Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds

Avrim Blum, Dawn Song, and Shobha Venkataraman

Carnegie Mellon University, Pittsburgh, PA 15213.
{avrim, shobha}@cs.cmu.edu, dawnsong@cmu.edu

Abstract. Intruders on the Internet often prefer to launch network intrusions indirectly, i.e., using a chain of hosts on the Internet as relay machines using protocols such as Telnet or SSH. This type of attack is called a *stepping-stone attack*. In this paper, we propose and analyze algorithms for stepping-stone detection using ideas from Computational Learning Theory and the analysis of random walks. Our results are the first to achieve provable (polynomial) upper bounds on the number of packets needed to confidently detect and identify encrypted stepping-stone streams with proven guarantees on the probability of falsely accusing non-attacking pairs. Moreover, our methods and analysis rely on mild assumptions, especially in comparison to previous work. We also examine the consequences when the attacker inserts chaff into the stepping-stone traffic, and give bounds on the amount of chaff that an attacker would have to send to evade detection. Our results are based on a new approach which can detect correlation of streams at a fine-grained level. Our approach may also apply to more generalized traffic analysis domains, such as anonymous communication.

Key words: Network intrusion detection. Evasion. Stepping stones. Interactive sessions. Random walks.

1 Introduction

Intruders on the Internet often launch network intrusions indirectly, in order to decrease their chances of being discovered. One of the most common methods used to evade surveillance is the construction of *stepping stones*. In a stepping-stone attack, an attacker uses a sequence of hosts on the Internet as relay machines and constructs a chain of interactive connections using protocols such as Telnet or SSH. The attacker types commands on his local machine and then the commands are relayed via the chain of "stepping stones" until they finally reach the victim. Because the final victim only sees traffic from the last hop of the chain of the stepping stones, it is difficult for the victim to learn any information about the true origin of the attack. The chaotic nature and sheer volume of the traffic on the Internet makes such attacks extremely difficult to record or trace back.

To combat stepping-stone attacks, the approach taken by previous research (e.g., [1–4]), and the one that we adopt, is to instead ask the question "What can we detect if we monitor traffic at the routers or gateways?" That is, we examine the traffic that goes in and out of routers, and try to detect which streams, if any, are part of a stepping-stone attack. This problem is referred to as the *stepping-stone detection problem*. A *stepping-stone monitor* analyzes correlations between flows of incoming and outgoing traffic which may suggest the existence of a stepping stone. Like previous approaches, in this paper we consider the detection of *interactive* attacks: those in which the attacker sends commands through the chain of hosts to the target, waits for responses, sends new commands, and so on in an interactive session. Such traffic is characterized by streams of packets, in which packets sent on the first link appear on the next a short time later, within some *maximum tolerable delay* bound Δ. Like previous approaches, we assume traffic is encrypted, and thus the detection mechanisms cannot rely on analyzing the content of the streams. We will call a pair of streams an *attacking pair* if it is a stepping-stone pair, and we will call a pair of streams a *non-attacking pair* if it is not a stepping-stone pair.

Researchers have proposed many approaches for detecting stepping stones in encrypted traffic. (e.g., [1–3]. See more detailed related work in Section 2.) However, most previous approaches in this area are based on ad-hoc heuristics and do not give any rigorous analysis that would provide provable guarantees of the false positive rate or the false negative rate [2,3]. Donoho et al. [4] proposed a method based on wavelet transforms to detect correlations of streams, and it was the first work that performed rigorous analysis of their method. However, they do not give a bound on the number of packets that need to be observed in order to detect attacks with a given level of confidence. Moreover, their analysis requires the assumption that the packets on the attacker's stream arrive according to a Poisson or a Pareto distribution — in reality, the attacker's stream may be arbitrary. Wang and Reeves [5] proposed a watermark-based scheme which can detect correlation between streams of encrypted packets. However, they assume that the attacker's timing perturbation of packets is independent and identically distributed (*iid*), and their method breaks when the attacker perturbs traffic in other ways.

Thus, despite the volume of previous work, an important question still remains open: how can we design an efficient algorithm to detect stepping-stone attacks with (a) provable bounds on the number of packets that need to be monitored, (b) a provable guarantee on the false positive and false negative rate, and (c) few assumptions on the distributions of attacker and normal traffic?

The paper sets off to answer this question. In particular, in this paper we use ideas from Computational Learning Theory to produce a strong set of guarantees for this problem:

Objectives: We explicitly set our objective to be to distinguish attacking pairs from non-attacking pairs, given our fairly mild assumptions about each. In contrast, the work of Donoho et al. [4] detects only if a pair of streams is correlated. This is equivalent to our goal if one assumes non-attacking pairs are perfectly uncorrelated, but that is not necessarily realistic and our assumptions about non-attacking pairs will allow for substantial coarse-grained correlation among them. For example, if co-workers work and take breaks together, their typing behavior may be correlated at a coarse-grained level even though they are not part of any attack. Our models allow for this type of behavior on the part of "normal" streams, and yet we will still be able to distinguish them from true stepping-stone attacks.

Fewer assumptions: We make very mild assumptions, especially in comparison with previous work. For example, unlike the work by Donoho et al., our algorithm and analysis do not rely on the Poisson or Pareto distribution assumption on the behavior of the *attacking* streams. By modeling a non-attack stream as a sequence of Poisson processes with varying rates and over varying time periods, our analysis results can apply to almost any distribution or pattern of usage of non-attack and attack streams. This model allows for substantial high-level correlation among non-attackers.

Provable bounds: We give the first algorithm for detecting stepping-stone attacks that provides (a) provable bounds on the number of packets needed to confidently detect and identify stepping-stone streams, and (b) provable guarantees on false positive rates. Our bounds on the number of packets needed for confident detection are only quadratic in terms of certain natural parameters of the problem, which indicates the efficiency of our algorithm.

Stronger results with chaff: We also propose detection algorithms and give a hardness result when the attacker inserts "chaff" traffic in the stepping-stone streams. Our analysis shows that our detection algorithm is effective when the attacker inserts chaff that is less than a certain threshold fraction. Our hardness results indicate that when the attacker can insert chaff that is more than a certain threshold fraction, the attacker can make the attacking streams mimic two independent random processes, and thus completely evade any detection algorithm. Note that our hardness analysis will apply even when the monitor can actively manipulate the timing delay. Our results on the chaff case are also a significant advance from previous work. The work of Donoho et al. [4] assumes that the chaff traffic inserted by the attacker is a Poisson process independent from the non-chaff traffic in the attacking stream, while our results make no assumption on the distribution of the chaff traffic.

The type of guarantee we will be able to achieve is that given a confidence parameter δ, our procedure will certify a pair as attacking or non-attacking with error probability at most δ, after observing a number of packets that is only quadratic in certain natural parameters of the problem and logarithmic in $1/\delta$. Our approach is based on a connection to sample-complexity bounds in Computational Learning Theory. In that setting, one has a set or sequence of hypotheses $h_1, h_2, \ldots$, and the goal is to identify which if any of them has a low true error rate from observing performance on random examples [6–8]. The type of question addressed in that literature is how much data does one need to observe in order to ensure at most some given δ probability of failure. In our setting, to some extent packets play the role of examples and pairs of streams play the role of hypotheses, though the analogy is not perfect because it is the relationship *between* packets that provides the information we use for stepping-stone detection.

The high-level idea of our approach is that if we consider two packet streams and look at the *difference* between the number of packets sent on them, then this quantity is performing some type of random walk on the one-dimensional line. If these streams are part of a stepping-stone attack, then by the maximum-tolerable delay assumption, this quantity will never deviate too far from the origin. However, if the two streams are *not* part of an attack, then even if the streams are somewhat correlated, say because they are Poisson with rates that vary in tandem, this walk *will* begin to experience substantial deviation from the origin. There are several subtle issues: for example, our algorithm may not know in advance what an attacker's tolerable delay is. In addition, new streams may be arriving over time, so if we want to be careful not to have false-positives, we need to adjust our confidence threshold as new streams enter the system.

Outline. In the rest of the paper, we first discuss related work in Section 2, then give the problem definition in Section 3. We then describe the stepping-stone detection algorithm and confidence bounds analysis in Section 4. We consider the consequences of adding chaff in Section 5. We finally conclude in Section 6.

2 Related Work

The initial line of work in identifying interactive stepping stones focused on *content*-based techniques. The interactive stepping stone problem was first formulated and studied by Staniford and Heberlein [1]. They proposed a content-based algorithm that created thumbprints of streams and compared them, looking for extremely good matches. Another content-based approach, Sleepy Watermark Tracing, was proposed by Wang et al. [10]. These content-based approaches require that the content of the streams under consideration do not change significantly between the streams. Thus, for example, they do not apply to encrypted traffic such as SSH sessions.

Another line of work studies correlation of streams based on *connection timings*. Zhang and Paxson [2] proposed an algorithm for encrypted connection chains based on periods of activity of the connections. They observed that in stepping stones, the On-periods and Off-periods will coincide. They use this observation to detect stepping stones, by examining the number of consecutive Off-periods and the distance of the Off-periods. Yoda and Etoh [3] proposed a deviation-based algorithm to trace the connection chains of intruders. They computed deviations between a known intruder stream and all other concurrent streams on the Internet, compared the packets of streams which have small deviations from the intruder's stream, and utilize these analyses to identify a set of streams that match the intruder stream. Wang et al. [11] proposed another timing-based approach that uses the arrival and departure times of packets to correlate connections in real-time. They showed that the inter-packet timing characteristics are preserved across many router hops, and often uniquely identify the correlations between connections. These algorithms based on connection timings, however, are all vulnerable to active timing pertubation by the attacker – they will not be able to detect stepping stones when the attacker actively perturbs the timings of the packets on the stepping-stone streams.

We are aware of only two papers [4, 5] that study the problem of detecting stepping-stone attacks on encrypted streams with the assumption of a bound on the maximum delay tolerated by the attacker. In Section 1, we discuss the work of Donoho et al. [4] in relation to our paper. We note that their work does not give any bounds on the number of packets needed to detect correlation between streams, or a discussion of the false positives that may be identified by their method. Wang and Reeves [5] proposed a watermark-based scheme, which can detect correlation between streams of encrypted packets. However, they assume that the attacker's timing perturbation of packets is independent and identically distributed (*iid*). Our algorithms do not require such an assumption. Further, they need to actively manipulate the inter-packet delays in order to embed and detect their watermarks. In contrast, our algorithms require only passive monitoring of the arrival times of the packets.

Wang [12] examined the problem of determining the serial order of correlated connections in order to determine the intrusion path, when given the complete set of correlated connections.

3 Problem Definition

Our problem definition essentially mirrors that of Donoho et al. [4]. A *stream* is a sequence of packets that belong to the same connection. We assume that the attacker has a maximum delay tolerance Δ, which we may or may not know. That is, for every packet sent in the first stream, there must be a corresponding packet in the second stream between 0 and Δ time steps later. The notion of maximum delay bound was first proposed by Donoho et al. [4]. We also assume that there is a maximum number of packets that the attacker can send in a particular time interval t, which we call p_t. We note that p_Δ is unlikely to be very large, since we are considering interactive stepping-stone attacks. As in prior work, we assume that a packet on either stream maps to only one packet on the other stream (i.e., packets are not combined or broken down in any manner).

Similar to previous work, we do not pay attention to the content or the sizes of the packets, since the packets may be encrypted. We assume that the real-time traffic delay between packets is very small compared to Δ, and ignore it everywhere. We have a stepping-stone monitor that observes the streams going through the monitor, and keeps track of the total number of packets on each stream at each time of observation. We denote the total number of packets in stream i by time t as $N_i(t)$, or simply N_i if t is the current time step.

By our assumptions, for a pair of stepping-stone streams S_1, S_2, the following two conditions hold for the true packets of the streams, i.e., not including chaff packets:

1. $N_1(t) \geq N_2(t)$.
   Every packet in stream 2 comes from stream 1.

2. $N_1(t) \leq N_2(t + \Delta)$.
   All packets in stream 1 must go into stream 2 — i.e., no packets on stream 1 are lost enroute to stream 2, and all the packets on stream 1 arrive on stream 2 within time Δ.

If the attacker sends no chaff on his streams, then all the packets on a stepping stone pair will obey the above two conditions.

We will find it useful to think about the number of packets in a stream in terms of the total number of the packets observed in the union of two streams: in other words, viewing each arrival of a packet in the union of the two streams as a "time step". We will use $\overline{N}_i(w)$ for the number of packets in stream i, when there are a total of w packets in the union of the two streams.

In Section 4.1, we assume that a normal stream i is generated by a Poisson process with a constant rate λ_i. In Section 4.2, we generalize this, allowing for substantial high-level correlation between non-attacking streams. Specifically, we model a non-attacking stream as a "Poisson process with a knob", where the knob controls the rate of the process and can be adjusted arbitrarily by the user with time. That is, the stream is really generated by a sequence of Poisson processes with varying rates for varying lengths of time. Even if two non-attacking streams correlate by adjusting their knobs together — e.g., both having a high rate at certain times and low rates at others — our procedure will nonetheless (with high probability) not be fooled into falsely tagging them as an attacking pair.

The guarantees produced by our algorithm will be described by two quantities:

- a *monitoring time* M measured in terms of total number of packets that need to be observed on both streams, before deciding whether the pair of streams is an attack pair, and
- a *false-positive probability* δ, given as input to the algorithm (also called the confidence level), that describes our willingness to falsely accuse a non-attacking pair.

The guarantees we will achieve are that (a) any stepping-stone pair will be discovered after M packets, and (b) any normal pair has at most a δ chance of being falsely accused. Our algorithm will never fail to flag a true attacking pair, so long as at least M packets are observed. For instance, our first result, Theorem 1, is that if non-attacking streams are Poisson, then $M = 2(p_\Delta + 1)^2 \log \frac{1}{\delta}$ packets are sufficient to detect a stepping-stone attack with false-positive probability δ. One can also adjust the confidence level with the number of pairs of streams being monitored, to ensure at most a δ chance of *ever* falsely accusing a normal pair.

All logarithms in this paper are base 2. Table 1 summarizes the notation we use in this paper.

Table 1. Summary of notation

Δ	maximum tolerable delay bound
p_Δ	maximum number of packets that may be sent in time interval Δ.
δ	false positive probability
S_i	stream i
M	number of packets that we need to observe on the union of the two streams in the detection algorithms
$N_i(t)$	number of packets sent on stream i in time interval t.
$\overline{N}_i(w)$	number of packets sent on stream i when a total of w packets is present on the union of the pair of stream under consideration.

4 Main Results: Detection Algorithms and Confidence Bounds Analysis

In this section, we give an algorithm that will detect stepping stones with a low probability of false positives. We only consider streams that have no chaff, which means that every packet on the second stream comes from the first stream, and packets can only be delayed, not dropped. We will discuss the consequences of adding chaff in Section 5.

Our guarantees give a bound on the number of packets that need to be observed to confidently identify an attacker. These bounds have a quadratic dependence on the maximum tolerable delay Δ (or more precisely, on the number of packets p_Δ an attacker can send in that time frame), and a logarithmic dependence on $1/\delta$, where δ is the desired false-positive probability. The quadratic dependence on maximum tolerable delay comes essentially from the fact that on average it takes $\Theta(p^2)$ steps for a random walk to reach distance p from the origin. Our basic bounds assume the value of p_Δ is given to the algorithm (Theorems 1 and 2); we then show how to remove this assumption, increasing the monitoring time by only an $O(\log\log p_\Delta)$ factor (Theorem 3).

We begin in Section 4.1 by considering a simple model of normal streams — we assume that any normal stream S_i can be modeled as a Poisson process, with a fixed Poisson rate λ_i. We then generalize this model in Section 4.2. We make no additional assumptions on the attacking streams.

4.1 A Simple Poisson Model

We first describe our detection algorithm and analysis for the case that p_Δ is known, and then later show how this assumption can be removed.

The Detection Algorithm Our algorithm is simple and efficient: for a given pair of streams, the monitor watches the packet arrivals, and counts packets on both streams until the total number of packets (on both streams) reaches a certain threshold $2(p_\Delta + 1)^2$.[1] If in this time the difference in the number of packets of the two streams ever exceeds the packet bound p_Δ, we know the streams are normal; otherwise, the monitor restarts. If the difference stays bounded for a sufficiently long time ($\log \frac{1}{\delta}$ such trials of $2(p_\Delta + 1)^2$ packets), the monitor declares that the pair of streams is a stepping stone. The algorithm is shown in Fig. 1.

We note that the algorithm is memory-efficient — we only need to keep track of the number of packets seen on each stream. We also note that the algorithm does not need to know or compute the Poisson rates; it simply needs to observe the packets coming in on the streams.

```
DETECT-ATTACKS (δ, p_Δ)
    Set m = log 1/δ, n = 2(p_Δ + 1)^2.
    For m iterations
        For w = 1 to n packets observed on S_1 ∪ S_2.
            Compute d(w) = N_1(w) − N_2(w)
            If |d(w)| > p_Δ return NORMAL.
        Reset N_1 = N_2 = 0.
    return ATTACK.
```

Fig. 1. Algorithm for stepping-stone detection (without chaff) with a simple Poisson model

Analysis We first note that, by design, *our algorithm will always identify a stepping-stone pair, providing they send M packets.* We then show that the false positive rate of δ is also achieved by the algorithm. Under the assumption that normal streams may be modeled as Poisson processes, we show three analytical results in the following analysis:

1. When p_Δ is known, the monitor needs to observe no more than $M = 2(p_\Delta + 1)^2 \log \frac{1}{\delta}$ packets on the union of the two streams under consideration, to guarantee a false positive rate of δ for any given pair of streams (Theorem 1).
2. Suppose instead that we wish to achieve a δ probability of false positive over *all* pairs of streams that we examine. For instance, we may wish to achieve a false positive rate of δ over an entire day of observations, rather than over a particular number of streams. When p_Δ is known, the monitor needs to observe no more than $M = 2(p_\Delta + 1)^2 \log \frac{i(i+1)}{\delta}$ packets on the union of the ith pair of streams, to guarantee a δ chance of false positive among all pairs of streams it examines (Theorem 2).

[1] The intuition for the parameters as well as the proof of correctness is in the analysis section.

3. When p_Δ is unknown, we can achieve the above guarantees with only an $O(\log\log p_\Delta)$ factor increase in the number of additional packets that need to observe (Theorem 3).

Below, we first give some intuition and then the detailed theorem statements and analysis.

Intuition We first give some intuition behind the analysis. Consider two normal streams as Poisson processes with rates λ_1 and λ_2. We can treat the difference between two Poisson processes as a random walk, as shown in Fig. 2. Consider a sequence of packets generated in the union of the two streams. The probability that a particular packet is generated by the first stream is $\frac{\lambda_1}{\lambda_1+\lambda_2}$ (which we denote μ_1), and probability that it is generated by the second stream is $\frac{\lambda_2}{\lambda_1+\lambda_2}$ (which we call μ_2). We can define a random variable Z to be the difference between the number of packets generated by the streams. Every time a packet is sent on either S_1 or S_2, Z increases by 1 with probability μ_1, and decreases by 1 with probability μ_2. It is therefore a one-dimensional random walk. Assuming that our observation of the random walk begins at some unknown position x, we care about the expected time for Z to exit the bounded region $[x-p_\Delta, x+p_\Delta]$. Without loss of generality, we may take $x=0$. Then, if $|Z|>p_\Delta$, the delay bound has to be violated for some packet.

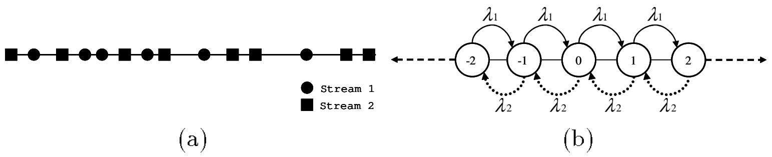

Fig. 2. (a) Packets arriving in the two streams. (b) Viewing the arrival of packets as a random walk with rates λ_1 and λ_2.

Theorem 1. *Under the assumption that normal streams behave as Poisson processes, the algorithm* DETECT-ATTACKS *will correctly detect stepping-stone attacks with a false positive probability at most δ for any given pair of streams, after monitoring $2(p_\Delta+1)^2\log\frac{1}{\delta}$ packets on the union of the two streams.*

Proof. Let $Z = N_1(w) - N_2(w)$. We first bound the probability that Z of n packets. Let T be the time taken for a one-dimensional random walk starting the origin to reach $p_\Delta+1$ or $-p_\Delta-1$ for the first time. Then, as in Feller[13], for a fair random walk,

$$E[T] = (p_\Delta+1)^2.$$

For a biased random walk starting at the origin, $E[T]$ is always strictly less than $(p_\Delta+1)^2$.

By Markov's inequality,

$$Pr[T \geq 2(p_\Delta + 1)^2] \leq \frac{1}{2}.$$

Thus, the probability that Z remains in the interval $[-p_\Delta, p_\Delta]$ throughout the arrival of n packets on the union of the streams is bounded by $\frac{1}{2}$.

To ensure that this is bounded by the given confidence level, we take m such observations of n time steps, so that $\left(\frac{1}{2}\right)^m \leq \delta$, or

$$m \geq \log \frac{1}{\delta} \ .$$

We need to observe m sets of n packets; therefore, we need $\log \frac{1}{\delta}$ intervals. □

We have just shown in Theorem 1 that our algorithm in Fig. 1 will identify any given stepping-stone pair correctly, and will have a probability δ of a false positive for any given non-attacking pair of streams. We can also modify our algorithm so that it only has a probability δ of a false positive among *all* the pairs of streams that we observe. That is, given δ, we distribute it over all the pairs of streams that we can observe, by allowing only $\frac{\delta}{i(i+1)}$ probability of false positive for the ith pair of streams, and using the fact that $\sum_{i=1}^{\infty} \frac{\delta}{i(i+1)} = \delta$. To see why this might be useful, suppose $\delta = 0.001$. Then, we would expect to falsely accuse one pair out of every 1000 pairs of (normal) streams. It could be more useful at times to be able to give a false positive rate of 0.001 over an entire month of observations, rather than give that rate over a particular number of streams.

Theorem 2. *Under the assumption that normal streams behave as Poisson processes, the algorithm* DETECT-ATTACKS *will have a probability at most δ of a false positive among all the pairs of streams it examines if, for the ith pair of streams, it uses a monitoring time of $2(p_\Delta + 1)^2 \log \frac{i(i+1)}{\delta}$ packets.*

Proof. We need to split our allowed false positives δ among all the pairs we will observe; however, since we do not know the number of pairs in advance, we do not split the δ evenly.

Instead, we allow the ith pair of streams a false positive probability of $\frac{\delta}{i(i+1)}$, and then use the previous algorithm with the updated false positive level. The result then follows from Theorem 1 and the fact that $\sum_{i=1}^{\infty} \frac{\delta}{i(i+1)} = \delta$. □

The arguments so far assume that the algorithm *knows* the quantity p_Δ. We now remove this assumption by using a "guess and double" strategy. Let $p_j = 2^j - 1$. When a pair of streams is "cleared" as not being a stepping-stone attack with respect to p_j, we then consider it with respect to p_{j+1}. By setting the error parameters appropriately, we can maintain the guarantee that any normal pair is falsely accused with probability at most δ, while guaranteeing that any attacking pair will be discovered with a monitoring time that depends only on the *actual* value of p_Δ. Thus, we can still obtain strong guarantees. In addition, even though this algorithm "never" finishes monitoring a normal pair of streams, the time between steps at which the monitor compares the difference $N_1 - N_2$ increases over the sequence. This means that for the streams that have been under consideration for a long period of time, the monitor tests differences less often, and thus does not need to do substantial work, so long as the stream counters are running continuously.

Theorem 3. *Assume that normal streams behave as Poisson processes. Then, even if p_Δ is unknown, we can use algorithm* DETECT-ATTACKS *as a subroutine and have a false positive probability at most δ, while correctly catching stepping-stone attacks within $O(p_\Delta^2(\log\log p_\Delta + \log\frac{1}{\delta}))$ packets, where p_Δ is the* actual *maximum value of $N_1(t) - N_2(t)$ for the attacker.*

Proof. As discussed above, we run DETECT-ATTACKS using a sequence of "p_Δ" values p_j, where $p_j = 2^j - 1$, incrementing j when the algorithm returns NORMAL. As in Theorem 2, we use $\frac{\delta}{j(j+1)}$ as our false-positive probability on iteration j, which guarantees having at most a δ false-positive probability overall. We now need to calculate the monitoring time. For a given attacking pair, the number of packets needed to catch it is at most:

$$\sum_{j=1}^{\lceil \log p_\Delta \rceil} 2 \cdot 2^{2j} \log \frac{j(j+1)}{\delta}.$$

Since the entries in the summation are more than doubling with j, the sum is at most twice the value of its largest term, and so the total monitoring time is $O(p_\Delta^2(\log\log p_\Delta + \log\frac{1}{\delta}))$. □

4.2 Generalizing the Poisson Model

We now relax the assumption that a normal process is Poisson with a fixed rate λ. Instead, we assume that a normal process can be modeled as a sequence of Poisson processes, with varying rates, and over varying time periods. From the point of view of our algorithm, one can view this as a Poisson process with a user-adjustable "knob" that is being controlled by an adversary to fool us into making a false accusation.

Note that this is a general model; we could use it to coarsely approximate almost any distribution, or pattern of usage. For example, at a high level, this model could approximately simulate Pareto distributions which are thought to be a good model for users' typing patterns [14], by using a Pareto distribution to choose our Poisson rates for varying time periods, which could be arbitrarily small. Correlated users can be modeled as having the same sequence of Poisson rates and time intervals: for example, co-workers may work together and take short or long breaks together.

Formally, for a given pair of streams, we will assume the first stream is a sequence given by (λ_{11}, t_{11}), (λ_{12}, t_{12}), ..., and the second stream by (λ_{21}, t_{21}), (λ_{22}, t_{22}), ... Let $N_i(t)$ denote the number of packets sent in stream i by time

t. Then, the key to the argument is that over any given time interval T, the number of packets sent by stream i is distributed according to a Poisson process with a single rate $\hat{\lambda}_{i,T}$, which is the weighted mean of the rates of all the Poisson processes during that time. That is, if time interval T contains a sequence of time intervals $j_{start}, \ldots, j_{end}$, then $\hat{\lambda}_{i,T} = \frac{1}{|T|}\sum_{j=j_{start}}^{j_{end}} \lambda_{ij} t_{ij}$ (breaking intervals if necessary to match the boundaries of T).

Theorem 4. *Assuming that normal streams behave as sequences of Poisson processes, the algorithm* DETECT-ATTACKS *will have a false positive rate of at most* δ*, if it observes at least* $\frac{7}{2}\log\frac{1}{\delta}$ *intervals of* n *packets each, where* $n = 4(p_\Delta + 1)^2$.

Proof. Let $S(t)$ be the number of packets on the union of the streams at time t. Let $D(t)$ be the difference in the number of packets at time t, i.e. $N_1(t) - N_2(t)$. Let $\hat{n} = 2(p_\Delta + 1)^2$. Let $\mathcal{E}_t$ be the event $\{\mathcal{E}_t : |D(t')| > p_\Delta \text{ for } t' \leq t\}$.

We define T to be the time when $Pr[S(T) \geq \hat{n}] = \frac{1}{2}$, and let $T' \geq T$. Then,

$$\begin{aligned}
Pr[\mathcal{E}_{T'}] &= Pr[\mathcal{E}_{T'}|S(T') \geq \hat{n}]Pr[S(T') \geq \hat{n}] \\
&\quad + Pr[\mathcal{E}_{T'}|S(T') < \hat{n}]Pr[S(T') < \hat{n}], \\
&\geq Pr[\mathcal{E}_{T'}|S(T') \geq \hat{n}]Pr[S(T') \geq \hat{n}], \\
&\geq \frac{1}{2}Pr[\mathcal{E}_{T'}|S(T') \geq \hat{n}], \\
&= \frac{1}{2}(1 - Pr[\neg\mathcal{E}_{T'}|S(T') \geq \hat{n}]).
\end{aligned}$$

From the proof of Theorem 1, for $\hat{n} = 2(p_\Delta + 1)^2$, we know

$$Pr[\neg\mathcal{E}_{T'}|S(T') \geq \hat{n}] \leq \frac{1}{2}.$$

$$\text{Therefore, } Pr[\mathcal{E}_{T'}] \geq \frac{1}{2}(1 - \frac{1}{2}) = \frac{1}{4}.$$

Now, note that $Pr[S(T) \geq k\hat{n}] \leq \frac{1}{2^k}$. Therefore, $Pr[t < T|S(t) \geq k\hat{n}] \leq \frac{1}{2^k}$. Then,

$$\begin{aligned}
Pr[\mathcal{E}_t|S(t) \geq k\hat{n}] &= Pr[\mathcal{E}_t|t \geq T]Pr[t \geq T|S(t) \geq k\hat{n}] \\
&\quad + Pr[\mathcal{E}_t|t < T]Pr[t < T|S(t) \geq k\hat{n}] \\
&\geq Pr[\mathcal{E}_t|t \geq T]Pr[t \geq T|S(t) \geq k\hat{n}] \\
&\geq \frac{1}{4}\left(1 - \frac{1}{2^k}\right).
\end{aligned}$$

Therefore, $Pr[\neg\mathcal{E}_t|S(t) \geq k\hat{n}] < 1 - \frac{1}{4}\left(1 - \frac{1}{2^k}\right)$.

To bound this by the given confidence level, we need to take m such observations of $k\hat{n}$ packets in the union of the streams, so that:

$$\left(1 - \frac{1}{4}\left(1 - \frac{1}{2^k}\right)\right)^m \leq \delta.$$

$$\text{Setting } k = 2, \ \left(1 - \frac{1}{4}\left(\frac{3}{4}\right)\right)^m \leq \delta.$$

$$m \geq \frac{\log \frac{1}{\delta}}{\log\left(\frac{16}{13}\right)}.$$

Since $\frac{1}{\log\left(\frac{16}{13}\right)} < \frac{7}{2}$, we set $m \geq \frac{7}{2} \log \frac{1}{\delta}$. □

Likewise, we have the analogues of Theorem 2 and Theorem 3 for the general model. We omit their proofs, since they are very similar to the proofs of Theorem 2 and Theorem 3.

Theorem 5. *Assuming that normal streams behave as sequences of Poisson processes, the algorithm* DETECT-ATTACKS *will have a probability at most δ of a false positive over all pairs of streams it examines, if, for the ith pair of streams, it observes $\frac{7}{2} \log \frac{i(i+1)}{\delta}$ intervals of n packets each, where $n = 4(p_\Delta + 1)^2$.*

Theorem 6. *Assuming that normal streams behave as sequences of Poisson processes, then if p_Δ is unknown, we can use repeated-doubling and incur an extra $O(\log \log p_\Delta)$ factor in the number of packets over that in Theorem 5, to achieve false-positive probability δ.*

5 Chaff: Detection and Hardness Result

All the results in Section 4 rely on the attacker streams obeying two assumptions in Section 3 — in a pair of attacker streams, every packet sent on the first stream arrives on the second stream, and any packet that arrives on the second stream arrives from the first stream. In this section, we examine the consequences of relaxing these assumptions.

Notice that only the packets that must reach the target need to obey these two assumptions. However, the attacker could insert some superfluous packets into either of the two streams, that do not need to reach the target, and therefore, do not have to obey the assumptions. Such extraneous packets are called *chaff*. By introducing chaff into the streams, the attacker would try to ensure that the number of packets observed in his two streams appear less correlated, and thus reduce the chances of being detected.

Donoho et al. [4] also examine the consequences of the addition of chaff to attack streams. They show that under the assumption that the chaff in the streams is generated by a Poisson process that is independent of the non-chaff packets in the stepping-stone streams, it is possible to detect correlation between stepping-stone pairs, as long as the streams have sufficient packets. However, an attacker may not wish to generate chaff as a Poisson process. In this section, we assume that a clever attacker will want to optimize his use of chaff, instead of adding it randomly to the streams. In Section 5.1 we explain how to detect stepping stones using our algorithm when the attacker uses a limited amount of chaff (Theorem 7). In Section 5.2 we describe how an attacker could use chaff to make a pair of stepping-stone streams mimic two independent Poisson processes, and thus ensure that the pair of streams are not correlated. We then give upper bounds on the minimum chaff the attacker needs to do this (Theorems 8 and 9).

5.1 Algorithm for Detection with Chaff

Recall that our algorithm DETECT-ATTACKS is based on the observation that, with high probability, two independent Poisson processes will differ by any fixed distance given sufficient time. An attacker can, therefore, evade detection with our algorithm by introducing a sufficient difference between the streams all the time. Specifically, our algorithm checks if the two streams have a difference that is greater than p_Δ packets every time either stream gets a packet, until there are $2(p_\Delta + 1)^2$ packets in the union of the streams. To evade our algorithm as it stands (in Fig. 1), all that the attacker might need to do is to send one packet of chaff on the faster stream.

Algorithm We now modify DETECT-ATTACKS slightly, to detect stepping-stone attacks under a limited amount of chaff. Instead of waiting for a difference of p_Δ packets between the two streams, we could wait for a difference of $2p_\Delta$ packets. The independent Poisson processes would eventually get a difference of $2p_\Delta$, but now, the attacker would need to send at least p_Δ packets in chaff in order to evade detection. He could get away with exactly p_Δ packets if he sends all of the chaff packets in the same time interval, on the same stream. However, as long as he sends less than p_Δ packets of chaff in every time interval, the monitor will flag his streams as stepping stones.[2] The complete algorithm is shown in Fig. 3.

```
DETECT-ATTACKS-CHAFF (δ, p_Δ)
    Set m = log 1/δ, n = 8(p_Δ + 1)^2.
    For m iterations
        For w = 1 to n packets observed on S_1 ∪ S_2.
            Compute d(w) = N_1(w) − N_2(w)
            If |d(w)| > 2p_Δ return NORMAL.
        Reset N_1 = N_2 = 0. NORMAL.
    return ATTACK.
```

Fig. 3. Algorithm for stepping-stone detection with fewer than p_Δ packets of chaff every $8(p_\Delta + 1)^2$ packets.

[2] We choose to wait for a difference of $2p_\Delta$ packets here, because it is the integral multiple of p_Δ that maximizes the rate at which the attacker may send chaff. We could replace it with the non-integral multiple of p_Δ that maximizes the rate at which the attacker must send chaff, but we omit the details here.

Analysis We now show that DETECT-ATTACKS-CHAFF will correctly identify stepping stones with chaff, as long as the attacker sends no more than p_Δ packets of chaff for every $8(p_\Delta+1)^2$ packets. Further, any given non-attacking pair of streams will have no more than a δ chance of being called a stepping stone.

Theorem 7. *Under the assumption that normal streams behave as Poisson processes, and the attacker sends less than p_Δ packets of chaff every $8(p_\Delta+1)^2$ packets, the algorithm* DETECT-ATTACKS-CHAFF *will have a false positive rate of utmost δ, if we observe $\log\frac{1}{\delta}$ intervals of $8(p_\Delta+1)^2$ packets each.*

Proof. The analysis is similar to that of Theorem 1.

Let $Z = N_1(w) - N_2(w)$, and let T be the time taken for a one-dimensional random walk starting the origin to reach $2p_\Delta+1$ or $-2p_\Delta-1$ for the first time. Again, as in Feller[13],

$$E[T] \leq (2p_\Delta+1)^2 \leq 4(p_\Delta+1)^2.$$

By Markov's inequality,

$$Pr[T \geq 8(p_\Delta+1)^2] \leq \frac{1}{2}.$$

Thus, the probability that Z remains in the interval $[-2p_\Delta, 2p_\Delta]$ throughout the arrival of n packets on the union of the streams is bounded by $\frac{1}{2}$.

On the other hand, for an attack pair with no chaff, we know that $\overline{N}_1(w) - \overline{N}_2(w) \leq p_\Delta$. When the attacker can add less than p_Δ packets of chaff in $8(p_\Delta+1)^2$ packets, $\overline{N}_1(w+n) - \overline{N}_2(w+n) < 2p_\Delta$, and thus, difference in packet count an attack pair cannot exceed $2p_\Delta$ in n packets. □

Note that Theorem 7 is the analogue of Theorem 1 when the chaff rate is bounded as described above. The analogues to the other theorems in Section 4 can be obtained in a similar manner.

Obviously, the attacker can evade detection by sending more than p_Δ packets of chaff for every $8(p_\Delta+1)^2$ packets. Further, if we count in pre-specified intervals, the attacker would only need to send p_Δ packets of chaff in *one* of the intervals, since the algorithm only checks if the streams differ by the specified bound in *any* of the intervals.

We could address the second problem by sampling random intervals, and checking if the difference Z in those intervals is at least $2p_\Delta$. We could also modify our algorithm to check if the difference Z stays outside $2p_\Delta$ for at least a fourth of the intervals, and analyze the resulting probabilities with Chernoff bounds. To defeat this, the attacker would have to send at least $\frac{1}{8(p_\Delta+1)}$ fraction the total packets on the union ($p_\Delta+1$ packets of chaff every $8(p_\Delta+1)^2$ packets) in an independent interval, so that every (sufficiently long) interval is unsuspicious.

However, if the attacker just chooses to send a lot of chaff packets on his stepping-stone streams, then he will be able to evade the algorithm we proposed. This type of evasion is, to some extent, inherent in the problem, not just the detection strategy we propose. In the next section, we show how an attacker could successfully mimic two independent streams, so that no algorithm could detect the attacker. We also give upper bounds on the minimum chaff the attacker needs to add to his streams, so that his attack streams are completely masked as independent processes.

5.2 Hardness Result for Detection with Chaff

If an attacker is able to send a *lot* of chaff, he can in effect ride his communication on the backs of two truly independent Poisson processes. In this section, we analyze how much chaff this would require. This gives limitations on what we could hope to detect if we do not make additional assumptions on the attacker.

Specifically, in order to simulate two independent Poisson processes exactly, the attacker could first generate two independent Poisson processes, and then send packets on his streams to match them. He needs to send chaff packets on one of the streams, when the constraints on the other stream do not allow the non-chaff packet to be forwarded to/from it. In this way, he can mimic the processes exactly, and pair of streams will not appear to be a stepping-stone pair, to any monitor watching it. Note that even if the inter-packet delays were actively manipulated by the monitor, the attacker can still mimic two independent Poisson processes, and therefore, by our definition, will be able to evade detection.

Let λ_1 be the rate of the first Poisson process, and λ_2 be the rate of the second Poisson process. In our analysis, we assume $\lambda_1 = \lambda_2 = \lambda \gg \frac{1}{\Delta}$. If $\lambda_1 \gg \lambda_2$, or $\lambda_1 \ll \lambda_2$ the attacker will need to send many more chaff packets on the faster stream, so $\lambda_1 = \lambda_2$ will be the best choice for the attacker.

We model the Poisson processes as binomials. We choose to approximate the two independent Poisson processes of rate λ as two independent binomial processes, for cleaner analysis. To generate these processes, we assume that the attacker flips two coins, each with λ bias (of getting a head), at each time step.[3] He has to send a packet (either a real packet or chaff) on a stream when its corresponding coin turns up heads, and should send nothing when the coin turn up as tails. That way, he ensures that the two streams model two independent binomial processes exactly. Since the attacker generates the independent binomial processes, he could flip coins Δ or more time steps ahead, and then decide whether a non-chaff packet can be sent across for a particular coin flip that obeys all constraints, or if it has to be chaff.

We now show how the attacker could simulate two independently-generated binomial processes with minimum chaff. First, the attacker generates two sequences of independent coin flips. The following algorithm, BOUNDED-GREEDY-MATCH, then produces a strategy that minimizes chaff for the attacker, for any pair of sequences of coin flips. Given two sequences of coin flips, the attacker matches a head in first stream at time t to the first unmatched head in the second stream in the time interval $[t, t + \Delta]$. All matched heads become real

[3] We could, equivalently, assume that the attacker flips a coin with $\frac{\lambda}{k}$ bias k times in a time step. As $k \to \infty$, the binomial approaches a Poisson process of rate λ.

(stepping-stone) packets, and all the remaining heads become chaff. An example of the operation of the algorithm is shown in Fig. 5.2.

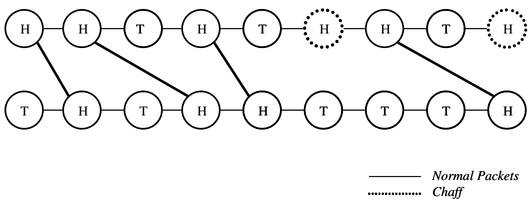

Fig. 4. An illustration of the matching produced by the algorithm BOUNDED-GREEDY-MATCH on two given sequences, with $\Delta = 2$.

The following theorem shows that BOUNDED-GREEDY-MATCH will allow the attacker to produce the minimum amount of chaff needed, when the attacker simulates two binomial processes that were generated independently.

Theorem 8. *Given any pair of sequences of coin flips generated by two independent binomial processes,* BOUNDED-GREEDY-MATCH *minimizes the chaff needed for a pair of stepping-stone streams to mimic the given pair of sequences.*

Proof. Suppose not, i.e., suppose there exists a sequence pair of coin flips σ for which BOUNDED-GREEDY-MATCH is not optimal. Let S be the strategy produced by BOUNDED-GREEDY-MATCH for σ. Let S' be a better matching strategy, so that $\mathit{Chaff}(S) > \mathit{Chaff}(S')$. Then there exists a head in σ such that h is matched with a head h' through S', but not through S.

Assume, wlog, that h is on the first stream at time t, and h' on the second stream. For S to be a valid match, h' should be in $[t, t + \Delta]$, and h' must be unmatched under S' to any other head. Let us suppose that h' is matched to another (earlier than t) head on the first stream under S (otherwise BOUNDED-GREEDY-MATCH would have generated a match between h and h' on S).

We track chain of the matching heads in the sequence backwards (starting from h) in this way: we take the currently matched head in one strategy, and look for the head that matches it in the other strategy. When this chain of matchings stops, we must have an unmatched head, and one of following two cases (the manner in which we trace the chain of matching heads, along with the assumption that the unmatched head h is on the first stream, implies that we find only matched heads on the second stream of S, and the first stream of S'):

- *Case 1*: The unmatched head is in stream 1 of S'. In this case, an unmatched head in S correlates with an unmatched head in S', and therefore, this particular case is not our counterexample, since each unmatched head under S will correspond to an unmatched head under S'.

– *Case 2*: The unmatched head is in stream 2 of S. In this case, we have to have reached this head (call it g_0) from its matching head g_1 in S'; we have to reach g_1 from matched head g_2 in S. Since we are tracing backwards in time, time of g_2 is greater than the time of g_0. However, since g_0 can be matched to g_1, we have a contradiction, since we are not matching the head g_1 to the earliest available head g_0, as per Bounded-Greedy-Match.

The analysis when h is on the second stream of S is similar.

Thus, with the algorithm Bounded-Greedy-Match, every unmatched head in S must have a corresponding unmatched head in S', therefore, $\mathit{Chaff}(S) \leq \mathit{Chaff}(S')$, creating a contradiction. □

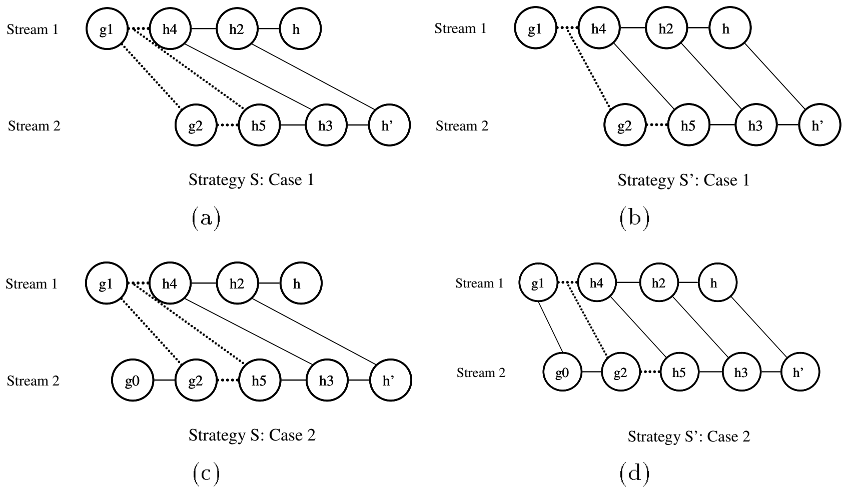

Fig. 5. The proof of Theorem 8. All the figures give an illustration of how the heads are traced back. (a) and (b) show case 1 of the proof, and (c) and (d) show case 2 of the proof. By assumption, h is unmatched in S and matched in S'. h is matched to h' in the strategy S'; in S, h' is matched to $h2$; then, we look at $h2$'s match in S', call it $h3$; use $h3$ to find $h4$ in S, $h4$ to find $h5$ in S', and so on. We continue tracing the matches of heads backwards in this manner until we stop, reaching either case 1 or case 2. In case 1, $g1$ is unmatched in strategy S', and in S, $g0$ is unmatched in S, but $g1$ is not matched greedily.

Now we examine upper bounds on the chaff that will need to be sent by the attacker, in terms of the total packets sent. We give an upper bound on the amount of chaff that the attacker must send in Bounded-Greedy-Match. We note that our analysis shows how the attacker could do this if he mimics two independent Poisson processes, but it may not be necessary for him to do this in order to evade detection.

Theorem 9. *If the attacker ensures that his stepping-stone streams mimic two truly independent Poisson processes, then, under* BOUNDED-GREEDY-MATCH, *the attacker will not need to send more than* $\frac{1}{\sqrt{2\lambda\Delta - 2\sqrt{2\lambda(1-2\lambda)\Delta}}} + 0.05$ *fraction of packets as chaff in expectation, when the Poisson rates of the streams are equal with rate* λ.

Proof. We divide the total time (coin flips) into intervals that are Δ long, and examine the expected difference in one of these intervals. Notice that for the packets that are within a specific Δ interval, matches are not dependent on the times when they were generated. (i.e., any pair of packets in this interval is no further than Δ apart in time, and therefore, could be made a valid match). Many more packets than this can be matched, across the interval boundaries, but this gives us an easy upper bound.

Consider the packets in the union of the two streams in this interval. Each packet in this union can also be considered as though it were generated from a (different) unbiased coin, with heads as stream 1 and tails as stream 2; once again, we have a uniform random walk. Since every head can be matched to any available tail, the amount of chaff is the expected (absolute) difference in the number of heads and tails. Call this difference Z, and the packets on the union of the streams X. X is then a binomial with parameters 2λ, and Δ. Therefore, $E[X] = 2\lambda\Delta$. The expectation of $\frac{Z}{X}$ is then the following:

$$\begin{aligned} E[\frac{Z}{X}] &= \sum_x \frac{1}{x} E[Z|X = x]P(X = x) \\ &= \sum_x \frac{1}{\sqrt{x}} P(X = x) \\ &\leq 0.05 + \frac{1}{\sqrt{2\lambda\Delta - 2\sigma}}, \text{ where } \sigma = \sqrt{2\lambda(1-2\lambda)\Delta}. \end{aligned}$$

Since every interval of size Δ is identical, the attacker needs to send no more than $\frac{1}{\sqrt{2\lambda\Delta - 2\sqrt{2\lambda(1-2\lambda)\Delta}}} + 0.05$ fraction as chaff in expectation. □

6 Conclusion

In this paper, we have proposed and analyzed algorithms for stepping-stone detection using techniques from Computational Learning Theory and the analysis of random walks. Our results are the first to achieve provable (polynomial) upper bounds on the number of packets needed to confidently detect and identify encrypted stepping-stone streams with proven guarantees on the probability of falsely accusing non-attacking pairs. Moreover, our methods and analysis rely on very mild assumptions, especially in comparison with previous work. We also examine the consequences when the attacker inserts chaff into the stepping-stone traffic, and give bounds on the amount of chaff that an attacker would have to send to evade detection. Our results are based on a new approach which can detect correlation of streams at a fine-grained level. Our approach may apply to more generalized traffic analysis domains, such as anonymous communication.

Acknowledgements

This work was supported in part by NSF grants CCR-0105488 and NSF-ITR CCR-0122581.

References

1. Staniford-Chen, S., Heberlein, L.T.: Holding intruders accountable on the internet. In: Proceedings of the 1995 IEEE Symposium on Security and Privacy, Oakland, CA (1995) 39–49
2. Zhang, Y., Paxson, V.: Detecting stepping stones. In: Proceedings of the 9th USENIX Security Symposium. (August 2000) 171–184
3. Yoda, K., Etoh, H.: Finding a connection chain for tracing intruders. In: F. Guppens, Y. Deswarte, D. Gollmann and M. Waidner, editors, 6th European Symposium on Research in Computer Security – ESORICS 2000 LNCS-1895, Toulouse, France (October 2000)
4. Donoho, D., Flesia, A.G., Shankar, U., Paxson, V., Coit, J., Staniford, S.: Multiscale stepping-stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In: Fifth International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science 2516, New York, Springer (2002)
5. Wang, X., Reeves, D.: Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays. In: Proceedings of the 2003 ACM Conference on Computer and Communications Security (CCS 2003), ACM Press (2003) 20–29
6. Kearns, M., Vazirani, U.: An Introduction to Computational Learning Theory. MIT Press (1994)
7. Valiant, L.: A theory of the learnable. Communications of the ACM **27** (1984) 1134–1142
8. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam's razor. Information Processing Letters **24** (1987) 377–380
9. Stoll, C.: The Cuckoo's Egg: Tracking a Spy through the Maze of Computer Espionage. Pocket Books (2000)
10. Wang, X., Reeves, D., Wu, S., Yuill, J.: Sleepy watermark tracing: An active network-based intrusion response framework. In: Proceedings of the 16th International Information Security Conference (IFIP/Sec'01). (2001) 369–384
11. Wang, X., Reeves, D., Wu, S.: Inter-packet delay-based correlation for tracing encrypted connections through stepping stones. In D.Gollmann, G.Karjoth, M.Waidner, eds.: 7th European Symposium on Research in Computer Security (ESORICS 2002), Lecture Notes in Computer Science 2502, Springer (2002) 244–263
12. Wang, X.: The loop fallacy and serialization in tracing intrusion connections through stepping stones. In: Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, ACM Press (2004) 404–411
13. Feller, W.: Probability Theory and its Applications. Volume 1. John Wiley and Sons, Inc. (1968)
14. Paxson, V., Floyd, S.: Wide-area traffic: The failure of poisson modeling. IEEE/ACM Transactions on Networking **3** (1995) 226–244