Collaborative Defense against Periodic Shrew DDoS Attacks in Frequency Domain*

YU CHEN, KAI HWANG, and YU-KWONG KWOK

University of Southern California, Los Angeles, CA 90089, USA

Abstract: The shrew or pulsing DDoS (*Distributed Denial-of-Service*) attacks, also known as RoQ (*Reduction of Quality*) attacks, are stealthy, periodic, and low-rate in volume. The shrew attacks could be even more detrimental to network resources than the flooding type of DDoS attacks. Shrew attacks appear periodically in low volume, thereby damaging the victim servers for a long time without being detected. This in turn leads to denying new visitors to the victim servers (which are mostly e-commerce sites). Hence, there is a pressing need to effectively detect shrew attacks in real-time, and to fend off these attacks at the victim sites at the earliest possible time. Unfortunately, there is still a void in research on effective detection of shrew DDoS attacks. In this paper, we propose a new *digital signal processing* (DSP) approach to detecting the shrew attacks embedded in legitimate traffic flows.

We detect with the frequency-domain characteristics from the autocorrelation sequence of Internet traffic streams. This approach enables collaborative detection across multiple routers. The new detection scheme appeals to hardware implementation based on DSP technology. Our new detection technique requires only a few seconds for successful detection of shrew DDoS attacks. Furthermore, the technique entails only lightweight implementation in a real-life network environment. We developed a network-layer multicast protocol *LocalCast* to support collaborative detection without burdening the end hosts. The protocol adds more intelligence to active networks and enables the routers to filter out malicious traffic flows with minimum collateral damage to legitimate traffic flows. This frequency-domain defense scheme provides more robustness and intelligence in IP networks, which will enable the integration of more security functionality into Internet routers to protect sensitive applications.

Categories and Subject Descriptors: C.2.1 [Computer Communication Networks]: Security and Protection

General Terms: Network Security, Internet Technology, Intrusion Detection

Keywords: Network anomalies, Distributed Denial-of-Service (DDoS), Reduction of Quality (RoQ), periodic TCP attacks, autocorrelation function, hypothesis test, and digital signal processing.

Table of Contents: (To be removed after review)

---

* Manuscript submitted to *ACM Transactions on Information and System Security* (TISSEC) on May 3, 2005. The research work reported here was supported by a NSF ITR Grant 0325409. Corresponding author: Kai Hwang, USC Internet and Grid Computing Lab, EEB 212, Los Angeles, CA 90089. E-mail: kaihwang@usc.edu, Tel.: (213) 740-4470.

1. INTRODUCTION

Distributed Denial-of-Service (DDoS) attacks have become one of the major threats to Internet services and electronic transactions [CHANG *et al.* 2002; MOORE *et al.* 2001; SPECHT *et al.* 2004]. A typical DDoS attack prevents legitimate users from accessing the victim site for certain services. The network resources could be denied by overwhelming the target with huge amount of traffic launched from many innocent *Zombies*. Essentially, this kind of attack is targeting at undermining the availability of certain Internet services. A traditional DDoS attack can be characterized as brute-force, sustained high-rate, or specifically designed to explore the protocol limitations or software vulnerabilities in servers. DDoS attacks degrade the performance of the networks even when the links are not saturated [LAN *et al.* 2003].

Recently, a variant category of DDoS attack has been identified by [DELIO 2001] on the Internet2 Abilene backbone. This periodic pulsing attack exploits the transients of a system's dynamic behavior. These low-rate pulsing DDoS attacks introduce system inefficiencies that tremendously reduce the system capacity or service quality. In the literature, this kind of DDoS attack is called *shrew attack* [KUZMANOVIC and KNIGHT 2003], pulsing DoS attacks [LUO and CHANG 2005], or the *Reduction of Quality* (RoQ) attack [GUIRGUIS *et al.* 2004, 2005]. Comparing to traditional DDoS flooding attacks, shrew attacks are even harder to detect. The shrew attacks can damage the victim for a long time without being detected [GUIRGUIS *et al.* 2005]. With a prolonged period of service degradation, an e-commerce Web site (e.g., Amazon.com) may experience significant financial losses or losing some new customers for disrupted services provided.

Taking advantage of vulnerabilities of a system's dynamic behavior, shrew attacks could achieve similar effects of traditional DDoS attacks, while escaping from detection by occupying unsuspicious fraction of resources. Instead of constantly injecting traffic flows with a huge rate into the network, shrew attackers send burst pulses periodically. Such low-rate attacks have high peak rate while maintaining a low average rate to exhibit a stealthy behavior. Shrew attacks exploit the deficiencies in the RTO (*Retransmission Time-Out*) mechanism of TCP. It throttles legitimate TCP flows by periodically sending burst pulses with high data rate. As such, the TCP flows always see congestion on the attacked link every time it recovers from RTO. Indeed, such a shrew attack may cut the throughput of TCP applications down to almost zero [KUZMANOVIC and KNIGHT 2003].

Given that more than 80% of traffics on Internet today are using TCP protocol [LAN *et al.* 2003], a majority of existing applications and commercial services are at stake. Unfortunately, it has been proven theoretically and experimentally that countermeasures developed for traditional DDoS attacks are not effective in fighting against shrew or pulsing attacks [GUIRGUIS *et al.* 2005; KUZMANOVIC and KNIGHT 2003; MAHAJAN *et al.* 2001]. Furthermore, being masked by the background traffic, shrew attacks are very difficult to identify in the time domain, which is the usual avenue of defense in combating network attacks. In the GridSec project at USC, we have developed complementary security schemes for

trusted Grid computing [SONG et al. 2005; HWANG et al. 2005], Internet worm detection and containment [CAI et al. 2005], and the defense of flooding type DDoS attacks [CHEN et al. 2005].

In recent years, researchers have explored the usage of signal analysis technology in traffic analysis for network security enforcement [ABRY *et al.* 1998; ABRY *et al.* 2002; BARFORD 2002; HUANG *et al.* 2001; PARTRIDGE *et al.* 2002]. Due to the behaviors defined by protocols or applications, the periodicity of traffic could be used as a signature for traffic monitoring or attack detection. The dominant link frequency is independent of the number of flows. Instead, it depends on the link bandwidth and packet size distribution [HE *et al.* 2004]. TCP traffic exhibits a periodicity on its PSD (*Power Spectrum Density*), when the packet arrival rate is analyzed in frequency domain. Thus, the lack of periodicity could indicate that DoS attacks are raging on [CHENG *et al.* 2002].

In contrast, the PSD of multi-sourced DDoS attacks are distributed in lower frequency band comparing to single-sourced DoS attacks [HUSSIAN *et al.* 2003]. However, none of these research works are capable of distinguishing malicious attack flows from legitimate ones. Previously, we proposed a HAWK algorithm by judiciously identifying malicious shrew packet flows [KWOK *et al.* 2005]. The HAWK scheme applies only to DoS attacks from a single source. [Sun *et al.* 2004] suggested detecting shrew attacks by matching pattern with obtained attack signature. They use a *deficit round robin* (DRR) algorithm to allocate bandwidth and protect legitimate flows. However, their method cannot distinguish malicious and legitimate flows accurately. Legitimate flows are still suffering in throughput significantly.

Recently, [LUO and CHANG 2005] have studied the characteristics of low-rate TCP-targeted DoS attack using a wavelet approach. They observed anomalies in fluctuation of incoming traffic rate and declining of outgoing TCP ACKs incurred by pulse streams. Based on these properties, they proposed a two-stage algorithm to detect the existence of pulse streams. In the first stage, they employed *Discrete Wavelet Transform* (DWT) to monitor these anomalies since DWT could capture the variability of incoming traffics and extract the trend of the outgoing TCP ACK traffics. A special CUSUM method was adopted to detect the changing points in the second stage.

When incoming traffic fluctuation and outgoing TCP ACK signals reach certain thresholds, the detection system confirms the existence of low-rate attack streams. They have verified that their approach could detect pulse streams effectively through NS-2 simulation and experiments on a NIST Net testbed. Unfortunately, they did not report the detection accuracy achieved. Furthermore, since the wavelet detection outcomes are largely dependent on the choice of detection parameters, it is difficult to find optimal parameters that are sensitive enough to detect low-rate distributed attacks while maintaining a acceptable false positive alarm rate.

In this paper, we propose a novel approach to detecting the shrew or pulsing attack streams from legitimate traffic flows. Our unique novelty is the combined use of DFT (*Discrete Fourier Transform*) and a hypothesis framework to cope with shrew DDoS attacks. To our best knowledge, this approach has not been challenged before. By calculating the autocorrelation sequence of the sample series and converting

them into the frequency domain using DFT, we find that the PSD of traffic containing shrew attacks has more energy in the low-frequency band than legitimate traffic does. A *shrew-filtering algorithm* is thus developed to identify whether a single flow is malicious or not.

The performance results of the newly shrew DDoS defense scheme are based on extensive NS-2 simulation experiments. By analyzing more than 8,000 simulated sample points, we cut off malicious flows with an accuracy of 99.9%. To support collaboration among distributed routers, we developed a network-layer multicast protocol called *LocalCast.* This protocol allows routers to exchange local detection results with its peers by joining existing multicast groups to establish its own multicast tree. For the convenience of readers, the notation and abbreviations used throughout the paper are summarized in Table 1.

Table 1. Notations and Abbreviations used in Paper

Symbol	Definition	Abbrev.	Definition
T	Attack period (sec)	RoQ	Reduction of Quality attacks
R	Attack pulse peak rate	DDoS	Distributed Denial of Service attacks
L	Attack pulse burst Length (s)	DSP	Digital Signal Processing
N_s	No. of shrew attack flows	NCPSD	Normalized Cumu. Power Spectrum Density
N_t	No. of TCP flows	NCAS	Normalized Cumulative Amplitude Spectrum
P_D	Anomaly detection rate	NFT	Nice flow table
P_{FP}	False positive alarm rate	SFT	Suspicious flow table
P_{FN}	False negative alarm rate	PDT	Permanent drop table
α	Detection accuracy	CoAlert	Collaborative alert algorithm
ρ	Normalized Throughput	LocalCast	Local Cast protocol
τ	Response Time		

The rest of this paper is organized as follows. In Section 2, we present the theoretical foundation of collaborative anomaly detection. Then we present frequency spectrum analysis techniques in Section 3, which also includes the distributed algorithms for anomaly detection and autocorrelation. The network-layer *LocalCast* protocol is described in Section 4. Section 5 presents the adaptive filtering algorithm for cutting off malicious flows of shrew attacks. Our simulation setup and performance results are reported in Section 6. Finally, we summarize the research findings and comment on further research work needed.

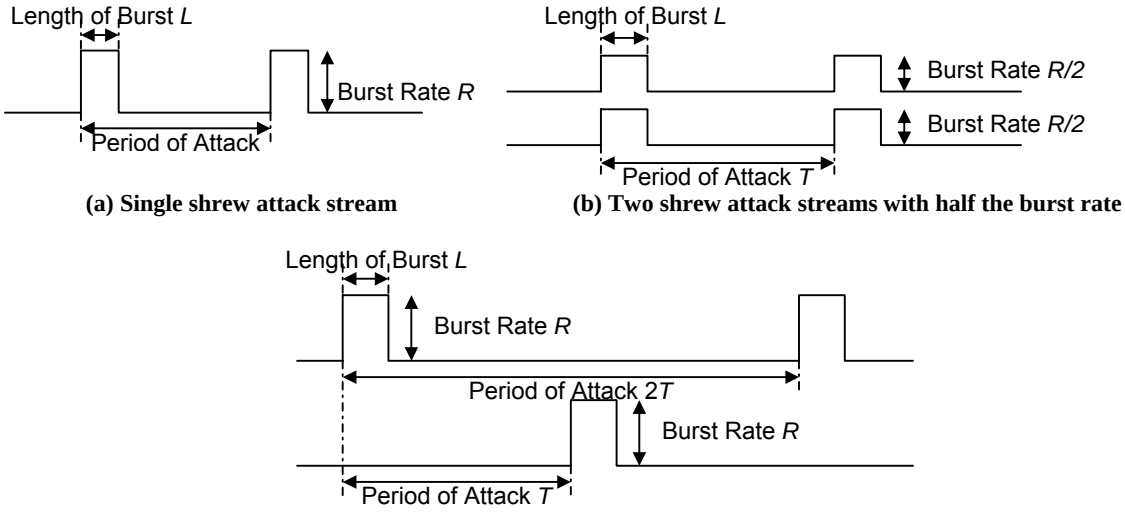
2. CHARACTERISTICS OF SHREW DDoS ATTACKS

In this section, we describe the properties of shrew attacks in both time and frequency domains. Then we analyze the autocorrelation functions on packet arrival rates. The attack properties in frequency domain set up the framework of our detection scheme.

2.1 Overview of Shrew DDoS Attacks

[KUZMANOVIC and KNIGHT 2003] pioneered the characterization of TCP targeted shrew attacks. They studied the rationale of the shrew attacks and identified the critical parameters that affect the

TCP flows. They indicated the limitation of using existing DDoS defense mechanisms against shrew attacks. However, they did not develop specific countermeasures against the low-rate attacks. As shown in Fig. 1, a single source shrew attack is modeled by a square waveform with a period of attack *T*, length of burst *L*, and the burst rate *R*. The period *T* is the time interval between two consecutive attack pulses. The burst length *L* indicates the time period during which attackers send packets in high rate. And the burst height exhibits the peak rate by which attacking flow is sent. The period *T* is calculated by the estimated TCP RTO timer implementation from trusted sources. During the burst with a peak rate *R*, the shrew pulses create a bursty and severe congestion on the links to the victim. The legitimate TCP flows must decrease their sending rate as governed by rate-limiting mechanism, accordingly.



(a) Single shrew attack stream

(b) Two shrew attack streams with half the burst rate

(c) Two shrew attack streams with doubled period and the same burst rate

Figure 1. Periodic characteristics of shrew attack streams from single source and multiple sources.

For higher throughput, the TCP protocol uses a predefined value of RTO with a fixed incrementing pattern [PAXSON and ALLMAN 2000]. The shrew attack takes advantage of this RTO recovery mechanism by adjusting its attack period. The mechanism occupies the link bandwidth periodically by sending pulses shown in Fig. 1(a). This makes the legitimate TCP flows face a heavily burdened link when they try to send packets. Such legitimate TCP flows must undergo congestion control and reduce their rates accordingly. It was estimated that a successful shrew attack might make the throughput of legitimate TCP traffics lower than 10% of the normal level.

It should be noted that in the case where legitimate TCP flows and shrew flows both are heading for the same destination, the shrew flows exhibit two important different behaviors: (1) the shrew flow's peak rate will remain *constant* while the TCP flow will increase linearly; and (2) the shrew flow will arrive at the destination in more or less fixed periods of time while the TCP flow will have a *continuous* arrival. However, periodic pulses are difficult to detect using existing traffic volume analysis method at the time domain. This is because the average share of bandwidth consumption is not remarkably high. In a

distributed scenario, attacks launched by multiple zombies could lower their individual traffic rates further, thereby making the detection even harder. As shown in Fig. 1(b) and 1(c), the distributed attack sources could decrease its average traffic either by lowering the peak rate or by using longer attack periods. Detecting the signs of such attacks using time series is therefore totally ineffective.

2.2 Autocorrelation Power Spectrum Analysis

Although general defense mechanisms against flooding DDoS attacks are unsuitable in detecting the low-rate shrew attacks, the periodicity itself provides a clue for developing new defense mechanisms. Periodic signals and aperiodic signals present different properties in the frequency domain. The differences could be detected conveniently using signal-processing techniques. Packet arrivals are often modeled by a stochastic process that is characterized by a set of probability distributions. While stochastic signals do not directly have Fourier transforms, the Fourier transform of its autocorrelation sequences provides useful interpretation of its characteristics in frequency domain.

With a simple analysis of the autocorrelation function, we discover a few important characteristics that are useful in our detection process. Two basic questions are often asked: Whether the attack time series follows a periodic pattern and what may be the expected frequency distribution? Unlike the flooding DDoS attacks that could be recognized by the victim, when it is overwhelmed by the huge traffic flows, the low-rate attacks may throttle the legitimate TCP flows destined to the victim. Therefore, the detection has to be deployed in the upstream routers, one or more hops away from the end server being protected.

We take the number of packet arrivals at the router as the discrete signal series and sample it with a period of 1 *ms*. According to Nyquist sampling theorem [OPPENHEIM and SCHAFER 1999], we can use the highest frequency of 500 Hz. Our sampling effectively plays the role of low pass filter that gets ride of high frequency noises. Another observation is, besides the shrew attack streams, that the sample also includes packets from legitimate flows. These packets are not necessarily all targeting at the same victim.

The packet arrivals are modeled by a random process: $\{X(t), t = n\Delta, n \in N\}$, where Δ is a constant time interval, which in our case is 1 *ms*. N is the set of positive integers, and for each t, $X(t)$ is a random variable. $X(t)$ represents the total number of packet arrivals at one router in $(t-\Delta, t]$. This random process is referred to as *packet process*. We assume a *Wide Sense Stationary* random process. Therefore, we define the autocorrelation function of the random signal $X(t)$ in the discrete-time case as:

$$R_{xx}[m] = \sum_{n=-\infty}^{\infty} (x[n]x[n+m]) \quad \textbf{(1)}$$

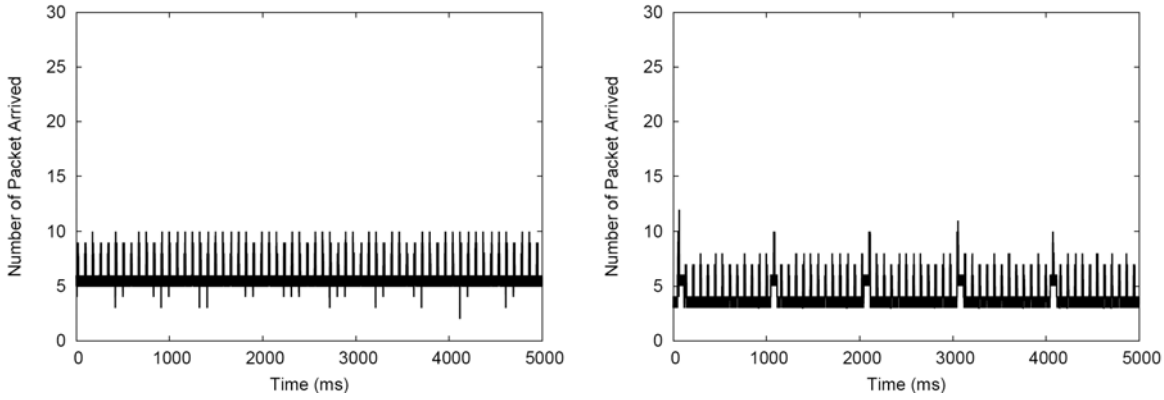
However, the whole random process is unavailable to us, and we still need to find out more characteristics of the random process. We estimate below the autocorrelation at the given interval:

$$R_{xx}[m] = \frac{1}{N-m} \sum_{n=0}^{N-m+1} (x[n]x[n+m]) \quad \textbf{(2)}$$

$R_{xx}[m]$ captures the correlation of the packet process and itself at interval m. If there is any periodicity exist, autocorrelation function is capable of enforcing it. The next step is to figure out the periodicity embedded inside the autocorrelation functions. The Fourier transform of the autocorrelation sequence brings the useful interpretation of the frequency distribution of the signal. We convert the autocorrelation time series by *Discrete Fourier Transform* (DFT) [ALLEN and MILLS 2004] and obtain its PSD as follows:

$$DFT(R_{xx}(m), K) = \frac{1}{N}\sum_{n=0}^{N-1} R_{xx}(m) \times e^{-j2\pi kn/N} \qquad \text{k=0,1,2,…,N-1} \qquad \textbf{(3)}$$

In the frequency domain, by using the energy distribution, we can detect whether there is any shrew stream embedded in legitimate flows.



(a) Number of packets arriving without shrew attack stream in it

(b) Number of packets arriving with one TCP, one UDP, and one shrew attack stream

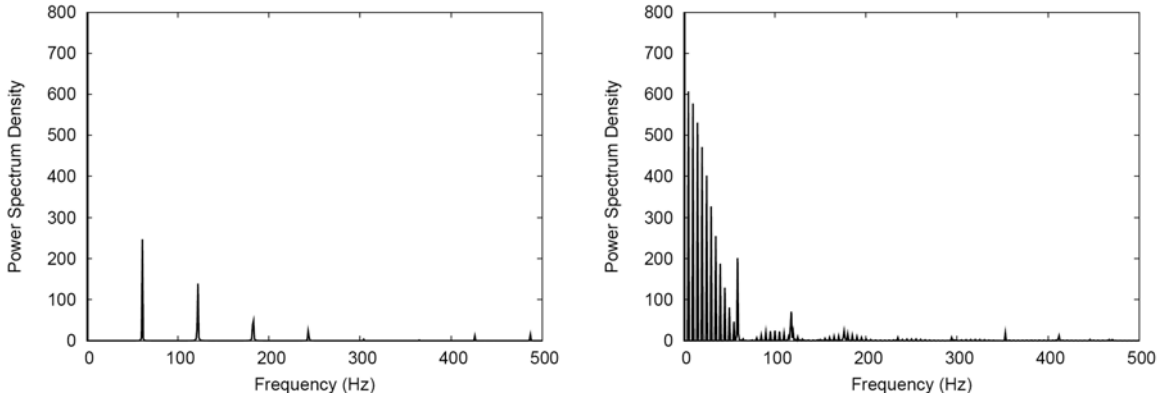
Figure 2. Comparison of traffic time series patterns, with and without shrew attack streams.

Figure 2 compares the time series and the autocorrelation of two scenarios: (1) One TCP flow plus one constant rate UDP flow with a rate of 300Kb/sec (see Fig. 2(a)); (2) The combination of one TCP flow, one constant rate UDP flow with rate 300Kb/sec, and one shrew attack stream with an attacking period of 1 sec and a peak rate of 200Kb/sec (see Fig. 2(b)). We can see that the shrew attack stream hides itself among normal traffic by making its peak rate even lower than the constant rate UDP flow. Before the link is saturated, the traffic volume analysis scheme may not be able to detect such a stealthy attack.

2.3 Amplitude and Power Spectrum

In the scenario shown in Fig.2(b), shrew attack stream escapes detection by occupying small share of bandwidth. However, the autocorrelation sequence will amplify the influence the periodical pattern of shrew attack stream has after it is converted into frequency domain. What exactly happens is that more power of the autocorrelation function is distributed in the lower frequency band if there is shrew stream contained in that traffic. Figure 3(a) shows the power spectrum density curve of autocorrelation sequence of traffic series consisting of one TCP flow and one constant rate UDP flow and Fig. 3(b) is the PSD

distribution of autocorrelation sequence of same legitimate traffics but plus one shrew stream. We can see clearly that the shrew attack generates a large amount of energy in the low frequency band, making its presence rather apparent. As mentioned above, the highest frequency of our analysis is 500 Hz.

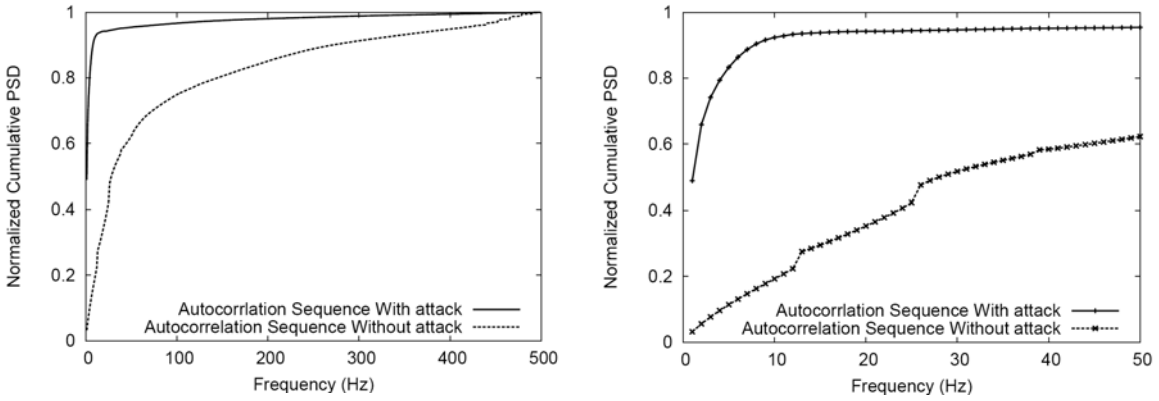


(a) Power Spectrum Density for traffic without shrew stream

(b) Power Spectrum Density for traffic containing shrew stream

Figure 3. Comparison of the power spectrum density of autocorrelation function containing shrew attacks and legitimate traffics.

Figure 4 presents the *normalized cumulative PSD* (NCPSD) curve of autocorrelation function of packet process. In the figure, we compare NCPSD curves of cases we discussed in Fig. 2. It is clear that more than 90% of the packet process's energy distributes in frequency band [0, 20] Hz if the traffic contains a shrew stream. By contrast, if there is no shrew stream contained, the energy located in this low frequency band is less than 40%. This implies that NCPSD is a robust criterion in detecting whether current sampled traffic contains shrew streams.



(a) NCPSD in entire frequency band

(b) NCPSD in low frequency band

Figure 4. Comparison of the normalized cumulative PSD of shrew attack streams with that of the normal traffic containing no shrew attacks.

3. Collaborative Anomaly Detection

In this section, we present our proposed autocorrelation based attack features extraction technique based on the analysis above. Our approach works by performing power spectrum data gathering locally at

each router, followed by a distributed autocorrelation data combining and hypothesis testing involving multiple routers.

3.1 Hypothesis Test Analysis

From Fig. 4, we can see that by choosing one detection point in the low frequency band where the distance between autocorrelation's NCPSD curves is large enough for us to tell whether there is a shrew stream hidden in legitimate flows. The result of the inspection is either there is shrew stream(s) embedded in legitimate traffic flows, or no shrew attacks. We set up the hypothesis test as follows:

$$\begin{cases} H_0 : No, \ \ Shrew \ \ Stream \ \ NOT \ \ Exists; \\ H_1 : Yes, \ \ Shrew \ \ Stream \ \ Exists; \end{cases}$$

Assume that the NCPSD distribution of traffic having no shrew attack streams is $N(\mu_0, \sigma_0^2)$, and the NCPSD distribution of traffic containing shrew attacks is conform to $N(\mu_1, \sigma_1^2)$. We obtain the probability distributions by collecting more than 8,000 data points with difference traffic patterns. According to the *Central Limit Theorem*, given an unknown distribution with a mean μ and variance σ^2, the sampling distribution follows a *Gaussian* distribution, if the number of sampling is large enough.

$$G(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \qquad \textbf{(4)}$$

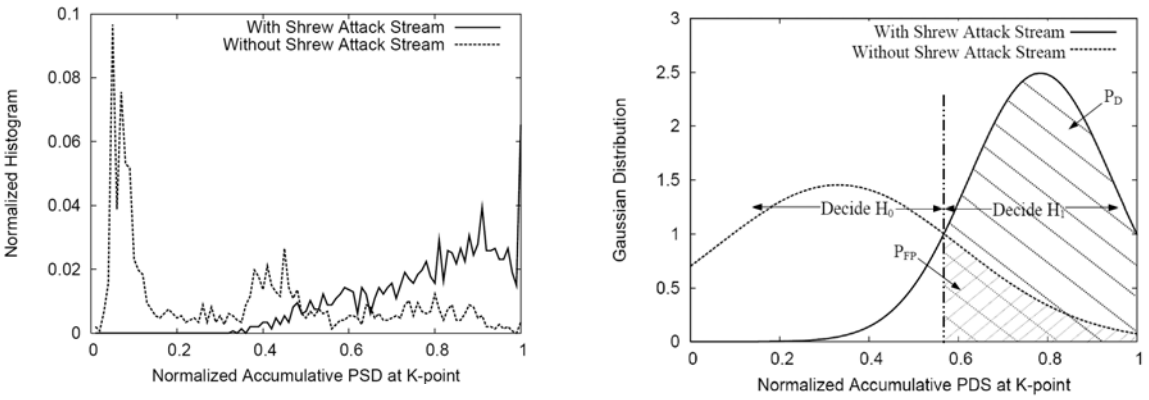
We define the error $P(H_i, H_j)$ as the probability of having the object H_i when it is indeed H_j. Therefore, we need to choose a detection threshold γ to maximize the *detection probability* P_D and minimize both *false negative alarm rate* P_{FN} and *false positive alarm rate* P_{FP} probabilities as follows:

$$\begin{cases} P_D = P(H_1;H_1) = P_r\{x > \gamma; H_1\} = \int_{\gamma}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_1}\exp\left\{-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right\} \\ P_{FN} = P(H_0;H_1) = P_r\{x < \gamma; H_1\} = \int_{-\infty}^{\gamma} \frac{1}{\sqrt{2\pi}\sigma_1}\exp\left\{-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right\} \\ P_{FP} = P(H_1;H_0) = P_r\{x > \gamma; H_0\} = \int_{\gamma}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_0}\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\} \end{cases} \qquad \textbf{(5)}$$

As suggested by the NCPSD curves in Fig. 4, we see that the distance between curves of traffic with and without shrew attack streams is the maximum around the frequency point 20 Hz. This implies that the maximum P_D and minimum false negative P_{FN} and false positive P_{FP} will be achieved. As such, we call this point as the *K-point*. In Section 6, our simulation on the flow level verifies that 20 Hz is the optimal point to identify whether a single flow is a shrew attack stream or not. The normalized histogram and Gaussian distribution of the NCPSD distribution of traffic at the K-point are shown in Fig. 5(a) and Fig. 5(b), respectively. The Gaussian distribution curves are drawn using the statistical results in the sample space with more than 3,000 data points:

$$Traffic\_Without\_Shrew: \begin{cases} Average(\mu_0) = 0.3310, \\ S\tan dard\_Deviation(\sigma_0) = 0.2746 \end{cases}$$

$$Traffic\_Containing\_Shrew: \begin{cases} Average(\mu_1) = 0.7815 \\ S\tan dard\_Deviation(\sigma_1) = 0.1641 \end{cases}$$



(a) Normalized histogram obtained by simulation **(b) Gaussian distribution curves and detection rule**

Figure 5. The NCPSD distribution of traffic with and without shrew attacks.

As shown in Fig. 5, there is some crossing area under NCPSD distribution curves of two hypotheses. It is clear that there does not exist a cutting point that could achieve a 100% detection probability P_D and maintain a 0% false positive alarm rate P_{FP} simultaneously. In another words, the 0% false negative probability P_{FN} is accompanied by a high false positive alarm rate P_{FP}. Therefore, we need to find an optimal cutting point that may approach an enough high detection probability with acceptable false negative/positive alarm rate.

In order to tell whether there is any shrew attack stream embedded in a legitimate flow, we set up our hypothesis test rule according to *Neyman-Pearson Theorem* [KAY 1999]. Let's define the *Likeliness Ratio Function L(x)* for each test result x as the likelihood of event H_1 versus the likelihood of event H_0:

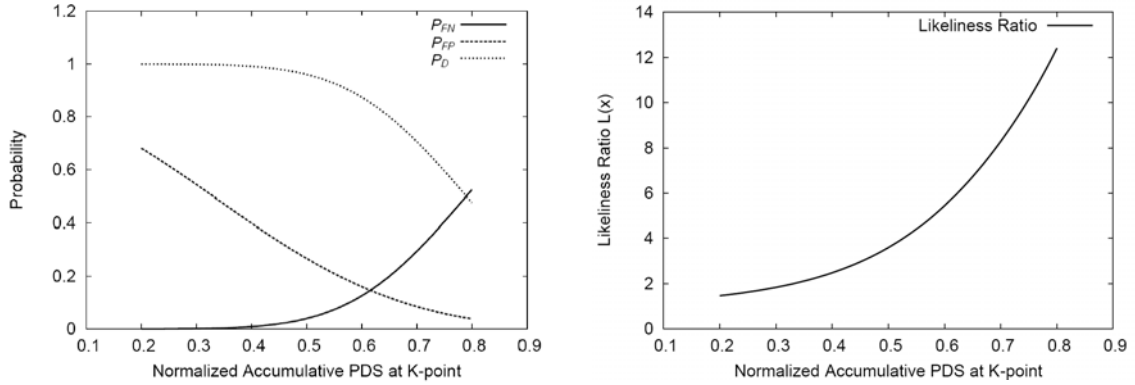
$$L(x) = \frac{p(x; H_1)}{p(x; H_0)} \qquad \textbf{(6)}$$

If $L(x) > \gamma$, then hypothesis H_1 is true, otherwise, H_0 it true. γ is certain detection threshold that is associated with false negative alarm rate.

Equation (6) also verifies what we have intuitively pointed out above that the maximum detection probability is closely related to the false positive alarm rate. Based on the simulation data, we calculated $L(x)$, P_D, P_{FN}, and P_{FP} using Equations (5) and (6) while different cutting points are applied. We see from Fig. 6 that with lower threshold, we may obtain higher detection probability and lower false negative alarm rate with the cost of higher false positive alarm rate. Since the alert may demand other flow identification mechanism, high false alarms would increase the router's burden unnecessarily.

On the other hand, by choosing a higher threshold, we can decrease the false positive alarm rate at the cost of lower detection probability. However, our distributed detection scheme discussed in next subsection is very effective to make up this disadvantage. By taking this tradeoff into account, we finally

selected the threshold γ=5.26 that corresponds to cutting point where *NCPSD*=0.6. This threshold gives us a detection probability P_D=0.880, false negative alarm rate P_{FN}=0.120, and false positive alarm rate P_{FP}=0.167.



(a) Detection probability P_D, false negative alarm rate P_{FN} and false positive alarm rate P_{FP}

(b) Likeliness ratio function

Figure 6. Anomaly detection rate P_D, false negative alarm rate P_{FN} , and false positive alarm rate P_{FP} and the likeliness ratio function under various cutting points.

3.2 CoAlert Anomaly Detection Algorithm

With the above hypothesis testing analysis and the parameters derived, we propose a distributed detection scheme called ***CoAlert*** (cooperative alert) based on a new network layer protocol named ***LocalCast*** protocol that constructs an overlay network purely in the network layer without involving any effort from end hosts. CoAlert algorithm enables routers exchange information efficiently to perform cooperative detection against distributed shrew attack streams. As mentioned before, the detection measurements have to be deployed in upstream routers that are several hops away from the server because the low-rate attacks may throttle the legitimate TCP flows destined to the victim. However, before reaching their attacked targets, the shrew streams may occupy a small share of bandwidth and the signal is too weak to be detected if we only look at local sampled data series.

Therefore, the traffic information of the neighbor routers could be useful references for a router to double-check its detection result. To avoid having large false positive alarm rate, we have tolerated some high false negative alarms. Algorithm 1 tries to make up this deficiency by selecting second threshold γ_c = 3.6 targeting at a higher detection rate (0.96), where the subscript "*c*" stands for cooperative threshold.

While *L(x)* is higher than the threshold γ, the router would start the shrew filtering mechanism and multicast alert to its neighbors located in two hops' distance. Routers whose *L(x)* is smaller than γ but larger than γ_c do not generate any alert but they need to decide whether to start local shrew filtering mechanism by analyzing the resources of alerts received from their neighbors. If *L(x)* is less than γ_c, there is nothing suspicious. Considering the potential pattern of distributed shrew attack, alerts from immediate routers are closer related to the receiver. Thus, we give received alerts two weights corresponding to the distance between the sender and receiver, and the immediate senders' alerts are weighted higher.

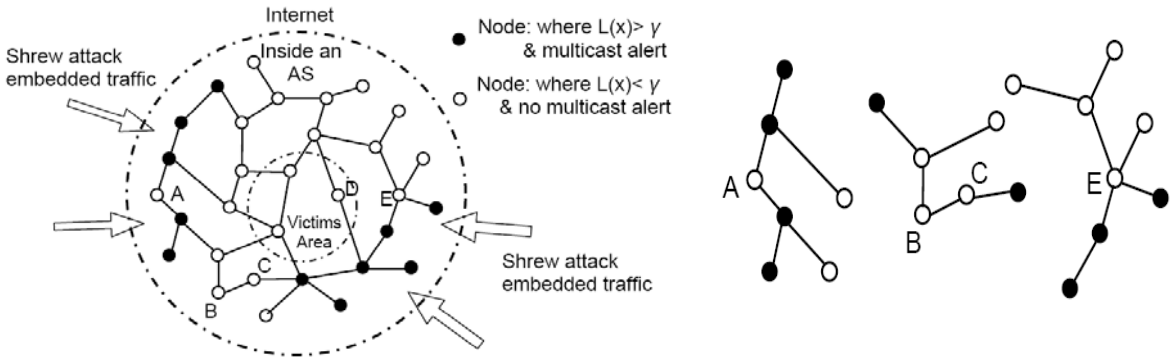
Algorithm 1: Collaborative Anomaly Detection (CoAlert)

```
01: While shrew detection algorithm is on
02:     If L(x) > γ Then
03:         Multicast alert to neighbor routers in two hops' distance
04:         Trigger the local shrew-filtering mechanism
05:     Else
06:         If L(x) > γc Then
07:             Listen the alerts from neighbor routers
08:             If more than 1/2 of immediate routers multicast alert Then
09:                 Start local shrew filtering mechanism
10:             Else If less than 1/2 of immediate routers plus their immediate
                        neighbors multicast alert Then
11:                 Start local shrew filtering mechanism
12:             Else
13:                 Do not start shrew filtering
14:     Else
15:         Do not start shrew filtering
```

Then the router who does not multicast alert would decide if it should start the shrew filtering mechanism on analyzing received alerts. The rationale of this mechanism is that distributed streams belonging to same attack are convergence while getting closer to the victim. Therefore, receiving alert indicates that the router is located close to or inside the range of suspicious traffics that may travel through. If most its immediate neighbors have detected shrew streams, plus the local likelihood function $L(x)$ is larger than γ_c, it is reasonable to start the shrew filtering process to check flows passing by.

Let us take the scenario in Fig. 7(a) as an example. Each node in the figure stands for a router in the AS, and the black nodes are routers that have detected shrew streams embedded in legitimate flows and multicast alert to its neighbors in two hops' distance. The white nodes are routers whose likeliness ratio $L(x)$ is lower than the threshold γ, so they do not generate alert. Moreover, each node is the root of a *decision making tree* (DMT) that contains all its neighbor in two hops' distance, three examples of DMTs of node A, B and E are presented in Fig. 7(b). These DMTs also play as the multicast group records that each root knows to whom its alert is multicast.

Distributed attack is launched from zombies located widely and multiple shrew streams are approaching to victim(s) located in the AS. Edge routers who detected $L(x) > \gamma$ (black node in Fig. 7) would multicast an alert along its DMT and start the shrew filtering mechanism to identify malicious flows and cut off them completely. However, as the randomly distribution of shrew streams, their strengths are not even to each edge router. The CoAlert mechanism gives routers a clue to check whether they are in the attacking range. When $L(x) < \gamma$ happens at node A, alerts from neighbors tell that it is a real attack.



(a) Distributed shrew attacks from two sides of the AS **(b) Typical decision making trees**

Figure 7. The *CoAlert* architecture and typical decision making tree.

Furthermore, the DMT let node A realize that it is located at the center of the attacking area. Then if $L(x) > \gamma_c$, node A starts the shrew filtering mechanism. In contrast, node B, who also received two alerts, will not trigger the shrew filtering. As all its immediate neighbors have not raised any alert, the further sourced alert just leads to a conclusion that there are suspicious flows coming into the AS, but node B is not in the attacking area. However, the DMTs and received alerts would tell nodes C, D and E that they are at the edge of attacking area. If there exists $L(x) > \gamma_c$, then they start the shrew filtering mechanism.

4. LOCALCAST PROTOCOL FOR ALERT CORRELATION

As shown above, routers whose $L(x)$ is less than the detection threshold reaches its decision based on multicast alerts from its neighbors. The CoAlert algorithm is implemented over a network layer protocol *LocalCast* that allow routers to construct an overlay network on the network layer without involved any efforts from the end hosts. In this section, we discuss design of our LocalCast protocol in detail.

4.1 Motivation and Related Work

To implement a distributed detection architecture that enables routers to cooperatively defend against shrew attacks, a reliable, low latency data exchange mechanism is mandatory. Only by monitoring the detection results of peers, routers can make more accurate decisions and catch up weak signals that may be ignored. A multicast mechanism is necessary to send the alert to multiple peers. Because the shrew attacks may be carrying out for a long time before the victim could realize, it is infeasible to make detection and defending functionality triggered by victims. The defense mechanism needs to be working without the instruction from victim. Therefore, this architecture should be deployed on routers transparently to servers and end hosts that we are intending to protect.

While the popularity of Internet keeps growing, multicasting has been an important research topic because it is an efficient approach to support multi-user applications. Beside the traditional bottom layer (IP multicasting) and high layer (application layer multicasting) solutions, overlay multicasting is a new category proposed recently that implements multicasting in the middle [LAO *et al.* 2005]. The major

concerns in multicasting protocol design are in management easiness and bandwidth utility efficiencies. In our study, as the major goal is to perform efficient inter-router communications, we consider only the overlay multi-casting technique.

Overlay multicast is supported by additional intermediate node called proxies or service nodes. They are specially deployed inside the network to construct an overlay infrastructure cooperatively and maintain multicast trees among themselves. Being considered to possess advantages of both IP multicast and ALM, a lot of interesting architecture have been proposed in recent years, like Scattercast [CHAWATHE *et al.* 2000a], RMX [CHAWATHE *et al.* 2000b], and AMCast [SHI *et al.* 2001, 2002]. All those designs aim to achieve high bandwidth utility efficiency without requiring direct router support or the presence of a physical broadcast mechanism.

In general, overlay network could be divided into three categories according to the topology of proxies' connectivity: mesh overlay such as Mithos [WALDVOGEL and RINALDI 2003], tree overlay such as oSTREAM [CUI *et al.* 2004], TAG [KWON and FAHMY 2002], OMNI [BANERJEE *et al.* 2003], Overcast [JANNOTTI *et al.* 2000], and hybrid mesh-tree overlay such as Narada [CHU *et al.* 2001]. While each of above multicast protocols has their own advantages and disadvantages, they are not suitable for our purpose since their multicast architectures are involved with transport layer protocols. Our functionality is restrained among routers and implemented in network layer (IP layer). Essentially, what we need is an application multicasting tree overlay network protocol that is transparent to end hosts and we do not expect any functionality is realized beyond network layer.

4.2 Our Proposed LocalCast Protocol

Enlightened by the work of Kademlia [MAYMOUNKOV and MAZIERES 2002] on a peer-to-peer information system, we propose a network layer multicast protocol named ***LocalCast*** that is not an IP multicast protocol in traditional sense because:

1) In the vertical direction of network layer model, our protocol sits on top of IP layer and is transparent to transport layer protocols.

2) From the angle of view of deployment, LocalCast holds its responsibility as limited in routers of a physical network. No end hosts or servers are involved.

The major tasks of LocalCast protocol include monitoring children node status, managing multicast trees, and exchanging data periodically.

A. The Node State

Every LocalCast node is a router in an AS that functions as root of a multicast tree, and they need to store information about its children such as routing, position, and status. For this purpose, each node keeps a list of tuple {*IP address, Distance, Connection Port, Age*} for nodes on its multicast tree. We call these lists s-list where "s" stands for status. The size of s-list is proportional to the maximum distance allows from root to its children. It would be idea if each node is capable of communicating with all peers in

the AS, each router would have a whole picture of the real time traffic condition of the AS. However, that may incur a huge overhead both in management and storage requirement. In our CoAlert algorithm, each node exchange detection result with its neighbors within a distance of two hops.

The item "connection port" in the tuple is designed to identify the relative position to the root among children, since beside the distance the position information is also critical for a router to interpret received information. It is not necessary for a node to record where its peers are located physically, instead that a relative position is enough. LocalCast nodes are routers in an AS, so we expect that their status is more stable than members of other multicast groups. Unlike end hosts who may leave the network as the user leave or power off, routers should be in duty unless facing some extraordinary scenario. For this reason, we set the maximum lifetime of each entry in s-list to 30 minutes.

Periodically, each router multicasts a message to let its children (in another angle of view, they are also its root) know that it is alive. On receiving a message from a child, no matters it generates or just forwards it, root refreshes the lifetime to 30 minutes. Also, root would update the s-list by sorting the entry with latest message arrival to the head of the list. If there is no message (request or reply) received from one child before the lifetime expires, when its age reaches –1 root would remove it from the s-list.

B. Procedures in Protocol

The LocalCast protocol consists of four *Remote Procedure Calls* (RPC): PING, JOIN, MULTICAST, and PULL. The PING RPC probes a node to see if it is online. When age of an entry in its s-list approaches zero, a root uses PING to check if that node is active, if there is no response after three duplicated PING and the lifetime reach –1, that node is pruned from the s-list. JOIN RPC is launched when a new router tries to set up its own multicast group, and this also enables it to join the groups rooted from its children.

When launch a JOIN RPC, router broadcast a "join" message indicating the distance (number of hops) to all its peers. On receiving it, the routers located within the distance would add this new node into its s-list, meanwhile, also response with a packets including its IP address and Distance. While receiving these responses, the initiator can figure out their position from the port where one response has been received. In this manner, the new router could construct an entry {*IP address, Distance, Connection Port, Age*} of one by one and finally establish its own s-list.

MULTICAST RPC is the fundamental function of LocalCast protocol. It enables each router share data with all its multicast group members. By selecting the maximum distance large enough, it functions as the broadcast to the whole AS. PULL RPC provides more intelligence for each node to require data from certain peer(s) if further interesting is incurred by the multicast message. Then node can initiate a unicast transmission for more data to exploit deeper. On receiving a PULL request, a node would reply with the asked data type. For example, to make our CoAlert algorithm more intelligent, on receiving an alert from a neighbor, one may ask for the exact $L(x)$ value or even the packet process sample series for further process before to trigger local shrew filtering mechanism.

C. Potential Applications

LocalCast protocol is a simple and flexible protocol that could be useful in any applications where the network layer cooperation among routers is desired. Beside our usage in the low-rate TCP-targeted DDoS attack detection and defending, it would be helpful to improve the performance of other security applications. For example, the detection and isolation of worms, distributed defending against flooding type of DDoS attacks, or the distributed intrusion detection mechanisms. Similar to the research in multicast mechanisms, researches in these security areas are also trying to design overlay networks on top of physical networks to achieve high bandwidth utility, high throughput, low latency, and low management overhead [ANDERSON et al. 2001; CHEN et al. 2004; DABEK et al. 2004; KEROMYTIS et al. 2001].

LocalCast could provide a high efficiency information exchanging mechanism in a layer closer to physical network. This is helpful to avoid the performance punish resulted from lack of aware of underlying network structure. In fact, the items defined in LocalCast s-list could be used with great flexibility. For example, the usage of "Distance" item is essentially a metrics to group the nodes. It could be defined according to certain application. For instance, each node may be assigned an unique ID, the distance could be the number of difference bits in their IDs; or each node know its coordinates in an Euclid space, then the Euclidian distance could be used.

5. Filtering of Shrew Attack Flows

Equipped with the energy distribution analysis based testing techniques similar as described in Section 3, we can effectively filter the shrew attack flows. In this section, we describe in detail our proposed filtering techniques.

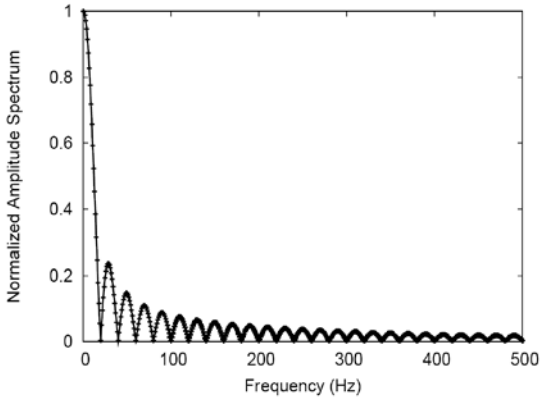
5.1 Analysis of Amplitude Spectrum Distributions

At each router, we treat the number of packet arrivals for every incoming flow as a signal series and sample it every 1 *ms*. Thus, for each flow, there is a sampled array *x(n)*. We directly convert the time series into its frequency-domain representation using DFT: Figure 8(a) shows the normalized amplitude spectrum of shrew attacks and Fig. 8(b) is the normalized amplitude spectrum of legitimate TCP flows.

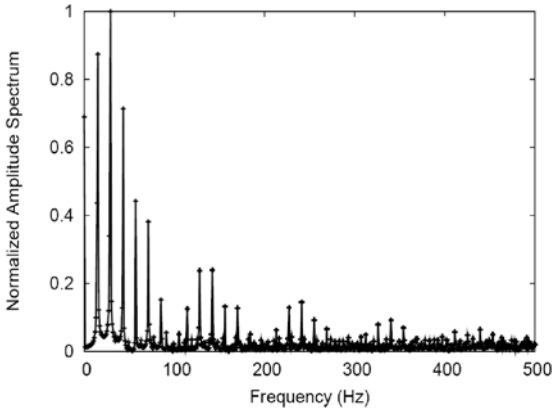
Again, Nyquist sampling theorem indicates that the highest frequency of our analysis is 500 Hz. Comparing to TCP flow, more energy of shrew pulse stream appears in lower frequency bands. This property is more profound in Fig. 8(c), which zooms into the low frequency band of [0, 50] Hz. Different from the detection of existing shrew streams, we use the amplitude spectrum over frequencies instead of PSD to evaluate the energy distribution of signal series. Essentially, amplitude spectrum represents the same matrices as PSD in evaluating the energy of signals.

Our approach adopts amplitude spectrum only because it is easier to tell shrew streams from legitimate TCP flows by showing wider distance between curves. Based on observing the normalized amplitude spectrum, we know that it is feasible to design a detection algorithm by comparing their energy

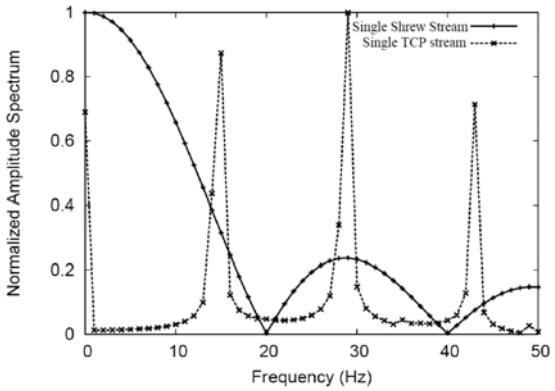
density in the low frequency band of 0 Hz to 50 Hz. The difference between the summations of amplitude in this range could be large enough to segregate shrew pulse streams from the legitimate TCP flows.



(a) Single shrew stream



(b) Single TCP flow



(c) Comparison in low frequency band

Figure 8. Normalized amplitude spectrum of the shrew pulse stream and of a TCP flow.

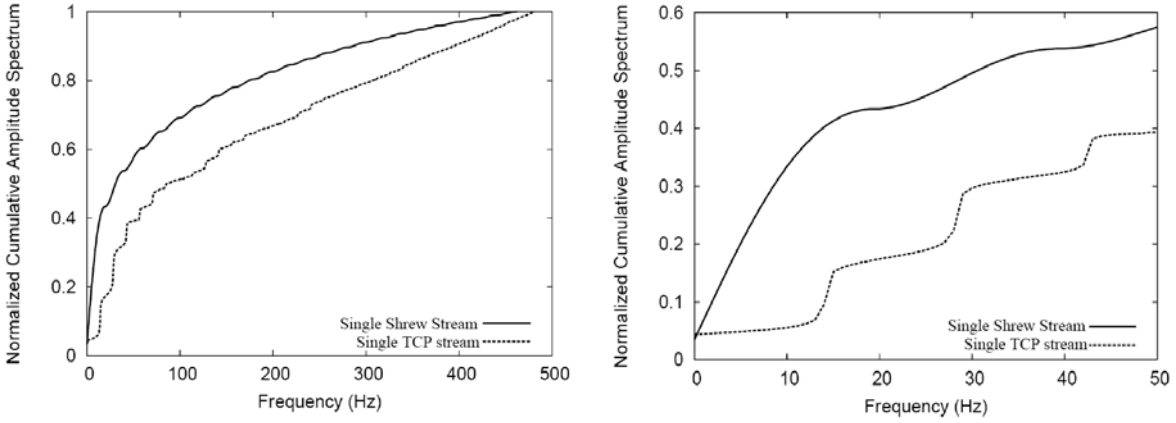
Figure 9(a) compares the *normalized cumulative amplitude spectrums* (NCAS) of TCP and shrew flows, and Fig. 9(b) zooms in low frequency band of 0Hz to 50Hz. It is around the frequency point of 20Hz that the distance of the two curves is the maximum. As shown above in Fig. 8(c), we see that the K-point is also the ending point of the first peak of amplitude spectrum curve of shrew stream. Now we need a rule to make decision on when a cumulative amplitude spectrum value has been calculated. Since there are two choices, the hypothesis test appears to be suitable in this application.

5.2 Hypothesis Detection Algorithm

Considering the fact that noise signals existing on the network and introduced in the sampling process are random, we need to confirm statistically that the variation of NCAS at the K-point are limited in such a range that allows us to distinguish shrew pulse streams from TCP flows with high confidence. Figure 10(a) presents the normalized histogram of NCAS' distribution at the K-point. Both TCP and shrew streams data are calculated in a sample space of more than 8,000 data points. The statistical results of TCP and shrew stream are given below:

$$TCP: \begin{cases} Average(\mu) = 0.1131, \\ S\tan dard\_Deviation(\sigma) = 0.026 \end{cases} \quad Shrew: \begin{cases} Average(\mu) = 0.4985 \\ S\tan dard\_Deviation(\sigma) = 0.038 \end{cases}$$

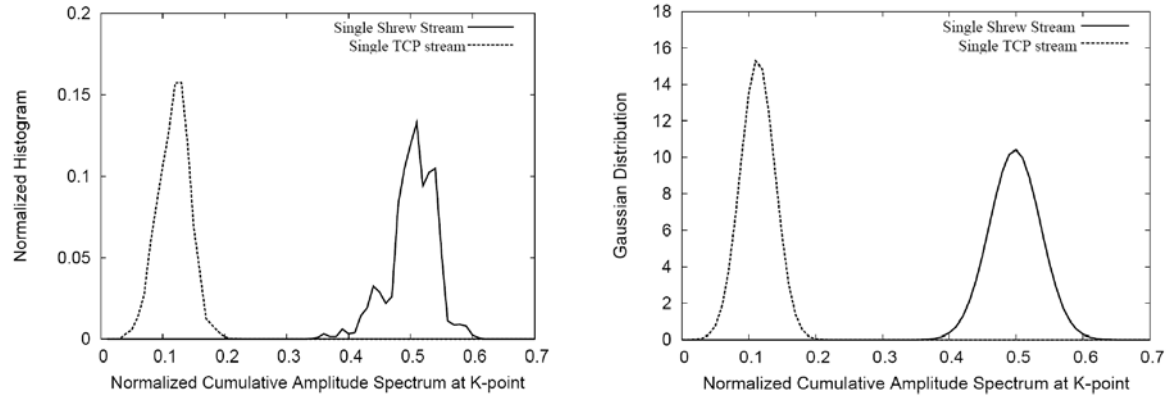
Again, according to *Central Limit Theorem* that given a distribution with a mean μ and variance σ^2, the sampling distribution approaches a *Gaussian (Normal)* distribution. Thus, we can describe distribution of NCAS at K-point using the Gaussian distribution model.



(a) On the whole frequency band **(b) On the low frequency band**

Figure 9. Normalized cumulative amplitude spectrum of a shrew attack flow compared with that of a normal TCP flow.

Figure 10(b) is the normal distribution curve of TCP flow and shrew pulse stream. In detection theory, 3σ Error Level could give us a confidence interval of 99.7% that an error level of ±3σ is good enough even in high precision detection scenarios [DEVORE and FARNUM 1999]. Table 2 below lists the confidence levels of TCP and shrew streams and their corresponding threshold settings. Figure 10(b) shows that the distance between distribution curves of TCP and shrew traffic is larger than ±3.29σ.



(a) Normalized histogram **(b) Gaussian distribution curves**

Figure 10. Normalized NCAS distribution of the shrew attack flow compared with that of a TCP flow at the K-point.

A hypothesis detection method similar to what we described in section 2 is used to distinguish shrew attack stream from TCP flows. As indicated in Table 2, simply choose the detection threshold at the K-point to be *L(0.3)*. There is no overlap of distribution curves, hence that ensures us with a confidence interval larger than 99.9%. In other words, the probability of cutting off a TCP flow as shrew attacks stream is lower than 0.1%. This shows that our hypothesis detection approach achieves pretty high accuracy and precision. The pseudo-code of our detection process is specified in Algorithm 2 below

Table 2. Confidence Levels of Gaussian Distribution

Error Level Name	Error Level	Prob. of Small Error	Prob. of Large Error	TCP Error level	Shrew Error Level
One Sigma	±σ	68%	~1:3	0.1311±0.026	0.4985±0.038
90% Error	±1.65σ	90%	1:10	0.1311±0.043	0.4985±0.046
"Two" Sigma	±1.96σ	95%	1:20	0.1311±0.051	0.4985±0.074
Three Sigma	±3σ	99.7%	1:370	0.1311±0.078	0.4985±0.114
Max. Error	±3.29σ	99.9%	1:1000	0.1311±0.086	0.4985±0.125

5.3 The Shrew-Filtering Algorithm

Based on the above hypothesis test framework, we proposed an algorithm to cut off flows with NCAS value at the K-point higher than the detection threshold. Although the source IP addresses are generally spoofed in attack packets, it is safe to use the 4-tuple {*Source IP, Source Port, Destination IP, Destination Port*} as flow labels. To minimize the storage overhead incurred by the extra lists needed to implement shrew-filtering algorithm, we store only the output of a hash function with the label as the input instead of the label itself. Our shrew-filtering algorithm handles the incoming packets according to records in the *Permanent Drop Table* (PDT), *Suspicious Flow Table* (SFT) and *Nice Flow Table* (NFT).

The flow chart of shrew filtering is shown in Fig.11. If the router has initiated the Shrew-Filtering algorithm while confirmed the alert, it start checking incoming packets. If a packet label is in the set NFT, this packet is routed normally. If it is in the set PDT, this packet is dropped. If it is in the set SFT, we continue sampling until time out. If there is no matching in any table, this packet belongs to a new flow and it would be added into the SFT, then sampling begins and timer starts. Once a timer is expired for certain flow, we convert the time-domain series into its frequency domain representation using DFT, and compare its NCAS at K-point with detection threshold. If its NCAS value is lower than the threshold, we move its record into NFT. All further incoming packets in this flow will be routed normally. If the NCAS value is higher than the threshold, we move it into the PDT and this flow will be cut off.

Algorithm 2: Shrew-Filtering for Removing Malicious Flows

01: **While** shrew filtering algorithm is triggered
02: Packet arrives, check which flow it belongs to
03: **If** this flow is on NFT **Then**
04: Do normal routing
05: **Else if** this flow is in PDT **Then**
06: Drop the packet
07: **Else if** this flow is in SFT **Then**
08: **If** sampling is not done **Then**
09: Continue sampling packets number per 1*ms*
10: **Else**
11: Convert the time-domain series into frequency domain
12: Calculate the NCAS value at K-point
13: **If** NCAS ≤ Threshold **Then**
14: Mark the flows as legitimate, move it into NFT
15: **Else**
16: Mark the shrew flow, move it to PDT, all incoming packet will be dropped
17: **Else**
18: Add this new flow into SFT, start sampling

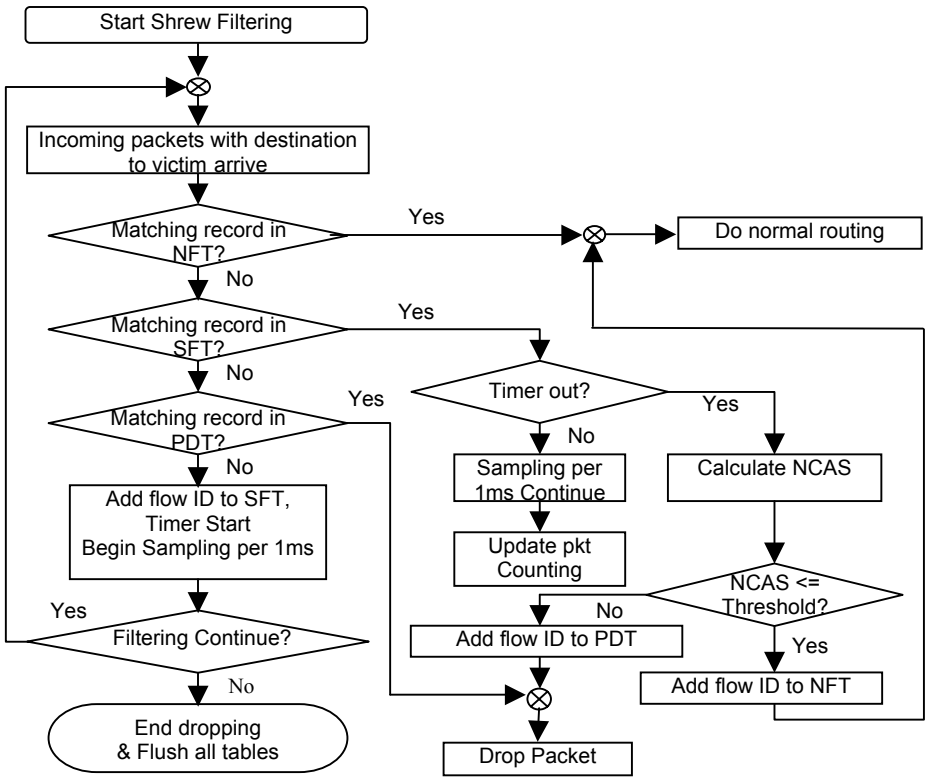


Figure 11. The Shrew-Filtering algorithm in a flow chart (NFT: *Nice Flow Table*, SFT: *Suspicious Flow Table*, PDT: *Permanent Drop Table,* NCAS: *Normalized Cumulative Amplitude Spectrum*).

6. SIMULATION EXPERIMENTS AND PERFORMANCE RESULTS

In this section, we describe our performance evaluation methodology and the results obtained using the network simulator NS-2.

6.1 NS-2 Based Simulation Experiments

We have implemented the shrew-filtering algorithm in the NS-2 simulator, which is a widely recognized packet level discrete event simulator [NS-2 2004]. The simulation have been executed on our Linux server running RedHat 9. A subclass of connector named *ShrewFilter* is added to the head of each *SimplexLink.* A *TrafficMonitor* is coded into the simulator to compute the traffic matrices. The *ShrewFilter* class is used to process the sampled array and to calculate the NCAS of flows leading to the victim. Then, the PDT or NFT entries would be set accordingly. Our NS-2 simulations are carried out with the topology shown in Fig. 12. Specifically, there are multiple potential attackers and multiple legitimate traffic sources in the simulated network. Thus, a mix of traffic flows is beamed to a single victim system.

The background traffic varies from single TCP flow and single UDP flow to multiple TCP flows and multiple UDP flows. The routers in Fig. 12 apply our LocalCast protocol to exchange traffic statistics. Specifically, when a router detects a suspicious shrew attack pattern, it initiates the LocalCast process to multicast the statistics obtained to other nearby routers. These neighboring routers will in turn perform the hypothesis testing based on the remote statistics. If suspicious activities are also observed, the propagation of updated statistics will continue in a breadth first manner.

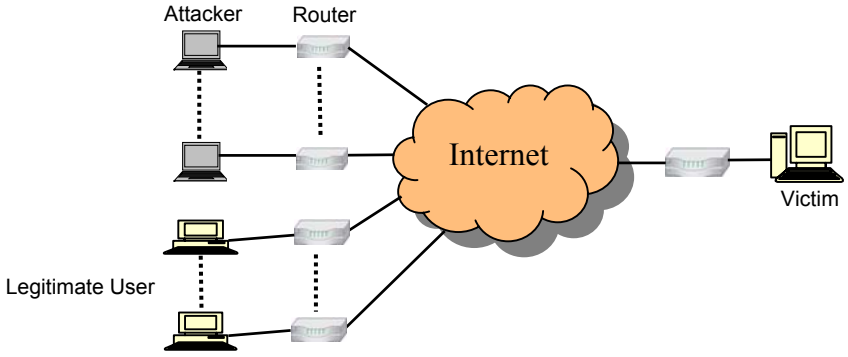


Figure 12. The simulation multi-router scenario and experimental setting with multiple attackers amid legitimate traffic streams

In our simulation experiments, the link capacity was set as 2 Mb/s and the RTTs of TCP flows are uniformly distributed from 60ms to 120ms. Similar to the low-rate attack settings by [KUZMANOVIC and KNIGHT 2003] and [SUN, et al 2004], we generate low-rate attack flows with a period T between 0.5s and 3.0s, the burst period L is in the range [30, 90] ms. The shrew attack periods are selected from 0.5s to 3.0s. For single source shrew attacks, the peak rate $R = 2$ *Mb/s* was adopted. In distributed attacks from multiple sources, the peak rate R varies between 300 Kb/s to 1 Mb/s. These traffic characteristics coincide those with shrew pulsing attacks used by previous researchers.

6.2 Performance Results and Analysis

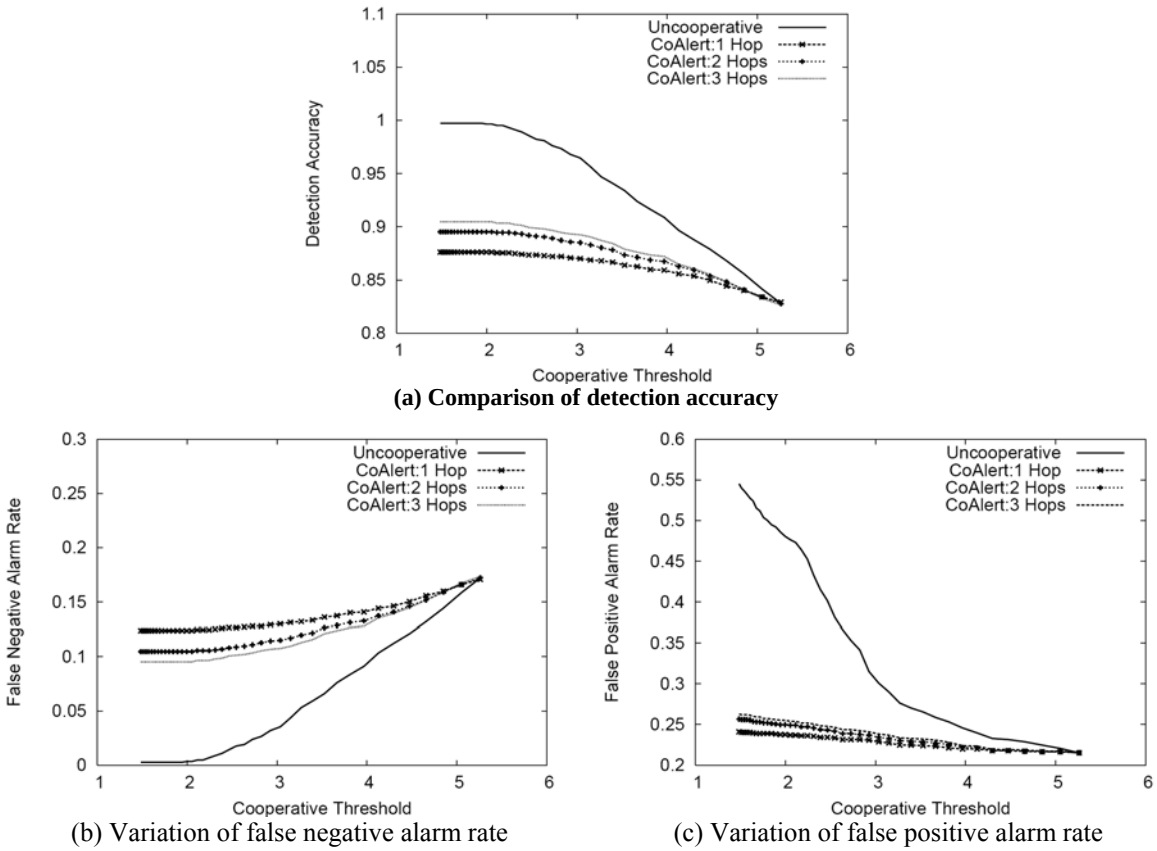
We compared the results of our shrew-filtering algorithm with the well-known *active queue management* (AQM) algorithm *Drop Tail*. We also examine the response time performance of our algorithm since it determines the duration of damage to a victim site. The notations used throughout the simulation have been presented in Table 1 in section 1.

A. Detection Accuracy

We define the *detection accuracy (α)* as the number of detected traffic flows having shrew streams embedded out of all such kind of flows as below:

$$\alpha = \frac{\text{Detected number of traffics containing shrew streams}}{\text{Total number of traffics containing shrew streams}} \quad \textbf{(7)}$$

Figure 13 presents our simulation results of detection probabilities achieved under 4 different scenarios: routers make decision individually without any cooperation; using CoAlert algorithm with cooperative distances of 1, 2 and 3 hops. Consider the individual detection scenario (presented as the solid curves in Fig. 13), where the CoAlert algorithm is not adopted.



(a) Comparison of detection accuracy

(b) Variation of false negative alarm rate

(c) Variation of false positive alarm rate

Figure 13. Comparison of the anomaly detection rates at a single victim site under 4 different scenaria: the independent anomaly detection vs. using the CoAlert scheme over multiple sites collaboratively that are 1 hop, 2 hops and 3 hops in distance.

In the case of no cooperation among the routers, the two thresholds are the same, i.e., $\gamma = \gamma_c$. We selected the threshold γ=5.26 that corresponds to a cutting point where NCPSD = 0.6, the theoretical detection accuracy of 88.0% (detection probability of P_D=0.880) is obtained in Section 2 at the cost of false negative probability P_{FN} = 0.120, and false positive probability P_{FP} = 0.167. Our simulation result shows a detection accuracy (82.6%) that is close to the theoretical value (88.0%).

To achieve even higher detection accuracy by shifting the threshold γ to a lower value, the false positive alarm rate grows quickly with the increase of detection accuracy as seen in Fig.13(c). When the CoAlert algorithm is activated, with a fixed local threshold (γ = 5.26), we can achieve higher detection accuracy while avoiding penalty of high false positive alarm rate by shifting the cooperative threshold γ_c to lower value. It is clear that higher detection accuracy is achieved at a much lower cost of false positive alarm rate.

We also studied the influence of different collaboration distances. The larger the collaboration distance, the deeper the DMTs we construct, also the further an alert is propagated. From Fig.13, we see that bigger improvement is achieved when the distance is increased from 1 to 2. However, further increase from 2 to 3 does not bring much advantage. Taking multicast overhead and link stress into account, our choice of 2-hop collaboration distance does make sense.

B. Normalized Throughput

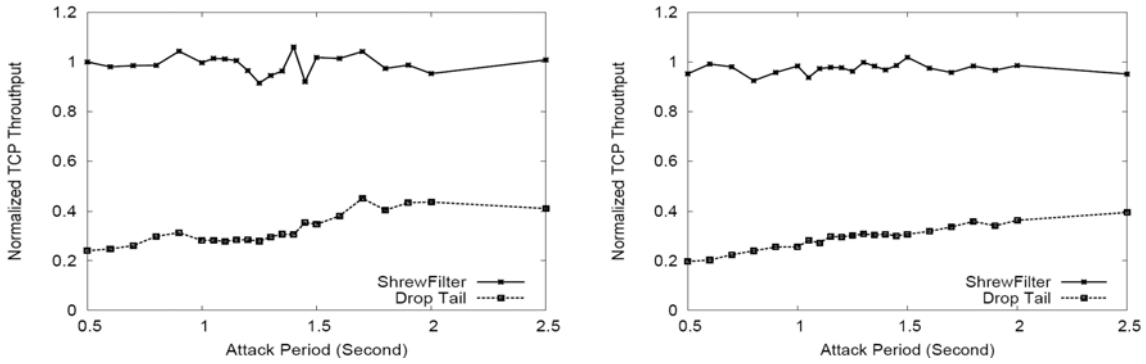
We compared the TCP throughputs achieved by our shrew-filtering algorithm triggered by detection results of the CoAlert mechanism and the Drop Tail algorithm using a *Normalized Throughput (ρ)* as the comparison metric as below:

$$\rho = \frac{\text{Average throughput achieved by the TCP flow (or aggregate) with DDoS stream}}{\text{Throughput achieved without DDoS stream}} \qquad \textbf{(8)}$$

The normalized throughput indicates the severity of the damage that the shrew streams have done to the performance of legitimate TCP flows. The lower the normalized throughput is, the greater the damage. Since all TCP variants are equally vulnerable to shrew DoS stream [KUZMANOVIC and KNIGHT 2003], we use TCP-Reno for the purpose of experiment. The shrew stream sources are injected as the topology in Fig.12. Their delay to the Internet is uniformly distributed from 30 ms to 60 ms.

Figure 14 compares the throughputs of TCP flows using the Drop Tail scheme and our shrew-filtering algorithm. The x-axis is the attack period and the y-axis is the normalized throughput TCP flow achieved. Figure 14(a) is the scenario of single TCP flow under attack of single shrew stream modeled in Fig. 1(a). Figure 14(b) is scenario of five TCP flows under attack from a single shrew stream. It is clear that under the Drop Tail, the throughput of legitimate TCP flow is far below the actual attainable throughput and the link utilization is very inefficient. With our shrew-filtering algorithm, the gain in TCP throughput is significant. It reaches what legitimate flows can reach when there is no shrew stream. This

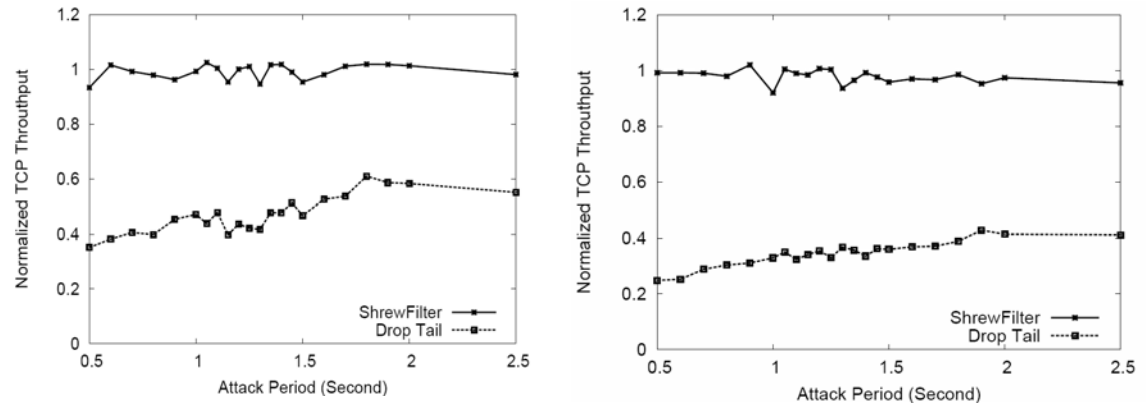
verifies that our hypothesis test model can identify shrew streams with high confidence level. We can filter out shrew streams before they hurt legitimate flows.



(a) Single TCP vs. single shrew attacks **(b) Five TCP vs. singled shrew attacks**

Figure 14. Scenarios of TCP Flows under single shrew attacks.

Distributed shrew streams are hard to be detected because of their much lower average traffic rates. Simulations are carried out using four shrew streams that distributed in either space domain (as in Fig. 1(b)) or time domain (as in Fig.1(c)), respectively. Again, we studied their effects on single and five legitimate TCP flows. Figure 15 presents the case where shrew streams are distributed in space but synchronized in Fig. 1(b). Four shrew streams are from four difference sources with the same attacking periods and the same burst lengths. However, their peak rate is only R/4. That means their average traffic rate is only 1/4 compared to the single source attack.

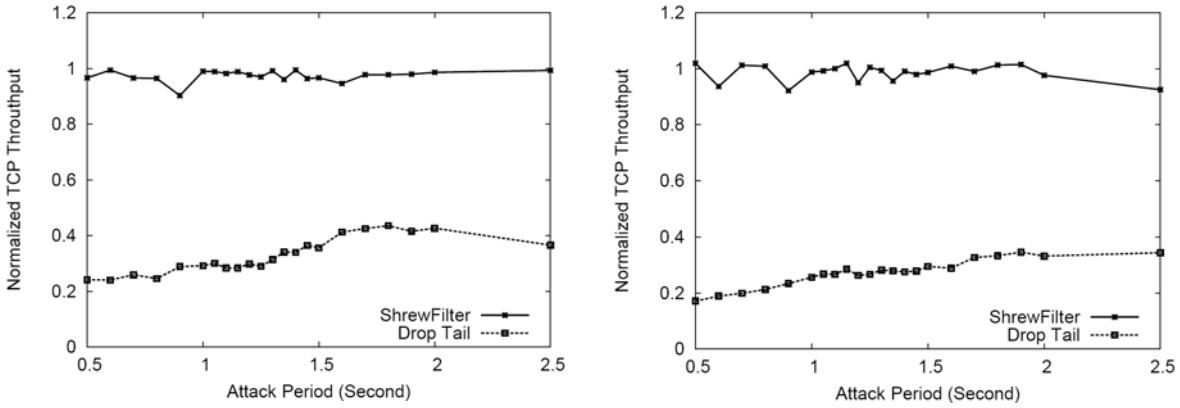


(a) Single TCP vs. 4 distributed shrew attacks **(b) Five TCP vs. 4 distributed shrew attacks**

Figure 15. The normalized TCP throughput vs. that of spatially distributed shrew attack streams

Figure 16 compares the throughputs of TCP flows under the Drop Tail algorithm and our shrew-filtering algorithm in case that shrew streams are distributed in time fashion but synchronized in Fig.1(c). Four shrew streams are from four difference sources with the same peak rates and the same burst lengths.

However, their attacking periods are 4T. This distribution makes the interval between pulses four times longer to bring down the average traffic rate to 1/4 of the single source attack pulse stream. The above results show that our shrew-filtering algorithm is indeed capable of recognizing shrew attack streams with lower average traffic rate. This is one major advantage of frequency spectrum technique over bandwidth utilization analysis. Even if the shrew streams were launched from more zombies to further lower their average bandwidth utility, their frequency spectrum would possess the same properties.



(a) Single TCP vs. 4 timely distributed shrew attacks **(b) Five TCP vs. 4 timely distributed shrew attacks**

Figure 16. The normalized TCP throughput vs. that of timely distributed shrew attack streams

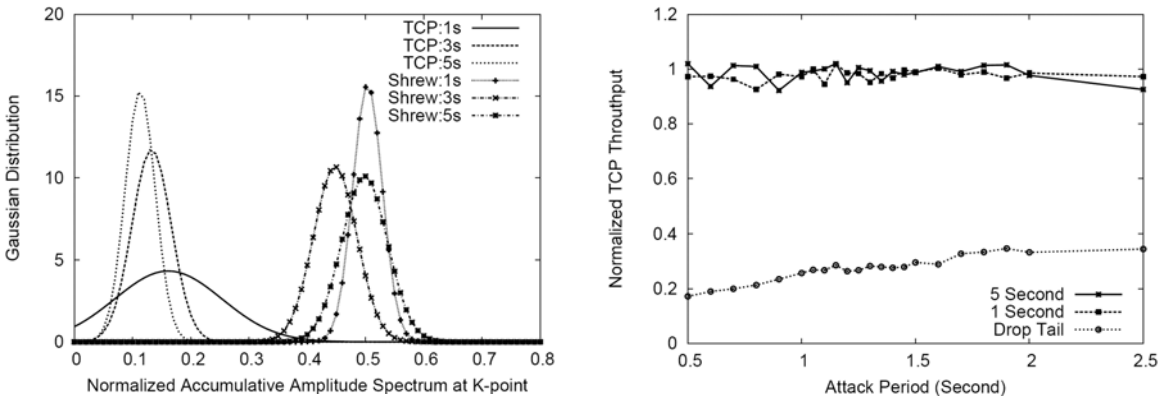
C. Response Time

The response time is a critical parameter to evaluate the performance of our shrew-filtering algorithm. In general, how long the time a DDoS defense algorithm takes to detect whether malicious flows exists or not is the time used to monitor the traffic conditions. That is varied according to the traffic load on the link. However, the load on the link does not affect the response time of our shrew-filtering algorithm. We find that the performance of the shrew-filtering algorithm is coherent under different traffics, where we used the same sampling time of 5 seconds.

The only issue that affects the response time is how long the sampled series is required to make decision precisely. The DFT considers the sequence *x[n]* with length *N* as a continuous periodic signal with a period *N* that $x[n]=x[n+rN]$ for any integer values of *n* and *r* [ALLAN and MILLS 2004]. The effects of variant sampling length are determined by the signal's periodicity. If the sampled sequence presents similar frequency characteristics of original signal, then the variance of sampling time won't impact on our detection precision.

Figure 17(a) presents the distribution of NCAS at the K-point of TCP flows and the shrew streams. They are sampled from 1 second, 3 second and 5 second. As the sampling time decreases, the NCAS at the K-point of TCP flows scatters wider. Therefore, the probability of treat a legitimate TCP flows as shrew stream increases. However, the distributions of NCAS at the K-point of shrew streams are

pretty stable. If we stick on the threshold of 0.3, the high detection confidence level is maintained even the sampling time decreases to 3 seconds.



(a) NCAS Distributions at the K-point **(b) Throughput with sample times 1, 3 and 5 seconds**

Figure 17. The effects of different sample lengths on the TCP and shrew-attack throughputs.

Table 3 shows the confidence levels of different sampling times. We observe ±1.96σ (95%), ±3σ (99.7%) and ±3.29σ (99.9%) error levels of TCP and shrew streams. When sampling time (the response time τ) is longer than 2 seconds, there is no overlap between the ±3.29σ error level ranges of TCP flow and shrew stream. Therefore, the confidence level of detecting and filtering shrew streams is very high (99.9%) while $\tau \geq 2$ seconds. With $\tau = 1$ second, we observed an overlap in both ±3σ and ±3.29σ error ranges, but no overlap for ±1.96σ error level. This implies that information carried by sampled signal series cannot separate TCP flows from shrew streams with such a high confidence level (99.7%). However, the shrew-filtering algorithm still could respond to the shrew attacks in 1 second. We cut off it with little sacrifice in confidence level (95%).

Table 3. Confidence Levels of Traffic Streams under Varying Sampling Time

Confidence Intervals	Traffic Types	1 Second	2 Second	3 Second	4 Second	5 Second
±1.96σ/95%	TCP Flow	0.1614±0.176	0.1445±0.094	0.1327±0.067	0.1258±0.078	0.1131±0.051
	Shrew	0.5036±0.050	0.4690±0.061	0.4508±0.067	0.4479±0.074	0.4985±0.074
±3σ/ 99.7%	TCP Flow	**0.1614±0.270**	0.1445±0.144	0.1327±0.102	0.1258±0.120	0.1131±0.078
	Shrew	**0.5036±0.076**	0.4690±0.093	0.4508±0.103	0.4479±0.112	0.4985±0.114
±3.29σ/ 99.9%	TCP Flow	**0.1614±0.296**	0.1445±0.158	0.1327±0.112	0.1258±0.132	0.1131±0.086
	Shrew	**0.5036±0.083**	0.4690±0.102	0.4508±0.113	0.4479±0.123	0.4985±0.125

We have simulated with sampling period of 1, 3 and 5 seconds, that is, to perform the detection using sampling series 1000, 3000 points and 5000 points since the sampling period is 1 ms. Figure 16(b) shows the throughput of five TCP flows under attack of four distributed shrew streams. Clearly, all sampling series achieved the throughput much higher than the Drop Tail algorithm. Statistically, there is no difference among them.

7. CONCLUSIONS

We have presented a new frequency-domain detection scheme for defending against periodic shrew DDoS attacks in open networks. Applying autocorrelation-based statistical signal analysis, our CoAlert algorithm achieved high accuracy in detecting shrew DDoS attacks. We use the PSD distribution on packet autocorrelation over the frequency domain. We find that shrew attacks are mostly distributed in the low frequency band in contrast to wideband distribution of legitimate traffic streams. Supported by more than 8,000 data points collected from simulation experiments, we established a hypothesis test framework to achieve high detection accuracy with low false alarms.

Equipped with the analysis of energy distribution over frequencies and hypothesis testing techniques, we investigated deeper into the flow level. We proposed a shrew-filtering algorithm to cut off low-rate TCP-targeted DDoS attack flows. Using a binary hypothesis test and a Gaussian distribution, we revealed that the new shrew-filtering algorithm achieves very high accuracy. The shrew filtering mechanism is capable of blocking malicious shrew flows with accuracy greater than 99.9%, while exhibiting a low probability (< 0.1%) in blocking legitimate TCP traffic flows.

We conducted DFT and frequency domain analysis, which are standard DSP methods that could be implemented efficiently by hardware. Therefore, our shrew-filtering algorithm would not incur much overhead in routers, since the whole detection process could be carried out by fast hardware devices. Our proposed scheme will not burden the routers from performing their normal operations. Another advantage of shrew-filtering algorithm is its ability to cut off malicious shrew flows almost totally. This can minimize the collateral damaging effect of shrew streams on legitimate packet flows.

To implement the *CoAlert* algorithm efficiently, we also proposed a network layer multicast protocol *LocalCast.* This protocol is completely deployed on edge routers. The protocol involves no efforts from the end hosts or servers. The protocol enables routers to detect suspicious patterns embedded in legitimate traffic flows. This CoAlert makes the IP protocol more robust with respect to adding more security functionality to satisfy real-life network applications. The hypothesis test framework can be extended to challenge other network attacks that present different characteristics in the frequency domain.

In our continued work, we plan to test the shrew DDoS defense scheme on the DETER testbed [DETER 2004]. More challengingly, the scheme should be extended to cope with the flooding type of DDoS attacks in frequency domain. Due to the aperiodic nature of the flooding DDoS attacks, this extension will be much more complex. If a frequency spectrum and special characteristics of Internet traffic are adequately abstracted, the frequency-domain approach will facilitate real-time flood detection by hardware at the router level. With this hardware solution, we expect lower burden on the end users in supporting network anomaly monitory and intrusion detection automatically. Through this approach, all the speed advantages of DSP technology can demonstrate their power to cope with the network security problems more efficiently in real-time.

REFERENCES

ABRY, P. and VEITCH, D. 1998. Wavelet analysis of long-range-dependent traffic. *IEEE Trans. Information Theory*, Vol. 44, No. 1, 2–15.

ABRY, P., BARANIUK, R., FLANDRIN, P., RIEDI, R., and VEITCH, D. 2002. Multiscale nature of network traffic. *IEEE Signal Processing Magazine*, Vol. 19, No. 3, 28–46.

ALLEN, R. L. and MILLS, D. W. 2004. *Signal Analysis: Time, Frequency, Scale, and Structure.* Wiley and Sons, New Jersey.

ANDERSEN, D. G., BALAKRISHNAN, H., KAASHOEK, M. F., and MORRIS, R. 2001. Resilient overlay networks. *Proc. of the 18th ACM Symposium on Operating Systems Principles (SOSP),* Banff, Canada, October 21-24.

BANERJEE, S. KOMMAREDDY, C., KAR, K., BHATTACHARJEE, B., and KHULLER, S. 2003. Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications. *IEEE INFOCOMM*, Orlando, FL., June 2-5.

BARFORD, P., KLINE, J., PLONKA, D., and RON, A. 2002. A signal analysis of network traffic anomalies. ACM *Proc. Internet Measurement Workshop.* Marseille, France, November 6-*8.*

CAI, M., HWANG, K., KWOK, Y.-K., SONG, S., and CHEN, Y. 2005. Fast Containment of Internet Worms and Tracking of DDoS Attacks with Distributed-Hashing Overlays. *IEEE Security and Privacy, November*.

CHANG, R. K. C. 2002. Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial. *IEEE Communications,* October.

CHAWATHE, Y., MCCANNE, S., and BREWER, E.A. 2000a. An Architecture for Internet Content Distributions as an Infrastructure Service. http://www.cs.berkeley.edu/yatin/papers/.

CHAWATHE, Y., MCCANNE, S., and BREWER, E.A.2000b. RMX: Reliable multicast for heterogeneous networks. *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, March 26-30.

CHEN, Y., BEACH, A., and SKICEWICZ, J. 2004. Cyber Disease Monitoring with Distributed Hash Tables: A Global Peer-to-Peer Intrusion Detection System. *Tech. Report NWU-CS-04-40*, Northwestern University, December.

CHEN, Y., KWOK, Y.-K., and HWANG, K. 2005. MAFIC: Adaptive Packet Dropping for Cutting Malicious Flows to Push Back DDoS Attacks. The *2nd International Workshop on Security in Distributed Computing Systems* (SDCS-2005), in conjunction with the *IEEE ICDCS 2005*, Columbus, OH, June 6-10.

CHENG, C.-M., KUNG, H., and TAN, K.-S., 2002. Use of spectral analysis in defense against DoS attacks. *Proc. IEEE GLOBECOM*, Taipei, China.

CHU, Y. H., RAO, S. J. G., SESHANAND, S., and ZHANG, H. 2001. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. *Proceedings of ACM SIGCOMM*, 55-67, San Diego, CA., August 27-31.

CUI, Y., LI, B., and NAHRESTEDT, K. 2004. oStream: Asynchronous Streaming Multicast in Application-Layer overlay Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1.

DABEK, F., LI, J., SIT, E., ROBERTSON, J., KAASHOWK, M. F., and MORRIS, R. 2004. Designing a DHT for low latency and high throughput. *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation* (NSDI'04), San Francisco, CA., March 29-31.

DELIO, M. 2001. New Breed of Attack Zombies Lurk.
http://www.wired.com/news/technology/0,1282,43697,00.html, May 11.

DETER and EMIST Network Project. 2004. Cyber Defense Technology Networking and Evaluation, *Communications of the ACM*, Vol.47, No.3, March

DEVORE, J. L. and FARNUM, N. R. 1999. *Applied Statistics for Engineers and Scientists*. Duxbury Press, Pacific Grove, CA.

GUIRGUIS, M., BESTAVROS, A., and MATTA, I. 2004. Bandwidth Stealing via Link Targeted RoQ Attacks. *Proc. 2nd IASTED International Conference on Communication and Computer Networks*, Cambridge, MA, November 8-10.

GUIRGUIS, M., BESTAVROS, A., MATTA, I., and ZHANG, Y. 2005. Reduction of Quality (RoQ) Attacks on Internet End Systems. *Proc. INFOCOM*, Miami, FL., March 13-17.

HE, X., PAPADOPOULOS, C., HEIDEMANN, J., and HUSSAIN, A. 2004. Spectral characteristics of saturated links. Under Submission, http://www.isi.edu/~johnh/PAPERS/He04a.html

HUANG, P., FELDMANN, A., and WILLINGER, W. 2001. A non-intrusive, wavelet-based approach to detecting network performance problems. *Proc. ACM SIGCOMM Internet Measurement Workshop.* San Francisco, CA., November 1-2.

HUSSAIN, A., HEIDEMANN, J., and PAPADOPOULOS, C. 2003. A Framework for Classifying Denial of Service Attacks. *Proc. ACM SIGCOMM.* Karlsruhe, Germany, August 25-29

HWANG, K., KWOK, Y. K., SONG, S., CAI, M., ZHOU, R., CHEN, Y., CHEN, Y., and LOU, X. 2005. GridSec: Trusted Grid Computing with Security Binding and Self-Defense against Network Worms and DDoS Attacks. *International Workshop on Grid Computing Security and Resource Management (GSRM 2005)*, in conjunction with the ICCS-2005, May 22-25.

JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., KAASHOEK, M. F., and O'TOOLE, J. W.JR. 2000. Overcast: Reliable multicasting with an overlay network. *Proc. 4th Symposium on Operating Systems Design and Implementation (OSDI'00)*, San Diego, CA., October 22-25.

KAY, S. M. 1999. *Fundamentals of Statistical Signal Processing:* Vol. II, Detection Theory. Prentice Hall, New Jersey.

KEROMYTIS, A. D., MISRA, V., and RUBENSTEIN, D. 2002. SOS: Secure Overlay Services. *ACM SIGCOMM'02*, Pittsburgh, PA. August 19-23.

KUZMANOVIC, A. and KNIGHTLY, E. W. 2003. Low-Rate TCP-Targeted Denial of Service Attacks— The Shrew vs. the Mice and Elephants. *Proc. ACM SIGCOMM 2003*, Karlsruhe, Germany, August 25-29.

KWOK, Y.-K., TRIPATHI, R., CHEN, Y., and HWANG, K., 2005. HAWK: Halting Anomaly with Weighted ChoKing to Rescue Well-Behaved TCP Sessions from Shrew DoS Attacks. *submitted to IEEE ICCNMC,* February.

KWON, M., and FAHMY, S. 2002. Topology-Aware Overlay Networks for Group Communication. *ACM NOSSDAV,* Miami Beach, FL., May 12-14.

LAN, K.-C., HUSSAIN, A., and DUTTA, D. 2003. The Effect of Malicious Traffic on the Network. *Proc. PAM 2003*, La Jolla, CA. April 6-8.

LAO, L., CUI, J.-H., GERLA, M., and MAGGIORINI, D. 2005. A Comparative Study of Multicast Protocols: Top, Bottom, or In the Middle. *Technique Report TR040054,* Computer Science Department, UCLA, January.

LUO, X. and CHANG, R. K. C. 2005. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. *Network and Distributed System Security Symposium* (NDSS'05), San Diego, CA., February 2-5.

MAHAJAN, R., FLOYD, S., and WETHERALL, D. 2001. Controlling high-bandwidth flows at the congested router. *Proc. ACM 9th International Conference on Network Protocols (ICNP),* Riverside, CA., November 11-14.

MAYMOUNKOV, P. and MAZIERES, D. 2002. Kademlia: A peer-to-peer information system based on the XOR metric. *Proc. Of the 1st IPTPS,* Cambridge, MA, March 7-8.

MOORE, D., VOELKER, G. M., and SAVAGE, S., 2001. Inferring Internet Denial-of-Service Activity. *10th USENIX Security Symposium*, Washington D.C., August 13-17.

NS-2 Network Simulator, 2004. http://www.isi.edu/nsnam/ns/.

OPPENHEIM, A. V. and SCHAFER, R. W. 1999. *Discrete-Time Signal Processing.* Pearson Education (Singapore), Delhi, India

PARTRIDGE, C., COUSINS, D., JACKSON, A., KRISHNAN, R., SAXENA, T., and STRAYER, W. T. 2002. Using Signal Processing to Analyze Wireless Data Traffic. *Proc. ACM workshop on Wireless Security*, Atlanta, GA, September 28-28.

PAXSON, V. and ALLMAN, M. 2000. Computing TCP's Retransmission Timer. *Internet RFC 2988,* November.

SHI, S., and TURNER, J. S. 2002. Routing in overlay multicast networks. *Proceedings of IEEE INFOCOM*, New York, June 23-27.

SHI, S., TURNER, J. S., and WALDVOGEL, M. 2001. Dimensioning server access bandwidth and multicast routing in overlay networks. *Proceedings of NOSSDAV'01*, New York, June 25-26.

SONG, S., HWANG, K. and KWOK, Y. K., 2005. Trusted Grid Computing with Security Binding and Trust Integration, *Journal of Grid Computing*, August.

SPECHT, S. M. and LEE, R. B. 2004. Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures. *Proc. PDCS,* San Francisco, CA., September 15-17.

SUN, H. B., LUI, J. C. S., and YAU, D. K. Y. 2004. Defending Against Low-rate TCP Attacks: Dynamic Detection and Protection. *Proc. IEEE International Conference on Network Protocols* (ICNP), Berlin, Germany, October 5-8.

WALDVOGEL, M. and RINALDI, R. 2003. Efficient Topology-Aware Overlay Network. *ACM SIGCOMM Computer Communications Review*, Vol. 33, No. 1, January.

BIOGRAPHICAL SKETCHES

Yu Chen received his B.S. from Chongqing University, China in 1994 and M.S. from University of Southern California (USC) in 2002. He is currently working as a Research Assistant at the USC Internet and Grid Computing Laboratory. He is pursuing the Ph.D. degree in Electrical Engineering at USC. His research interest includes Internet security, DDoS attack detection & defense, Internet traffic analysis and distributed security infrastructure. He can be reached at cheny@usc.edu.

Kai Hwang is a Professor of Electrical Engineering and Computer Science and Director of Internet and Grid Computing Laboratory at the University of Southern California. He received the Ph.D. from the University of California, Berkeley. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, Grid and cluster computing, and distributed computing systems. Dr. Hwang is an Editor of the *Journal of Parallel and Distributed Computing.* He is also on the editorial board of *IEEE Transactions on Parallel and Distributed Systems.* Presently, he leads the NSF-supported ITR GridSec project at USC. The GridSec group develops security-binding techniques for trusted Grid computing. Dr. Hwang can be reached at USC via Email: kaihwang@usc.edu or through the URL: http://GridSec.usc.edu/Hwang.html.

Yu-Kwong Kwok received the Ph.D. degrees in Computer Science from the Hong Kong University of Science and Technology, Hong Kong, in 1997. He is an Associate Professor in the Department of Electrical and Electronic Engineering, University of Hong Kong. His contribution to this work was carried out during his sabbatical visit of USC in 2004-05. His research interests include Grid computing, mobile computing, wireless communications, network protocols, and distributed computing algorithms. Dr. Kwok is a member of the ACM and a senior member of the IEEE Computer Society and Communications Society. He can be reached at ykwok@hku.hk.