{ij} = \frac{r_{ij}}{\bar{r}_i}$.

Propagated rules are not all equally useful to their recipients. To capture the metric of relationship on helpfulness, we use a matrix $\mathbf{C} = [C_{ij}]_{i,j\in\mathcal{N}}$ to denote the *compatibility ratio* between two nodes, where $C_{ij} \in [0,1], \forall i,j \in \mathcal{N}$, representing the probability or likelihood that a rule useful to node i is also useful to node j. Note that the compatibility matrix can be asymmetric, i.e., $C_{ij} \neq C_{ji}$.

Our goal is to devise a system-wide rule propagation protocol so that the rules contributed by all contributors are fairly distributed to other nodes so as to optimize their impact on the system. To achieve this goal, we model our system based on a two-level optimization problem formulation sketched in Figure 4. At the lower level, an IDS i solves the optimization problem (PPi) where it chooses its propagation rate $\vec{r}_i$ to optimize its public utility function. At the upper level, an IDS i determines the request rate to all neighbors $\vec{R}_i$ from a private optimization problem (Pi). The choice of R_{ji} at the upper level influences the decision-making at the lower public optimization level.

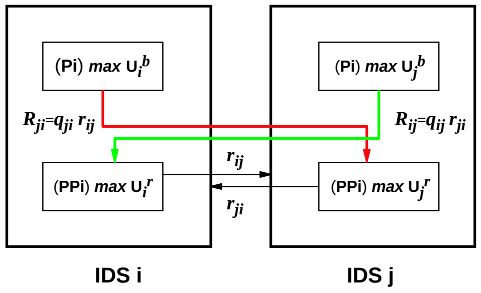

Fig. 4. An illustration of the rule propagation protocol between IDS i and IDS j. Each IDS has a two-level decision process. IDS i optimizes the propagation rate r_{ij} based on an altruistic or public optimization (PPi) and uses a private optimization problem (Pi) to determine the requested sending rate R_{ji} which will be passed to IDS j for its propagation decisions. It can be seen that the (PPj) decision of IDS j depends on the decision from (Pi) of IDS i. The interdependence of the agents leads to a Nash equilibrium.

A. Lower Level – Public Utility Optimization

In this subsection, we formulate an optimization framework for each node to decide the propagation rate to all its neighbors to maximize its utility. The utility of each node U_i has two components: a public utility function U_i^r and a private utility function U_i^b. The utility U_i^r measures the aggregated satisfaction level experienced by node i's neighbors weighted by their compatibility ratios. It allows a node to propagate its rules more toward those with whom it is more compatible. On the other hand, U_i^b measures the satisfaction level of a node with respect to the amount of help it receives from its neighbors.

An IDS i can control two sets of variables, $\vec{r}_i = [r_{ij}]_{j\in\mathcal{N}_i}$ and $\vec{R}_i = [R_{ji}]_{j\in\mathcal{N}_i}$. We call $q_{ji} = \frac{R_{ji}}{r_{ij}}$ the greed factor, which reflects the greediness of the request from node j. $q_{ji} > 1$ indicates that node j requests a higher rule propagation rate from node i than the rate it propagates to node i. The introduction of greed factor serves two major purposes: 1) it sets an expectation of return ratio so that a node i can determine its rule propagation rate r_{ij} and R_{ij}/r_{ji} can reach q_{ij} to achieve maximum satisfaction from node j; 2) it serves as an upper bound for communications between nodes i and j, i.e., $r_{ij} \leq q_{ij} r_{ji}$, or equivalently, $r_{ij} \leq R_{ij}$. It circumvents potential denial-of-service attacks from a malicious node who sends an excessive volume of traffic to node j.

The public optimization problem (PPi) seen by each node $i, i \in \mathcal{N}$, is given by

$$\text{(PP}i\text{)} \max_{\vec{r}_i \in \mathbb{R}^{n_i}} U_i^r(\vec{r}_i) \quad := \quad \sum_{j\in\mathcal{N}_i} C_{ji} S_{ij}(r_{ij}) \tag{1}$$

$$\sum_{j\in\mathcal{N}_i} r_{ij} \quad \leq \quad M_i, \tag{2}$$

$$r_{ij} \quad \leq \quad R_{ij}, \tag{3}$$

$$0 \leq r_{ij} \quad \leq \quad \bar{r}_i, \tag{4}$$

where $S_{ij} : \mathbb{R} \to \mathbb{R}$ is the satisfaction level of node j in response to the propagation rate r_{ij} of node i. We let S_{ij} take the following form

$$S_{ij}(r_{ij}) := C_{ij} \log\left(1 + \frac{r_{ij}}{R_{ij}}\right). \tag{5}$$

The concavity and monotonicity of the satisfaction level indicate that a recipient becomes increasingly pleased when more rules are received but the marginal satisfaction decreases as number of received rules increases. The parameter C_{ij} in (5) suggests that a node j is more content when the compatibility or usefulness of rules sent from node i is high.

The objective function $U_i^r : \mathbb{R}^{n_i} \to \mathbb{R}$ in (1) aggregates the satisfaction level S_{ij} of node j by the compatibility factor C_{ji}. The utility U_i^r can be viewed as a public altruistic utility in that a node i seeks to satisfy its collaborators by choosing propagation rates $\vec{r}_i$. The problem (PPi) is constrained by (2) in that the total sending rate of a node i is upper bounded by its communication capacity. The additional constraint (4) ensures that the propagation rate does not exceed its rule contribution rate $\bar{r}_i$. Note that the constraint (3) is imposed by its recipient while constraint (4) is set by node i itself.

Define the set $\mathcal{F}_i^1 := \{\vec{r}_i \in \mathbb{R}^{n_i} : \sum_{j\in\mathcal{N}_i} r_{ij} \leq M_i, M_i \in \mathbb{R}_{++}\}$ and $\mathcal{F}_i^2 := \cap_{j\in\mathcal{N}_i}\mathcal{F}_{ij}^2$, where $\mathcal{F}_{ij}^2 := \{r_{ij} \in \mathbb{R}_+ : r_{ij} \leq \min(R_{ij}, \bar{r}_i)\}$. The optimization problem is feasible if and only if $\mathcal{F}_i := \mathcal{F}_i^1 \cap \mathcal{F}_i^2$ is not empty. The feasible set is a convex polytope and it can be represented by the convex hull of its finite set of K_i extreme points $\mathcal{K}_i = \{k_1, k_2, \cdots, k_{K_i}\}$, where $K_i = |\mathcal{K}_i|$. Since the utility function (1) is strictly convex in $\vec{r}_i$ and the feasible set is convex, the optimization problem (PPi) is in a form of convex programming and admits a unique solution.

It can be seen that when M_i is sufficiently large and (2) is an inactive constraint, the solution to (PPi) becomes trivial and $r_{ij} = \min(R_{ij}, \bar{r}_i)$ for all $j \in \mathcal{N}_i$. The situation becomes more interesting when (2) is an active constraint. Assuming that q_{ij} and hence R_{ij} have been appropriately set by node j, we form the Lagrangian functional $\mathcal{L}^i : \mathbb{R}^{n_i} \times \mathbb{R} \times \mathbb{R}^{n_i} \to \mathbb{R}$

$$\mathcal{L}^i(\vec{r}_i, \mu_i, \delta_{ij}) := \sum_{j\in\mathcal{N}_i} C_{ji}C_{ij} \log\left(1 + \frac{r_{ij}}{R_{ij}}\right)$$
$$-\mu_i \left(\sum_{j\in\mathcal{N}_i} r_{ij} - M_i\right) - \sum_{j\in\mathcal{N}_i} \delta_{ij}(r_{ij} - \bar{r}_i), \quad (6)$$

where $\mu_i, \delta_{ij} \in \mathbb{R}_+$ satisfy the complementarity conditions $\mu_i \left(\sum_{j\in\mathcal{N}_i} r_{ij} - M_i\right) = 0$, and $\delta_{ij}(r_{ij} - \bar{r}_{ij}) = 0, \forall j \in \mathcal{N}_i$, where $\bar{r}_{ij} := \min(R_{ij}, \bar{r}_{ij})$. We minimize the Lagrangian with respect to $\vec{r}_i \in \mathbb{R}^{n_i}_+$ and obtain the first-order Kuhn-Tucker condition: $\frac{C_{ij}C_{ji}}{r_{ij}+R_{ij}} = \mu_i + \delta_{ij}, \quad \forall j \in \mathcal{N}_i$. When (2) is active but (3) and (4) are inactive, we can find an explicit solution supplied with the equality condition

$$\sum_{j\in\mathcal{N}_i} r_{ij} = M_i \quad (7)$$

and consequently, we obtain the optimal solution

$$r^\star_{ij} = r^*_{ij} := \frac{C_{ij}C_{ji}}{\sum_{u\in\mathcal{N}_i} C_{iu}C_{ui}} \left(M_i + \sum_{v\in\mathcal{N}_i} R_{iv}\right) - R_{ij}. \quad (8)$$

When either one of the constraints (3) and (4) is active, the optimal solution is attained at one of the extreme points of the polytope. Since the $\log$ function has the fairness property, the optimal solution $r^\star_{ij}$ has non-zero entries when the resource budget $M_i > 0$. In addition, due to the monotonicity of the objective function, the optimal solution $r^\star_{ij}$ is attained when all resource budgets are allocated, i.e., constraint (2) is active. Hence, the optimal solution $r^\star_{ij}$ to (PPi) is always on the face of the polytope where (7) holds .

Remark 1: We can interpret (8) as follows. The solution r^*_{ij} is composed of two components. The first part is a proportional division of the resource capacity M_i among $|\mathcal{N}_i|$ neighbors by their compatibilities. The second part is a linear correction on the proportional division by balancing the requested sending rate R_{ij}. It is also important to notice that by differentiating r^*_{ij} with respect to R_{ij}, we obtain $\frac{\partial r^*_{ij}}{\partial R_{ij}} = \frac{C_{ij}C_{ji}}{\sum_{u\in\mathcal{N}_i} C_{iu}C_{ui}} - 1 < 0$, suggesting that at the optimal solution, the propagation rate decreases as the recipient sets a higher requested sending rate. If a node wishes to receive higher propagation rate from its neighbors, it has no incentive to overstate its level of request. Rather, a node j has the incentive to understate its request level to increase r^*_{ij}. However, the optimal solution is upper bounded by $\min(barr_i, R_{ij})$. Hence, by understating its request R_{ij}, the optimal propagation rate is achieved at its boundary point $\min(\bar{r}_i, R_{ij})$.

B. Upper Level – Private Utility Optimization

An IDS i has another degree of freedom to choose its level of requested sending rate R_{ji} of its neighbors. R_{ji} states the maximum rule propagation rate from node j to i that node i can accept. In contrast to the public utility optimization, the optimization at this level is inherently non-altruistic or private. The objective of an IDS i is to choose $\vec{R}_i$ so that its private utility $U^b_i : \mathbb{R}^{n_i}_+ \to \mathbb{R}$ is maximized, i.e.,

$$(\text{P}i) \quad \max_{\vec{R}_i \in \mathbb{R}^{n_i}_+} U^b_i(\vec{R}_i), \quad (9)$$

subject to the following constraint from the total receiving capacity $\bar{R}_i$, i.e., $\sum_{j\in\mathcal{N}_i} R_{ji} \le \bar{R}_i$. Let U^b_i take the form of $U^b_i := \sum_{j\in\mathcal{N}_i} C_{ji} \log(1 + r^\star_{ji})$, where $r^\star_{ji}$ is the optimal solution attained at (PPi). The $\log$ function indicates that an IDS intends to maximize its own level of satisfaction by choosing an appropriate level of request. The request capacity is imposed to prevent excessive incoming traffic as a result of high level of requests. We assume that the capacity is sufficiently large so that the constraint is inactive. Therefore, the decision variable R_{ji} is uncoupled and the problem (Pi) can be equivalently separated into $|\mathcal{N}_i|$ optimization problem with respect to each j, i.e., for every $j \in \mathcal{N}_i$,

$$(\text{P}ij) \quad \max_{R_{ji}\in\mathbb{R}_+} \log(1 + r^\star_{ji}). \quad (10)$$

The following proposition characterizes the optimal choice of R_{ji} or q_{ji} of node i.

Proposition 1: Assume that $\bar{r}_i$ is sufficiently large so that the constraint (4) is inactive. The optimization problem (Pi) admits an optimal solution given by

$$R^*_{ji} = q^*_{ji} r_{ij} = \frac{1}{2} \frac{C_{ij}C_{ji}}{\sum_{u\in\mathcal{N}_j} C_{ju}C_{uj}} \left(M_j + \sum_{v\in\mathcal{N}_j} R_{jv}\right). \quad (11)$$

Proof: The proof of Proposition 1 is in Appendix A. ■

Combining the solutions to optimization problems (PPi) and (Pi) with the above result, we arrive at

$$r^\star_{ij} = R^*_{ij} = \frac{1}{2} \frac{C_{ij}C_{ji}}{\sum_{u\in\mathcal{N}_i} C_{iu}C_{ui}} \left(M_i + \sum_{v\in\mathcal{N}_i} R_{iv}\right). \quad (12)$$

Equation (12) suggests that an optimal response of node i to node j is to propagate rules at the same rate as the requested rate, which is proportional to the propagation rate sent by node j by the optimal greed factor q^*_{ij} since $R^*_{ij} = q^*_{ij} r_{ji}$.

The properties of the solutions to (Pi) and (PPi) are illustrated in Figure 5 for an IDS i and its two neighboring peers. In this illustrative example, we look at the optimal propagation rule for node i to communicate with node 1 and 2. Node i solves (PPi) with constraints (1) $r_{i1}+r_{i2} \le M_i$, (2) $r_{i1} \le R_{i1}$, and (3) $r_{i2} \le R_{i2}$. The shaded region is the feasible set of the optimization problem. The optimal allocation can be points on the face of $r_{i1}+r_{i2} = M_i$ of the feasible set. Given the request rates R_{i1} and R_{i2}, suppose the optimal allocation is found at the red point. At the higher level, nodes 1 and 2 need to

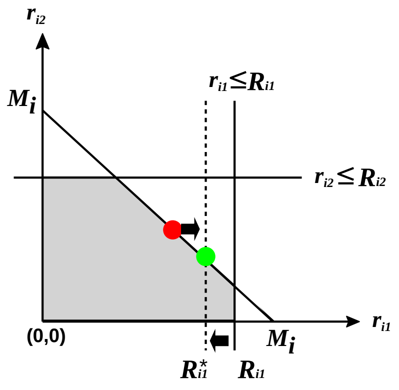

Fig. 5. An illustrative example of a 3-person system involving the set of nodes $\{i, 1, 2\}$. Node i solves (PPi) while nodes 1 and 2 solve (P1i) and (P2i), respectively.

solve the optimization problems (P1i) and (P2i), respectively. They have incentives to understate their requests. For example, node 1 can request a lower rate until it hits R^*_{i1} and the optimal allocation will increase until it reaches R^*_{i1}. This fact leads to the green point which is the optimal solution to (PPi) found on the vertex of its feasible set given that $r_{i1} \leq R^*_{i1}$. Node 2 makes the same decision and results in R^*_{i2}.

C. Nash Equilibrium

In a collaboration network, each node responds to other nodes by choosing optimal propagation rates and request rates. The two-level optimization problem leads to two game structures of interest. Let $\mathbf{G1} := \langle \mathcal{N}, \{\vec{r}_i\}_{i\in\mathcal{N}}, \{U^r_i\}_{i\in\mathcal{N}} \rangle$ be the game that corresponds to optimization problem (PPi) in which each node chooses its propagation rates given the requested sending rates from its neighbors. Hence, the utilities of the users in Equation (5) reduce to mere functions of r_{ij}. Denote by $\mathbf{G2} := \langle \mathcal{N}, \{\vec{r}_i, \vec{R}_i\}_{i\in\mathcal{N}}, \{U^r_i, U^b_i\}_{i\in\mathcal{N}} \rangle$ the game that corresponds to the two-level optimization problem (PPi) together with (Pi). In $\mathbf{G2}$, each node i chooses its propagation rates as well as its request rates. We study the existence and uniqueness properties of Nash equilibrium (NE) of these two games as follows.

Proposition 2: There exists a NE for $\mathbf{G1}$ and $\mathbf{G2}$.

The proof of Proposition 2 is in Appendix B.

Theorem 1: There exists a NE such that $r_{ij} = R_{ij}, \forall i, j \in \mathcal{N}$ in $\mathbf{G2}$. We call such NE a prime NE.

The proof of Theorem 1 is provided in Appendix C. In the following, we propose two results on the uniqueness of NE in $\mathbf{G1}$ and $\mathbf{G2}$. Their proofs are in Appendices D and E, respectively.

Proposition 3: Assume that only (2) is an active constraint in optimization problem (Pi) of each node i in $\mathbf{G1}$. Let $\lambda_{ij} = \frac{C_{ij}C_{ji}}{\sum_{u\in\mathcal{N}_i} C_{iu}C_{ui}}$. There exists a unique NE for $\mathbf{G1}$ if $q_{ij}q_{ji} \neq \frac{1}{(1-\lambda_{ij})(1-\lambda_{ji})}$ for each pair of neighbor nodes i, j.

Proposition 4: Assume that $\bar{r}_i$ is sufficiently large and the response of each node follows (12). There exists a unique NE for $\mathbf{G2}$ if $n_i\lambda_{ij} < 2$ for every pair of neighbor nodes i and j.

D. Dynamic Algorithm to Find the Prime NE

Algorithm 1 Distributed Dynamic Algorithm to Find the Prime NE at node i

```
 1: Initialization :
 2: $\vec{R}^{in} \Leftarrow \{\epsilon, \epsilon, ..., \epsilon\}$ // Small request rates for new neighbors.
 3: $\vec{R}^{out} \Leftarrow$ SendReceive($\vec{R}^{in}$) // Exchange requested sending rates
    with all neighbors.
 4: set new timer event($t_u$, "SpUpdate") // Update sending rates
    and request rates periodically.
 5: Periodic update:
 6: at timer event ev of type "SpUpdate" do
 7: // Update the sending rate to all neighbors and then update the
    requested sending rates from all neighbors.
 8: for $k = 0$ to $B$ do
 9:     $\vec{r}^{out} \Leftarrow$ OptimizeSending($\mathbf{C}, \vec{R}^{out}, \mathbf{M}, \bar{r}$) // (PP$i$) optimiza-
        tion.
10:     $\vec{r}^{in} \Leftarrow$ SendReceive($\vec{r}^{out}$) // Exchange sending rate with all
        neighbors.
11:     $\vec{R}^{in} \Leftarrow$ OptimizeRequest($\mathbf{C}, \vec{r}^{in}, \bar{R}$) // (P$i$) optimization.
12:     $\vec{R}^{out} \Leftarrow$ SendReceive($\vec{R}^{in}$) // Exchange requested sending
        rate with all neighbors.
13: end for
14: set new timer event($t_u$, "SpUpdate")
15: end timer event
```

In this subsection, we describe a distributed algorithm (Algorithm 1) for each node to decide its rule propagation rates. The subscript i is removed for the convenience of presentation. The goal of the algorithm is to lead the system to converge to a prime NE that we defined previously. In the beginning, nodes set a small requested sending rate for all new neighbors (line 2). An update process is triggered periodically where function **OptimizeSending** is used for the nodes to find their optimal sending rates $\vec{r}^{out}$ based on the compatibility matrix $\mathbf{C}$ and requested sending rate $\vec{R}^{out}$, which is informed by the acquaintances in process **SendReceive** (line 3). $\mathbf{M}$ and $\bar{r}$ are the sending capacity and rule contribution rate of i respectively. Function **OptimizeRequest** is used for the nodes to find optimal $\vec{R}^{in}$ ($\mathbf{G2}$) which gives the maximal private utility, given the $\mathbf{C}$, the incoming sending rate $\vec{r}^{in}$, and the receiving capacity $\bar{R}$. The update process is repeated B rounds to yield a converged result.

The purpose of Algorithm 2 is to find the optimal numerical solution for (PPi) under general conditions. This algorithm is based on the properties that the marginal weighted satisfactions from all neighbors are continuous and monotonically decreasing, i.e., $(C_{ji}S''_{ij}) < 0, \forall i, j$. $\mathcal{S}$ contains the sorted marginal weighted satisfactions of all neighbors at their boundaries $\{0, \min(\bar{r}_i, R_{ij})\}$. The idea is to find the "cutoff" marginal satisfaction, where neighbors with both marginals higher than the "cutoff" take their upper-bounds, neighbors with both marginals lower than the 'cutoff" take their lower-bounds, and others takes inner solutions with their marginals equal to the "cutoff". We start "cutoff" low, increase it step by step, and move nodes to $\mathcal{S}_H$ and $\mathcal{S}_M$ accordingly until the sending resource exceeds its capacity. The computation complexity of Algorithm 2 is $O(|\mathcal{N}|)$.

Algorithm 2 Function OptimizeSending($\mathbf{C}$, $\vec{R}^{out}$, M, $\bar{r}$)

```
1: // Sort the marginal satisfaction of all neighbors at their lower-bound and upper-bound. $\mathcal{N}$ is the acquaintance list.
2: $\mathcal{S} \leftarrow \emptyset$ // $\mathcal{S}$ is a descending ordered set, initially empty.
3: for $i = 0$ to $|\mathcal{N}|$ do
4:     $\mathcal{S} \Leftarrow \mathcal{S} \cup \{\frac{\mathbf{C}_i}{\vec{R}_i^{out}}, \text{“}L\text{”}, \mathcal{N}_i\}$ // Add the marginal weighted satisfaction at the lower bound 0 of node $i$.
5:     $\mathcal{S} \Leftarrow \mathcal{S} \cup \{\frac{\mathbf{C}_i}{\vec{R}_i^{out}+\min(\bar{r},\vec{R}_i^{out})}, \text{“}H\text{”}, \mathcal{N}_i\}$ // Add the marginal weighted satisfaction at the upper bound of node $i$.
6: end for
7: // Three sets, containing neighbors taking upper-bound, medium value, and lower-bound at the optimal solution.
8: $\mathcal{S}_H = \emptyset$, $\mathcal{S}_M = \emptyset$, $\mathcal{S}_L = \mathcal{N}$
9: for each $V \in \mathcal{S}$ do
10:    NextCutOff← FirstElementOf($V$) // Marginal satisfaction.
11:    if Resource($\mathcal{S}_L, \mathcal{S}_M, \mathcal{S}_H$,NextCutOff) $< M$ then
12:        if SecondElementOf(V) = “$L$” then
13:            move the associated neighbor of $V$ from $\mathcal{S}_L$ to $\mathcal{S}_M$
14:        else
15:            move the associated neighbor of $V$ from $\mathcal{S}_M$ to $\mathcal{S}_H$
16:        end if
17:    else
18:        go to FinalStep: // The final cutoff marginal is lower than the NextCutOff.
19:    end if
20: end for
21: FinalStep: // Assign sending rates to all neighbors.
22: for $j = 0$ to $|\mathcal{N}|$ do
23:    if $\mathcal{N}_j \in \mathcal{S}_H$ then
24:        $\vec{r}_j = \min(\bar{r}, \vec{R}_j^{out})$ // Nodes take upper-bounds.
25:    else if $\mathcal{N}_j \in \mathcal{S}_L$ then
26:        $\vec{r}_j = 0$ // Nodes take lower-bounds.
27:    else
28:        // All the other neighbors have inner solutions. We use Kuhn-Tucker condition to find their solutions.
29:        $\vec{r}_j = \frac{\mathbf{C}_j(M-\sum_{k\in\mathcal{S}_H}\min(\bar{r},\vec{R}_k^{out})+\sum_{k\in\mathcal{S}_M}\vec{R}_k^{out})}{\sum_{k\in\mathcal{S}_M}\mathbf{C}_k} - \vec{R}_j^{out}$
30:    end if
31: end for
32: return $\vec{r}$
```

Algorithm 3 Function OptimizeRequest($\mathbf{C}$, $\vec{r}^{in}$, $\vec{R}^{in}$, $\bar{R}$)

```
1: $\mathcal{S} \leftarrow \emptyset$ // Sorted set descending, initially empty.
2: for $i = 0$ to $|\mathcal{N}|$ do
3:     if $\vec{r}_i^{in} = 0$ then
4:         $\vec{R}_i^{in} \Leftarrow \vec{R}_i^{in}/2$ // Request to $i$ is too high, cut in half.
5:     else if $\vec{r}_i^{in} < \vec{R}_i^{in}$ then
6:         $\vec{R}_i^{in} \Leftarrow \vec{r}_i^{in}$ // Tune down request to approach Prime NE.
7:     else
8:         $\mathcal{S} \Leftarrow \mathcal{S} \cup \{C_i, \mathcal{N}_i\}$ // $\mathcal{S}$ is sorted by $C_i$, the compatibility ratio of neighbor $i$.
9:     end if
10: end for
11: // Increase the requested sending rate of the half neighbors with higher compatibility by a small amount.
12: for $j \in$ TopHalf($\mathcal{S}$) do
13:     $\vec{R}_j^{in} \Leftarrow \vec{R}_j^{in} + \Delta$ // Increase the request rate slightly.
14: end for
15: $U = \sum_{k\in|\mathcal{N}|} \vec{R}_k^{in}$ // Total request rate.
16: if $U - \bar{R} > 0$ then
17:     $\vec{R}^{in} \Leftarrow \frac{\bar{R}}{U}\vec{R}^{in}$ // Normalize into constraint $\bar{R}$.
18: end if
19: return $\vec{R}^{in}$
```

Algorithm 3 is used to adjust the requested sending rate of all neighbors according to their last status. We use an fast decrease and linear increase strategy for request adjustment. If the requested sending rate from last cycle is not fully claimed, then the next request is adjusted to be the claimed amount; otherwise, increase the request slightly. The computation complexity of Algorithm 3 is $O(|\mathcal{N}|)$.

V. Bayesian Learning Model for Compatibility

In Section IV, we assumed that the compatibilities of all neighbors are known. In practice, they can be learned from past experience. In this section, we introduce a Bayesian learning approach for nodes to learn the compatibilities of neighbors.

Past decisions to *accept* or *reject* rules from a neighbor can be seen as a Bernoulli trial with parameter p as the compatibility ratio from the neighbor. In the SMURFEN system, a node j uses a beta distribution to estimate the compatibility C_{ij} from its neighbor i. In the beginning, node j sets an initial belief on i. The posterior probability is updated at each step using the empirical data on the outcome of the acceptance/rejection decision.

A. Baysian Learning Model for Compatibility Ratio

When node j receives a new detection rule from peer i, it can choose either to *accept* the rule ($o = 1$), or *reject* the rule ($o = 0$). Let $X \in \Omega := \{0, 1\}$ be a random variable which denotes the decision outcome: *rejected* or *accepted*. Note that on the case that a rule is accepted in the beginning and then rejected due to high false positive rate, the weight of the reject decision is doubled to reverse the impact of the previous acceptance decision. Since the definition of compatibility ratio of an acquaintance is the probability a rule is accepted from the acquaintance, we have $C_{ij} = \mathbb{P}[X = 1]$, where $\mathbb{P} : 2^\Omega \to [0, 1]$ is a probability measure. We estimate C_{ij} based on the past observations $O^n := \{o^k\}_{k=1}^n \in \{0, 1\}^n$. The distribution of C_{ij} can be written as a beta distribution in the form of

$$C_{ij}^{(n)} \overset{pdf}{\sim} \frac{\Gamma(\alpha^{(n)} + \beta^{(n)})}{\Gamma(\alpha^{(n)})\Gamma(\beta^{(n)})} x^{\alpha^{(n)}-1}(1-x)^{\beta^{(n)}-1}, \qquad (13)$$

$$\alpha^{(n)} = \sum_{k=1}^{n} \lambda^{t_k} \times o^k + C_0\lambda^{t_0} \times \alpha^0, \qquad (14)$$

$$\beta^{(n)} = \sum_{k=1}^{n} \lambda^{t_k} \times w_k(1 - o^k) + C_0\lambda^{t_0} \times \beta^0, \qquad (15)$$

where α^0, β^0 are the initial beliefs of $C_{ij}, 1 - C_{ij}$, respectively; $\alpha^{(n)}, \beta^{(n)}$ represent the Beta parameters after n decision outcomes; $o^k \in \{0, 1\}$ is the k^{th} experience; t_k is the age of the k^{th} experience; $C_0 \in (0, +\infty)$ is the weight of the initial belief. $w_k \in \{1, 2\}$ is the weight of the reject decision, $w_k = 1$ for the rejection of new rule and $w_k = 2$ for the rejection of previously accepted rule. We put more weights on recent experience by introducing a forgetting factor λ, which is used to discount older experiences.

B. The Estimation of Compatibility Ratio

Denote the cumulative density function (CDF) of C_{ij} by

$$C_{ij}^{(n)} \overset{CDF}{\sim} F(x;\alpha^{(n)},\beta^{(n)}) = I_x(\alpha^{(n)},\beta^{(n)}), \tag{16}$$

where $I_x(\alpha^{(n)},\beta^{(n)})$ is the *regularized incomplete beta function* [24]. Let $\widetilde{C}_{ij}^{(n)}$ denote the estimated compatibility after n observations. We assign the 90% credential lower-bound of the CDF as $\widetilde{C}_{ij}^{(n)}$, i.e., $\widetilde{C}_{ij}^{(n)} = I_{0.1}^{-1}(\alpha^{(n)},\beta^{(n)})$. We name the above estimate a *credible-bound* compatibility. The estimate $\widetilde{C}_{ij}^{(n)}$ has several properties as follows:

(P1) for each node i, increasing the rule sharing rate increases its compatibility ratio with others.

(P2) when $\alpha^{(n)}$ and $\beta^{(n)}$ are sufficiently large (> 10), the beta density function can be approximated by a Gaussian distribution. Therefore, we have $\widetilde{C}_{ij} \approx \mu_x - 2\sigma_x = \frac{\alpha^{(n)}}{\alpha^{(n)}+\beta^{(n)}} - 2\sqrt{\frac{\alpha^{(n)}\beta^{(n)}}{(\alpha^{(n)}+\beta^{(n)})^2(\alpha^{(n)}+\beta^{(n)}+1)}}$, where μ_x and σ_x are the mean and the variance of the random variable X, respectively.

VI. Evaluation

In this section, we use a simulation network to demonstrate the appealing properties of the SMURFEN system. All our experiments are based on the average of a large number of experiment replications with different random seeds. Confidence intervals are small enough to be neglected.

A. Simulation Setup

We simulate a network of n nodes. Each node $i \in \{1, 2, \cdots, n\}$ is labeled with an expertise level $e(i) \in [0, 1]$, which is the probability that a rule propagated by node i is effective for intrusion detection. Each node i contributes detection rules to the network following a Poisson distribution with an average arrival rate $\bar{r}_i$. The rule propagation follows the two-level game design described in Section IV. We evaluate the scalability, efficiency, incentive compatibility, fairness, and robustness of the rule propagation system. The parameters we used in our experiments are shown in Table I.

In SMURFEN, C_{ij} is learned by j through past experiences using the credible-bound compatibility method described in Section V-B. In the following experiments, we compare the credible-bound method with other commonly used learning methods such as Simple Average (SA) and Moving Average (MA) through a stochastic discrete event simulation. The simple average and moving average learning schemes are summarized as follows.

1) Simple Average: Node j takes the average of the past experiences, i.e., $C_{ij}^{S} = \frac{\sum_{k=0}^{n} o^k}{n}$.

2) Moving Average: Node j takes the moving average of the past experiences. Older experiences are discounted exponentially over time, i.e., $C_{ij}^{M} = \frac{\sum_{k=0}^{n} \lambda^{t_k} o^k}{\sum_{k=0}^{n} \lambda^{t_k}}$.

B. Compatibility Learning

We set up a simple network containing node 0 and node 1. Node 0 with expertise level 0.8 sends rules to node 1 following a Poisson process with $r_{01} = 10$ rules/day. At the beginning of day 40, node 0 is compromised and starts spamming node 1. Node 1 evaluates and compares C_{01} using the three different methods. The forgetting factor used is $\lambda = 0.95$.

Figure 6 shows that C_{01} converges after a few days and the credible-bound method yields slightly lower value compared to the other two methods. From the 40^{th} day, all methods observe a fast dropping of C_{01}. However, the learning speeds of the moving average and the credible-bound methods are faster than the simple average method. This is because the forgetting factor puts higher weights on new experiences. We then change r_{01} from 1 to 19 and observe the compatibility C_{01} at the 40^{th} day using the three methods. From Figure 7, we see that C_{01} increases and approaches to 0.8 asymptotically under the credible-bound method, while C_{01} from the two other methods mostly stay at 0.8. Therefore, nodes contribute to the collaboration network more have higher compatibility ratios when the credible-bound method is used.

C. Convergence of Distributed Dynamic Algorithm

In this experiment, we evaluate the convergence speed of the dynamic algorithm (Algorithm 1) for the participants to achieve the equilibrium. We configure a network of 4 nodes sharing the same set of vulnerabilities; the expertise levels of nodes are $0.9, 0.8, 0.7$, and 0.5, respectively. Parameter settings are: $M_i = 10$ rule/day, $\bar{R}_i = 100$ rules/day, and $\bar{r}_i = 10$ rules/day, for all i. All nodes start with small sending rates and small request rates to all neighbors, and adjust them following Algorithm 1. The number of updating and exchanging is controlled by parameter B. To make an appropriate choice of B, we try different values of B and observe the sending rate from node 0 to other nodes after B rounds of optimal adjustment and information exchange. The result is shown in Figure 8. We can see that the sending rates have a fast convergence speed. The similar situation occurs under some other parameter settings. Therefore, we fix $B = 10$ in the rest of the experiments.

D. Scalability and Information Quality

In this experiment we compare the scalability and information quality using the traditional mailing list and SMURFEN propagation system. We set up a network with size starting from 10 nodes and we increase it by 30 nodes each round till

TABLE I
SIMULATION PARAMETERS

Parameter	Value	Description
M_i	10, 100	The propagation sending capacity of node i
$\bar{R}_i$	10, 100	The receiving capacity of node i
$\bar{r}_i$	1, 10	The rule contribution rate of node i
λ	0.95	Forgetting factor for Equation (14)
B	10	Computation rounds for Algorithm 1

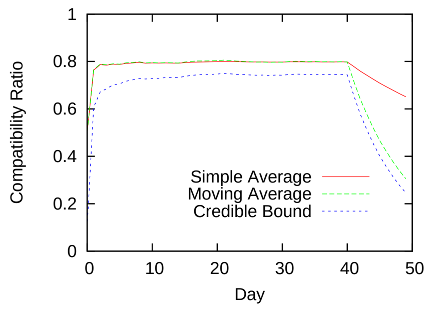

Fig. 6. Compatibility under Different Learning Methods

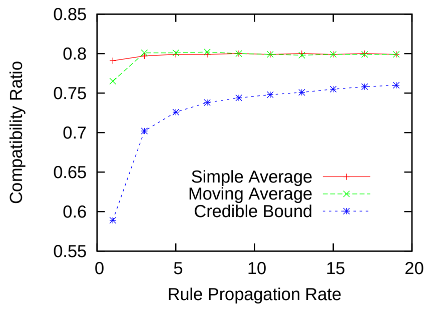

Fig. 7. The Credible-Bound Compatibility vs. Sample Rate

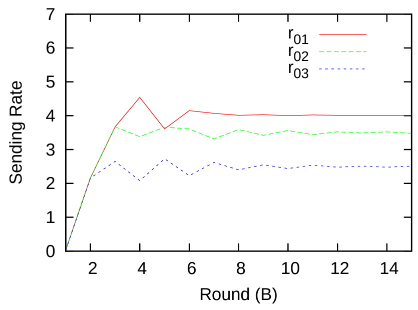

Fig. 8. The Convergence of Dynamic Algorithm to Find the Prime NE

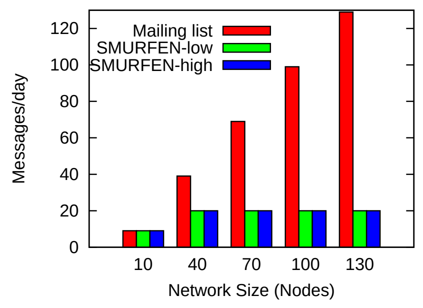

Fig. 9. The Comparison of Scalability between Mailing list and SMURFEN

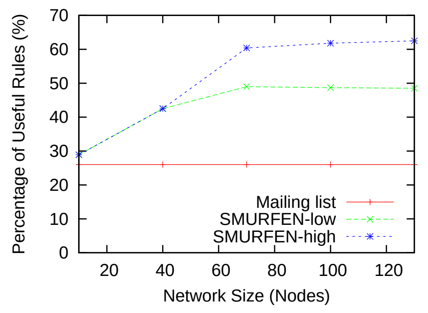

Fig. 10. The Comparison of Information Quality between Mailing list and SMURFEN

130. Among all the nodes, 20% are expert nodes with expertise level 0.9, 80% are novice nodes with expertise level 0.1. All nodes have the same setting, i.e., $M = \bar{R} = 20$ rules/day and $\bar{r} = 1$ rule/day.

Figure 9 shows the rule receiving rate from both methods. We can see that when using the mailing list propagation, the number of rules a node receives increases linearly with the network size. When the network size is large, the receiving rate may exceed the tolerance of users and may be considered as spam. SMURFEN system controls the received rule rate within the predefined capacity, and does not increase with the network size. Therefore, SMURFEN system is scalable regarding to the network size.

The information quality for both methods is plotted in Figure 10.We define the information quality to be the percentage of useful rules that nodes receive. We see that when using the SMURFEN system, the information qualities received by both the low-expertise and the high-expertise nodes are significantly improved compared to the mailing list method. The high-expertise nodes receive higher quality rules than low-expertise nodes, which reflects the incentive-compatibility of the system.

E. *Incentive Compatibility and Fairness*

Incentive compatibility is a required feature for a collaboration network since it determines the long-term sustainability of the system. In an incentive-compatible system, a well-behaving node benefits more than an ill-behaving one. In this experiment, we change the expertise level and rule contribution rate of a participating node, and observe the output of its return benefit, which is the expected number of useful rules a node receives per day.

In this experiment, we configure a network with 30 nodes with random expertise levels in $[0, 1]$, and we set $M = \bar{R} = 100$ rules/day and $\bar{r} = 10$ rules/day for all nodes. We change the expertise level of node 0 from 0.1 to 1.0 and observe its return benefit. We compare our results with two other information propagation methods, namely uniform gossiping and best neighbor mechanism. In the uniform gossiping mechanism, rules are propagated to randomly selected nodes uniformly from the neighborhood. The receiver drops rules from less compatible neighbors if the total receiving rate is over limit. In the best neighbor mechanism, rules are always propagated to a few fixed (best) neighbors. The sending capacity and receiving capacity also apply to the uniform gossiping and best neighbor propagation. Therefore, we also configure their sending and receiving capacities to be 100 rules/day.

Figure 11 shows that uniform gossiping provides no incentive to nodes with higher expertise levels. On the other hand, the best neighbor propagation scheme provides incentive but no fairness. Nodes of the same expertise levels may have very different return benefit. This is because under the best neighbor mechanism, nodes form collaboration groups. Nodes of the same expertise level may join different groups. Since the return benefit largely depends on which group a node belongs to, nodes with the same expertise levels may have significantly different return benefit. On the contrary, SMURFEN has a continuous concave utility on the return benefit over expertise

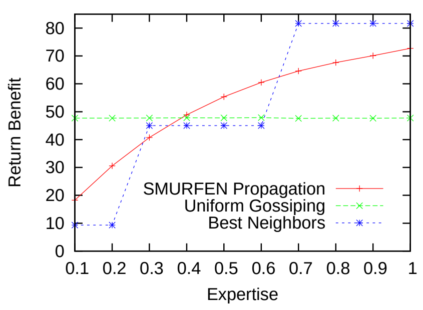

Fig. 11. Incentive on Expertise Level

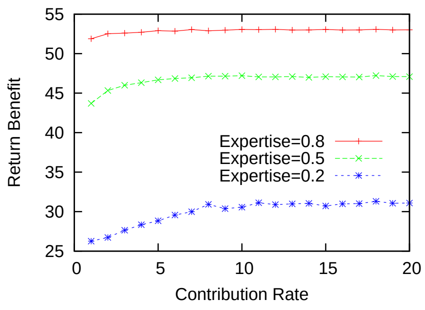

Fig. 12. Incentive of Contribution Rate

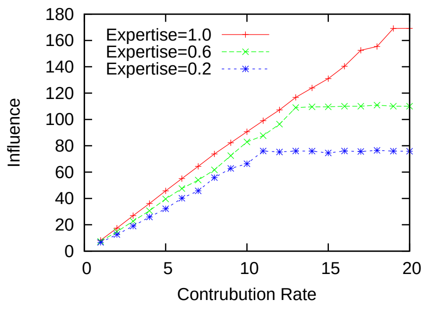

Fig. 13. The Influence vs. Sending Rate

levels. It ensures incentive compatibility as well as fairness. Next we change the rule contribution rate $\bar{r}_0$ of node 0 and observe the return benefit. We can see in Figure 12 that the collaboration benefit increases with rule contribution rate. Contributing more to the collaboration network brings a higher return benefit. Our propagation system is incentive compatible on both contribution quality and quantity.

F. Robustness of the System

The purpose of this experiment is to demonstrate the robustness of the system in the face of denial-of-service attacks. We fix $M = \bar{R} = 100$ rules/day and $\bar{r} = 10$ rules/day for other nodes except the attacker node 0. We let node 0 increase its contribution rate $\bar{r}_0$ from 1 to 20 with unlimited M_0, which is a typical strategy of a spammer. We observe the influence of the node on all the other nodes. We define the influence of a node i to be the total number of rules received by all the neighbors of i per day. The larger the influence of a node, the higher potential of damage the node can cause once it is malicious. We can see from Figure 13 that the influence of a node is bounded in the system. This is because the SMURFEN system enforces propagation agreements between each pair of nodes. Each node sets a rule propagation limit to all its neighbors using the two-level game (see Section IV). Therefore, when a node intends to launch a DoS attack, the amount of rules it is allowed to send to others is bounded by the limits set by its neighbors. Nodes sending excessive traffic to neighbors will be revealed as potential malicious nodes, and thus removed from the neighbor list of others.

VII. Conclusion

In this paper, we have introduced a peer-to-peer rule sharing system called SMURFEN for collaborative intrusion detection and used a game-theoretic framework for its protocol design. The propagation mechanism has been derived from a decentralized two-level optimization problem formulation. We have shown that at the equilibrium our system has the properties of incentive compatibility, and robustness to denial-of-service attacks. Moreover, the system has also been proved to be fair, efficient and scalable. We used Bayesian learning to estimate the compatibility between nodes based on empirical data. By simulation, we have corroborated these important CIDN properties. As future work, we intend to show system robustness to different insider attacks. In addition, under the current rule propagation protocol, we can further demonstrate the macroscopic behavior of SMURFEN arising from multi-hop rule propagations and analyze the time evolution of the rule propagation at the system level.

Appendix

A. Proof of Proposition 1

Proof: From Remark 1, we learn that r^*_{ij} is a monotonic decreasing function with respect to R_{ij} or q_{ij}. Since the utility function in (Pij) is monotonically increasing with r^*_{ji}, increasing R_{ji} will decrease the utility. Hence, an IDS seeks to lower R_{ji} until the optimal utility is achieved to be $U_i^{b\star} = \log(1 + \bar{r}_{ji})$. In other words, an optimal solution R^*_{ji} achieves at

$$r^*_{ji} = \bar{r}_{ji}. \tag{17}$$

Assume that $\bar{r}_i$ is sufficiently large, we have $\bar{r}_{ji} = R_{ji}$. Then R^*_{ji} solves

$$R^*_{ji} = \frac{C_{ij}C_{ji}}{\sum_{u\in\mathcal{N}_j} C_{ju}C_{uj}} \left(M_j + \sum_{v\in\mathcal{N}_j} R_{jv} \right) - R^*_{ji}, \tag{18}$$

which yields (11). It is easy to see that any requests $0 < R_{ji} < R^*_{ji}$ will lower the optimal allocation $r^\star_{ij}$ and hence its utility. ■

B. Proof of Proposition 2

Proof: In **G1**, for each $i \in \mathcal{N}$, the feasible set $\mathcal{F}_i$ is a closed, bounded and convex subset of $\mathbb{R}^{n_i}$. The public utility function U_i^r is jointly continuous in its arguments and strictly convex in $\vec{r}_i$. Hence, using Theorem 4.3 in [25], we can show that **G1** admits a Nash equilibrium in pure strategies.

In **G2**, without relaxation, the convex program (PPi) admits a solution $\tilde{r}_{ij}$, which is continuous in $\vec{R}_i$ [26]. The feasible set of (Pi) is compact and convex and the U_i^b is jointly continuous in its arguments and strictly convex in $\vec{R}_i$. Hence, **G2** has a Nash equilibrium at the level of private optimization. We can determine $r^\star_{ij}$ which yields an equilibrium at the level of public optimization. Therefore, **G2** admits a Nash equilibrium in pure strategies of $\{(\vec{r}_i, \vec{R}_i), i \in \mathcal{N}\}$. ■

C. Proof of Theorem 1

We first introduce a few definitions and then prove Proposition 5, which will be used in the proof of Theorem 1.

Definition 1: Let $\vec{R}_i^*, \vec{r}_i, i \in \mathcal{N}$, be a NE. The non-prime degree $\overline{D}$ of an equilibrium is the number of distinct pairs $\{i,j\}, j \in \mathcal{N}_i$, such that $R_{ij}^* \neq r_{ij}^*$. Note that a prime NE has non-prime degree 0.

Proof: In this proof, we show that any non-prime NE can be reduced to a prime NE with $\overline{D} = 0$. From Proposition 2, we know there exists at least one NE for $\mathbf{G2}$. Let $\mathbf{R}^* = [\vec{R}_i^*]_{i\in\mathcal{N}}$ and $\mathbf{r}^* = [\vec{r}_i^*]_{i\in\mathcal{N}}$ be a NE. Suppose it is not a prime NE. Hence, there must exist at least one pair that satisfies $r_{uv}^* < R_{uv}^*$ for some pair $\{u,v\}$. Construct a feasible solution $(\mathbf{R}', \mathbf{r}^*)$ from $(\mathbf{R}^*, \mathbf{r}^*)$ such that $R_{ij}^{'} = R_{ij}^*$, for every $\{i,j\} \in \bigcup_{i\neq j, j\in\mathcal{N}_i, i\in\mathcal{N}}\{i,j\}\backslash\{u,v\}$, and $R_{ij}^{'} = r_{ij}^*$, for $\{i,j\} = \{u,v\}$. From Proposition 5, we can show that $(\mathbf{R}', \mathbf{r}^*)$ also constitutes a NE, whose non-prime degree becomes $\bar{D}_i - 1$. By an iterative process, a non-prime NE $(\mathbf{R}*, \mathbf{r}^*)$ can be reduced to a prime NE. Hence, there exists a prime NE in $\mathbf{G2}$. ■

Proposition 5: Let $(\mathbf{R}^*, \mathbf{r}^*)$ be a NE with $\bar{D} \neq 0$ and $\{u,v\}$ be a pair of nodes such that $r_{uv}^* < R_{uv}^*$. Let $(\mathbf{R}', \mathbf{r}')$ be a constructed feasible solution such that $\mathbf{r}' = \mathbf{r}^*$, $R_{ij}^{'} = R_{ij}^*$, for every $\{i,j\} \in \bigcup_{i\neq j, j\in\mathcal{N}_i, i\in\mathcal{N}}\{i,j\}\backslash\{u,v\}$, and $R_{ij}^{'} = r_{ij}^*$, for $\{i,j\} = \{u,v\}$. Then $(\mathbf{R}', \mathbf{r}^*)$ is a NE of $\mathbf{G2}$.

Proof: We need to show that $\mathbf{r}^*$ is an optimal response to $\mathbf{R}^{'}$ and then nodes have no incentive to deviate from $\mathbf{R}^{'}$. For a feasible solution $(\mathbf{R}, \mathbf{r})$, we say that r_{ij} is a boundary allocation if $r_{ij} = \min(\bar{r}_i, R_{ij})$; otherwise, we say that r_{ij} is an internal allocation. At an NE solution, the marginal gains $\frac{\partial U_i^r}{\partial r_{ij}}, j \in \mathcal{N}_i$, are equal for internal allocation points. In addition, the marginal gain of i at boundary allocations is no less than the marginal gains of i at internal allocations.

Since $\mathbf{R}^*$ is a $\mathbf{G2}$ NE, node v has no incentive to move by changing R_{uv}. If a node v decreases its request to u from value R_{uv}^* to value r_{uv}^*, then the allocation from node u will not increase. This can be easily shown by contradiction as follows.

Suppose the reverse is true, then there must exist an internal allocation r_{um} to m whose marginal gain is higher than the marginal gain at $R_{uv}^{'}$. However, from (2) and (5), we can see that by understating the requests, nodes can increase their marginal gains. Hence, the marginal gain at r_{um}^* is larger than the marginal gain at r_{uv}^*. Therefore, we can conclude that $\mathbf{r}^*$ is not an optimal solution of configuration $\mathbf{R}^*$, which contradicts with the property of NE.

We also observe that node v can not gain from u by either decreasing or increasing its request at $R_{uv}^{'}$. Decreasing the request results in decreasing the allocation from u, since the resource is bounded by the request. On the other hand, increasing the request at $R_{uv}^{'}$ shall not increase the allocation from u, since it will otherwise contradict with the properties of NE $\mathbf{R}^*$ that nodes v can not gain better utility by changing its request at an NE.

Therefore, after the node v decreases R_{uv}^* to $R_{uv}^{'} = r_{uv}^*$, we arrive at $\mathbf{r}^{'} = \mathbf{r}^*$. The constructed solution $\mathbf{R}^{'}$ and $\mathbf{r}^{'}$ is another NE of $\mathbf{G2}$. ■

D. Proof of Proposition 3

Proof: For each pair of collaborative nodes i, j we have

$$\mathbf{r}_{ij} = \mathbf{A}_{ij}\mathbf{r}_{ij} + \mathbf{b}_{ij}, \tag{19}$$

where $\mathbf{r}_{ij} = [r_{ij}, r_{ji}]^T$, and

$$\mathbf{b}_{ij} = \begin{bmatrix} \lambda_{ij}\left(M_i + \sum_{v\neq j, v\in\mathcal{N}_i} q_{iv}r_{vi}\right) \\ \lambda_{ji}\left(M_j + \sum_{v\neq i, v\in\mathcal{N}_j} q_{jv}r_{vj}\right) \end{bmatrix}, \tag{20}$$

$$\mathbf{A}_{ij} = \begin{bmatrix} 0 & (\lambda_{ij}-1)q_{ij} \\ (\lambda_{ji}-1)q_{ji} & 0 \end{bmatrix} \tag{21}$$

Given that the existence of Nash equilibrium and the assumption on q_{ij} and q_{ji}, the uniqueness of the Nash equilibrium is ensured only when $\mathbf{A}_{ij}$ is non-singular. ■

E. Proof of Proposition 4

Proof: From (12), we can find optimal response R_{ij}^* to other nodes is given by

$$R_{ij}^* = \frac{\lambda_{ij}}{2-\lambda_{ij}}\left(M_i + \sum_{v\neq j, v\in\mathcal{N}_i} R_{iv}\right). \tag{22}$$

Since R_{ij}^* is linear in $R_{iu}, u \in \mathcal{N}_i$, we can build (22) into a linear system of equations with the variables $R_{ij}, i,j \in \mathcal{N}$ stacked into one vector. The linear system has a unique solution if the condition of diagonal dominance holds, leading to the condition. ■

References

[1] "ZDnet." http://www.zdnet.com/blog/security/confickers-estimated-economic-cost-91-billion/3207 [Last accessed in Nov 18, 2010].

[2] R. Vogt, J. Aycock, and M. Jacobson, "Army of botnets," in *ISOC Symp. on Network and Distributed Systems Security*, 2007.

[3] T. Holz, M. Engelberth, and F. Freiling, "Learning more about the underground economy: A case-study of keyloggers and dropzones," in *ESORICS'09*.

[4] "The honeynet project. know your enemy: Fast-flux service networks," 13 July, 2007. http://www.honeynet.org/book/export/html/130.

[5] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 1, pp. 20–35, 2008.

[6] J. Ullrich, "DShield." http://www.dshield.org/indexd.html.

[7] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the domino overlay system," in *NDSS'04*.

[8] M. Cai, K. Hwang, Y. Kwok, S. Song, and Y. Chen, "Collaborative internet worm containment," *IEEE Security & Privacy*, vol. 3, no. 3, pp. 25–33, 2005.

[9] C. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Trust management for host-based collaborative intrusion detection," in *19th IFIP/IEEE International Workshop on Distributed Systems*, 2008.

[10] J. Oberheide, E. Cooke, and F. Jahanian, "Cloudav: N-version antivirus in the network cloud," in *Proceedings of the 17th USENIX Security Symposium*, 2008.

[11] J. Goodall, W. Lutters, and A. Komlodi, "I know my network: collaboration and expertise in intrusion detection," in *ACM conference on Computer supported cooperative work*, 2004.

[12] C. Fung, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," in *11th IFIP/IEEE International Symposium on Integrated Network Management*, 2009.

[13] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, "A trust-aware, p2p-based overlay for intrusion detection," in *DEXA Workshops*, 2006.

[14] D. Dash, B. Kveton, J. Agosta, E. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman, "When gossip is good: Distributed probabilistic inference for detection of slow network intrusions," in *AAAI'06*.

[15] G. Zhang and M. Parashar, "Cooperative detection and protection against network attacks using decentralized information sharing," *Cluster Computing*, vol. 13, no. 1, pp. 67–86, 2010.

[16] V. Ramasubramanian, R. Peterson, and E. G. Sirer, "Corona: a high performance publish-subscribe system for the world wide web," in *NSDI'06*.

[17] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi, "Meghdoot: content-based publish/subscribe over p2p networks," in *Middleware '04*.

[18] "Snort." http://www.snort.org/ [Last accessed in Nov 18, 2010].

[19] "OSSEC." http://www.ossec.net/ [Last accessed in Nov 18, 2010].

[20] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM*, pp. 149–160, ACM, 2001.

[21] http://www.csoonline.com/article/593237/inside-sourcefire-s-vulnerability-research-team?page=2 [Last accessed in Nov 18, 2010].

[22] M. Roesch and C. Green, "Snort users manual," *Snort Release*, vol. 1, April 2010.

[23] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[24] F. Olver, D. Lozier, R. Boisvert, and C. Clarke, "NIST handbook of mathematical functions," *National Institute of Standards and Technology, Gaithersberg, Maryland*, 2009.

[25] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. SIAM Series in Classics in Applied Mathematics, 1999.

[26] S. Zlobec, *Stable Parametric Programming*. Springer, 1st ed., 2001.