Online Sketching of Network Flows for Real-Time Stepping-Stone Detection

Baris Coskun
Electrical and Computer Engineering
Polytechnic Institute of NYU
Brooklyn, NY, USA
Email: baris@isis.poly.edu

Nasir Memon
Computer Science and Engineering
Polytechnic Institute of NYU
Brooklyn, NY, USA
Email: memon@nyu.edu

***Abstract*—We present an efficient and robust stepping-stone detection scheme based on succinct packet-timing sketches of network flows. The proposed scheme employs an online algorithm to continuously maintain short sketches of flows from a stream of captured packets at the network boundary. These sketches are then used to identify pairs of network flows with similar packet-timing characteristics, which indicates potential stepping-stones. Succinct flow sketches enable the proposed scheme to compare a given pair of flows in constant time. In addition, flow sketches identify pairs of correlated flows from a given list of flows in sub-quadratic time, thereby allowing a more scalable solution as compared to known schemes. Finally, the proposed scheme is resistant to random delays and chaff, which are often employed by attackers to evade detection. To explore its efficacy, we mathematically analyze the robustness properties of the proposed flow sketch. We also experimentally measure the detection performance of the proposed scheme.**

***Keywords*-Network Security, Stepping-Stones, Streaming Algorithms, Data Sketching**

I. Introduction

We consider the real-time network-based stepping-stone detection problem. Our approach is based on data sketches, which are widely used in the context of streaming algorithms [1][2][3]. In general, these methods maintain short sketches of data streams, which are used to efficiently answer various queries about the data stream, such as rangesum, heavy hitters, quantiles etc. Adopting a similar philosophy, we propose a novel stepping-stone detection scheme based on an online algorithm, which continuously maintains sketches of network flows from a stream of captured packets at the border of a network. Using these sketches, the proposed scheme identifies stepping-stones faster than the existing methods, without compromising robustness to timing perturbations, such as jitter and chaff.

Stepping-stones are one of the effective strategies adopted by network perpetrators to maintain anonymity of an attack. In a stepping-stone strategy, instead of direct communication, an attacker uses a series of intermediate nodes, called stepping-stones, to relay her commands to a victim. Consequently, if the victim detects that he is under attack, he will only know that the attack packets are coming from the closest intermediate node.

An intermediate node of a stepping-stone chain essentially relays information from one of its ingress flows to one of its egress flows. In general, it is possible to observe a certain correlation between relaying ingress-egress flow pairs, such as identical payload or similar packet timings. Therefore, one can detect stepping-stones in a network by searching for such correlations between ingress and egress flows at the network boundary. This kind of stepping-stone detection at network borders can be utilized in two major ways:

- If a pair of flows is detected to be a part of a stepping-stone chain, they can be blocked immediately to stop the attack, thereby preventing further harm.
- If the records of correlated flow pairs are collected from different networks, one can compile them to potentially traceback stepping-stone paths and identify the source of an attack. To give an example, "Cooperative Intrusion Traceback and Response Architecture (CITRA)" [4] is a framework that potentially benefits from a flow correlation based stepping-stone detection running at network borders. CITRA enables firewalls, routers, intrusion detection systems etc. from different networks to collaborate and exchange information using the "Intrusion Detection and Isolation Protocol (IDIP)" [5]. It's objective is to trace intrusions across networks boundaries, as close as possible to the true origin, and to automatically generate immediate responses in order to prevent intrusions from causing any further damage. Hence, incorporating a flow correlation based stepping-stone detection in CITRA potentially enables both prevention and attribution of stepping-stone attacks.

A network-based stepping-stone detection scheme has to possess two key properties in order to be reliably employed:

i) Efficiency/Scalability: To detect and block ongoing attacks, a stepping-stone detection scheme should be able to identify correlated flows in real time. For this purpose, it has to process a dense packet stream, composed of numerous concurrent ingress and egress flows, very efficiently both in terms of computation and memory.

ii) Robustness: A stepping-stone detection scheme should be resistant to network imperfections (jitter, packet drops) and to various evasion techniques (chaff, random packet delays) often employed by attackers. In general, such perturbations disrupt the correlation between flows of a stepping-stone chain, thereby potentially preventing a stepping-stone detection scheme from detecting underlying timing correlations.

There are several stepping-stone detection schemes proposed in the literature. In earlier, [6][7][8][7] authors proposed several flow-correlation algorithms which can quickly identify correlated flows based on simple timing features, such as packet counts, inter-packet time difference etc. However, they provide very limited or no resistance to some of the aforementioned timing perturbations, especially packet drops/retransmissions and chaff. On the other hand, the schemes which are designed to resist chaff [9][10] are relatively slower. That is, in order to decide if given a pair of flows are correlated, they essentially try to find a matching packet on one flow for each packet on the other one. However, comparing a pair of flows in linear time in the number of packets and doing it for every pair of ingress-egress flows does not scale to moderate to large networks. Another potential scalability issue is that existing schemes are not designed to work directly on packet streams. Basically they need to continuously reconstruct flows from a packet stream and frequently compare each of the active m ingress flows with each of the active n egress flows in $O(nm)$ time, which potentially raises several memory and computation issues.

In this work, we aim to design an efficient stepping-stone detection scheme without significantly compromising resistance against timing perturbations. For this purpose, we propose a stepping-stone detection scheme based on flow packet-timing sketches. A packet-timing sketch of a flow is a short, constant-length integer array, which summarizes the flow's packet-timing information. The proposed stepping-stone detection scheme continuously maintains succinct, constant-length sketches of active flows' packet-timing information from a stream of captured packets at a network border. These sketches are then used to efficiently identify correlated flows. The proposed flow sketches are maintained very efficiently by a streaming algorithm. The algorithm performs a few arithmetic operations for each packet, thereby allowing simultaneous sketching of thousands of concurrent active flows from a packet stream. In addition, the sketches of a pair of correlated flows remain similar, even if the flows encounter various timing perturbations. Hence, the proposed scheme is able to detect the correlation between flows of a stepping-stone chain under the presence of random delays and chaff packets.

To demonstrate the efficacy of the proposed scheme, in this paper we present various experimental results where we used real network traces. In addition, we also mathematically explain and analyze the proposed stepping stone detection scheme, where we make the following **contributions:**

- We prove that, given chaff rate and maximum packet delay values, the difference between sketches of a pair of correlated flows has an upper-bound in the expectation sense. This upper-bound justifies that the proposed scheme is expected to be able to identify correlated flows with high probability as long as the introduced packet delays and chaff are within acceptable limits.
- Exploiting the fact that the sketches of correlated flows are similar, we show that the proposed stepping-stone detection scheme can find correlated flows, with high probability, among m ingress and n egress flows in $O(n + \sqrt{nm})$ time.

The remaining of this paper is organized as follows: In Section II, we present preliminaries and define the problem formally. Then, in Section III, we explain the proposed stepping-stone detection scheme in greater detail. Following that, to demonstrate the efficacy of the proposed scheme, we present our experiments and results in Section IV. We present the limitations of the proposed scheme and possible solutions in Section V. Finally we present related work and conclusions in Section VI and Section VII, respectively.

II. Preliminaries And Problem Definition

Definitions:

• ***A network flow*** is defined as the collection of packets having the common five-tuples of source IP, source port, destination IP destination port and protocol (UDP or TCP).

• ***An active flow*** is defined as a flow which transmits its most recent packet not more than $T_{MaxIdle}$ seconds ago. $T_{MaxIdle}$ is the maximum idle threshold and we set $T_{MaxIdle} = 60$ seconds in this work.

• ***An ingress (or egress) flow*** with respect to a network is a flow which **comes into (or goes out of)** the network. More specifically, the destination IP of the packets of an ingress flow is within the network whereas that of egress flows is out of the network.

Stepping-Stone Attacks:

A stepping-stone attack incorporates several consecutive flows established between intermediate nodes, as illustrated below, where the attacker (A) first makes a connection to an intermediate node (Nd_1); then from Nd_1 he makes another connection to another intermediate node (Nd_2) and so on all the way to the victim.

$$A \rightarrow Nd_1 \rightarrow Nd_2 \rightarrow \ldots \rightarrow Nd_{h-1} \rightarrow Nd_h \rightarrow V$$

This chain of connections is referred as a **stepping-stone chain** and the intermediate nodes are often called **stepping-stones**. Each connection between two successive nodes on a stepping-stone chain is a separate network flow. These flows relay information from A to V through intermediate nodes. However, one cannot observe this directly from their payloads, as each flow is assumed to be encrypted with a different key in most cases, such as an SSH tunnel based stepping-stone chain. The response of V to A's packets might be relayed back to A through a series of flows with reverse directions, such as "$A \leftarrow Nd_1 \ldots \leftarrow Nd_h \leftarrow V$", or might be relayed through a completely different path.

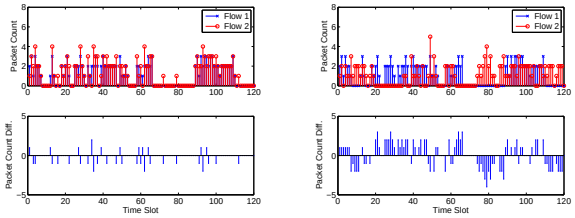
Throughout this work we assume the attacker is constrained by a **maximum tolerable delay** [11], since he needs to interactively communicate with the victim. In other

words, packets cannot be delayed more that the maximum tolerable delay along a stepping stone chain. As a result, the flows on a stepping stone chain have similar packet timing characteristics, which allows timing based stepping-stone detection systems to identify stepping-stones in a network.

Packet Delays, Jitter, Retransmissions and Chaff: There are various factors that potentially perturb the similar packet-timing characteristics among the flows of a stepping-stone chain, thereby making their discovery difficult. For instance, networks might introduce jitter on packet timings or packets might be delayed at intermediate nodes since they might need to first process each packet (i.e. re-encryption) before relaying. In addition, some packets might be dropped in the network and retransmitted. On the other hand, attackers might introduce intentional random packet delays to disrupt the correlation between the flows of a stepping-stone chain in the hope of evading detection. Attackers might also introduce superfluous packets, called **chaff**, which contain no valuable information and are not relayed to the succeeding flow on the chain. A packet-timing based stepping-stone detection scheme should be resistant to such perturbations.

Real-Time Stepping-Stone Detection Problem Definition: A stepping-stone detection system, monitoring network traffic at the network border, observes numerous active ingress and egress flows at any given time. Let I_i^t and E_j^t denote these active ingress and egress flows at time t, respectively, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$. If there's an active stepping-stone attack passing through the network, there should be at least one pair of ingress/egress flows, which have similar timing characteristics. More formally, there should be at least one flow pair, $\{I_i^t, E_j^t\}$, such that $Diff(I_i^t, E_j^t) < \mathcal{T}$, where $Diff(.,.)$ is some packet-timing difference measure based on the packets observed so far and $\mathcal{T}$ is a threshold value. We refer the flows of a such ingress/egress flow pair having similar packet-timings as **correlated flows**.

Notice that, I_i^t and E_j^t don't necessarily have a common endpoint, since two successive intermediate nodes might be in the same network and hence the flow between them is not observable at the network border. Hence, one needs to consider that every pair of ingress/egress flows is a potential correlated flow pair, thereby forcing an $O(nm)$ time search. As a result, the stepping-stone detection problem at time t can be defined as: *given the flows I_i^t and E_j^t, find all ingress/egress flow pairs $\{I_i^t, E_j^t\}$, such that $Diff(I_i^t, E_j^t) < \mathcal{T}$*. We refer to the procedure of solving this problem as **correlated flow search**. Finally, since the network has to be continuously monitored, the correlated flow search has to be periodically repeated at times $t, t + \Delta, t + 2\Delta, t + 3\Delta, ...etc$, where Δ should be selected small enough such that, a stepping-stone attack shouldn't be able to start and finish between two search procedures and evade detection.



(a) Correlated Flows (b) Uncorrelated Flows

Figure 1. Packet-count vectors of a pair of correlated flows (a) and a pair of uncorrelated flows (b). The upper figures plot packet-count vectors separately whereas the lower figures are the difference between packet-count vectors. All flows last for 60 seconds and the time-slot length is $L_{TS} = 500\ ms$.

Solving the real-time stepping-stone detection problem, especially for large networks, is a challenging task as the solution has to be:

- **Memory Efficient:** A stepping-stone detection scheme should consume minimal amount of memory for each flow in order to be able to process hundreds of thousands of active connections.
- **Computationally Efficient:** A stepping-stone detection scheme should complete each correlated flow search procedure within Δ, before the subsequent search begins. Therefore, it should employ a correlated flow search algorithm faster than $O(mn)$.

III. Sketch-Based Stepping Stone Detection in Real-Time

In this section, we present a real-time stepping stone detection schem. In summary, the proposed scheme employs an online algorithm to continuously maintain packet-timing sketches of active flows from a stream of packets. As a result, the scheme is **memory efficient**, since, for each flow, it keeps only a succinct sketch, which is a constant length integer array. In addition, while maintaining flow sketches, the proposed scheme periodically performs the correlated flow search, based on the Hamming Distances between binarized flow sketches, which yields a **computationally-efficient** search algorithm as presented in Section III-B.

A. Flow Packet-Timing Sketch

1) Overview and Basic Idea:

In order for the proposed scheme to be efficacious, sketches of correlated flows should be as similar to each other as possible, whereas those of uncorrelated flows should be as different from each other as possible. In addition, for efficiency, these sketches are required to be succinct.

Sketch Computation: One way to obtain a sketch with such properties is to compress packet-timing characteristics of a flow using a linear transformation. A linear transformation maps an input vector from one space to another space by projecting input vector onto a set of basis vectors. The projection values, which are called the transform coefficients, highly depend on the dynamics of the input

vector. Therefore, transform coefficients of similar input vectors will be also similar to each other. As a result, one can select a small set of linear transform coefficients for each flow as its sketch, and use it to distinguish between correlated and uncorrelated flows. Based on this strategy, in this work, we compute the sketch of a flow as follows: **i)** We first convert the packet-timing information of the flow to a standard vector representation, which we call **the packet-count vector**. Elements of a packet-count vector, which will be explained in the next subsection, are basically the number of packets that the flow transmits in consecutive time-slots. It is clear that, correlated flows have similar packet-count vectors whereas uncorrelated flows have different. An example case is illustrated in Figure 1, where the packet-count vectors of two correlated flows and the difference between them is shown in Figure 1(a). On the other hand, the difference between packet-vectors of two uncorrelated flows is shown in Figure 1(b). **ii)** Then we apply a random linear transformation, whose basis vectors are composed of random integers, to the packet-count vector and obtain a small set of coefficients as the sketch of the flow. The resulting sketch is an integer array and we refer this sketch as the **integer-array sketch** of the flow.

Efficient Search for Correlated Flows: When it comes to the correlated flow search, we use binarized sketches. The binarized (or binary) sketch of a flow is nothing but the signs of the elements of its integer-array sketch. They enable us to efficiently search for the correlated flows using the Hamming Distance. The basic idea of the proposed efficient search is that although the binary sketches of two correlated flows are not exactly the same, a short random subset of their sketch bits match exactly with high probability. Therefore, instead of comparing the sketches of every ingress flow with that of every egress flow, one can compare only the pairs whose certain bits match with each other and thus end up comparing pairs of correlated flows, with high probability. The details of the algorithm is given in Section III-B.

2) "Packet-Count Vector" Representation:

To a employ a linear transformation, we first need to represent the flow's packet-timing information as a vector. For this purpose, we consider the time axis as a series of non-overlapping consecutive time slots. Then, using these time-slots, we define the packet-count vector of a flow as the number packets that the flow transmits at each time-slot. More formally, let L_{TS} denote the length of these time-slots forming the time axis. Then time slot t is defined as the t^{th} time interval after an epoch (T_{epoch}) such that $[T_{epoch} + (t-1)L_{TS}\ ,\ T_{epoch} + (t)L_{TS}]$. Based on these time-slots, we can specify the packet-count vector of flow F as V_F, such that $V_F(t)$ is equal to the number of packets that flow F transmits during time-slot t.

Conceptually V_F is an infinite length vector ranging from $t = -\infty$ to $t = \infty$. However, it is obvious that $V_F(t)$ can get non-zero values only during the lifetime of flow F, as F transmits no packet before it begins or after it ends.

3) Random Linear Transformation and Integer-Array Sketch:

After computing the packet-count vector of a flow, we apply a random linear transformation to obtain the integer-array sketch, by projecting the packet-count vector V_F onto the k random basis vectors $B_{i=1,2,\ldots,k}$, as follows:

$$C_F(i) = \sum_{t=-\infty}^{\infty} B_i(t) V_F(t) \tag{1}$$

where the elements of random bases are random numbers drawn from a Bernoulli distribution with $p = 0.5$ such that:

$$Pr(B_i(t) = 1) = Pr(B_i(t) = -1) = \frac{1}{2} \tag{2}$$

Notice that, although the summation is from $t = \infty$ to $t = \infty$, we only need to sum over the time slots during which flow F transmitted at least one packet (i.e. when $V_F(t) \neq 0$). Therefore, the coefficients can be computed in real time as the packets arrive, as explained in Section III-A6.

4) Binarizing the Integer-Array Sketch:

While continuously maintaining the integer-array sketch of a flow, we perform correlated flow search based on binary sketches, which allow us to design faster search algorithms. For this purpose, we simply compute the binary sketch from the signs of these coefficients when needed. More specifically the i^{th} bit of the sketch of flow F is computed as:

$$S_F(i) = \begin{cases} 1\ , & \text{if } C_F(i) > 0 \\ 0\ , & \text{if } C_F(i) \leq 0 \end{cases} \tag{3}$$

5) Resistance of Binary Sketch to Timing Perturbations:

It is important for the proposed scheme that a binary sketch stays similar even though the flow encounters timing perturbations, such as packet delays and chaff. To mathematically investigate this, we consider a pair of correlated flows, namely F and F', representing two flows on the same stepping-stone chain. Since they are correlated, the Hamming Distance between the binary sketches of F and F' is required to be low, such that "$HamDist(S_F, S_{F'}) < threshold$". Or equivalently, the bit-error probability between the sketches of F and F' should be low. To represent bit-error probability between S_F and $S_{F'}$, we define the error probability of i_{th} bit location, $Pe_i^{[F,F']}$, as the probability of i^{th} bit of S_F and i^{th} bit of $S_{F'}$ being different than each other. Combining this with Equation (3) we can write:

$$\begin{aligned} Pe_i^{[F,F']} &= Pr\left[S_F(i) \neq S_{F'}(i)\right] \\ &= Pr\left[sign(C_F(i)) \neq sign(C_{F'}(i))\right] \end{aligned} \tag{4}$$

To determine this probability with respect to the difference between F and F', we denote the difference (or error) between the packet-count vectors of F and F' by $\mathcal{E}$ as, such that:

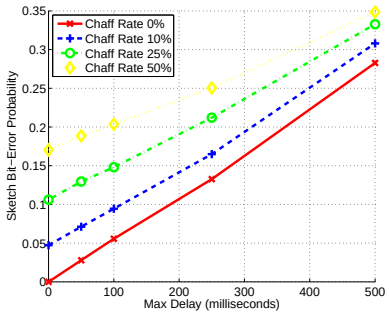


Figure 2. The upper-bound for average Bit-Error Probability of a sketch for different $maxDelay$ and $chaffRate$.

$$\mathcal{E}(t) = V_{F'}(t) - V_F(t) \tag{5}$$

Then, combining this with Equation (1), we can write i^{th} linear transform coefficient of F' as:

$$\begin{aligned} C_{F'}(i) &= \sum_{t=-\infty}^{\infty} B_i(t)\left[V_F(t) + \mathcal{E}(t)\right] \\ &= C_F(i) + \sum_{t=-\infty}^{\infty} B_i(t)\mathcal{E}(t) \end{aligned} \tag{6}$$

Recall that, a bit-error occurs only when $sign(C_F(i)) \neq sign(C_{F'}(i))$. And this only happens when $\left|\sum_{t=-\infty}^{\infty} B_i(t)\mathcal{E}(t)\right| > |C_F(i)|$ and $sign(\sum_{t=-\infty}^{\infty} B_i(t)\mathcal{E}(t)) \neq sign(C_F(i))$. Therefore, we can write:

$$Pe_i^{[F,F']} \leq Pr\left[\left|\sum_{t=-\infty}^{\infty} B_i(t)\mathcal{E}(t)\right| > |C_F(i)|\right] \tag{7}$$

Since $B_i(t)$ values are i.i.d. Bernoulli random variables with $\mu = 0$ and $\sigma^2 = 1$ as presented in Equation (2), one can derive that $\sum_{t=-\infty}^{\infty} B_i(t)\mathcal{E}(t)$ is also a random variable with $\mu = 0$ and $\sigma^2 = \sum_{t=-\infty}^{\infty} \mathcal{E}(t)^2$. Hence, from Equation (7) and Chebyshev Inequality, we can write:

$$Pe_i^{[F,F']} \leq \frac{\sum_{t=-\infty}^{\infty} \mathcal{E}(t)^2}{|C_F(i)|^2} \tag{8}$$

The above bound indicates that, as a pair of correlated flows become more different than each other, the bit-error probability between their binary sketches increases. It also shows that as the magnitude of linear transform coefficients increases, which essentially means that the flows transmit more packets, their sketches become more resistant to perturbations.

To visualize how this bound changes with respect to delay and chaff, we computed $\sum_{t=-\infty}^{\infty} \mathcal{E}(t)^2$ and $|C_F(i)|^2$ for 1000 correlated flow pairs. We used 512-bit sketches, therefore we have $i = 1, 2, ...512$. We obtained correlated flows by modifying 1000 original flows, which are 60-second long SSH flows captured at our network border, with uniformly distributed packet delays and random chaff insertions. Recall that, the bound is different for each bit position depending on the magnitude of the corresponding transform coefficient. Hence, for each correlated flow pair we computed the bit-error probability in expectation sense by using average coefficient magnitude, such that $Pe_{exp} = \frac{\sum_{t=-\infty}^{\infty} \mathcal{E}(t)^2}{\frac{1}{512}\sum_{i=1}^{512} |C_F(i)|^2}$. Figure 2 plots the average Pe_{exp} of all 1000 flow pairs for different $maxDelay$ and $chaffRate$ values. As expected, the average bit-error probability bound increases with both increasing max delay and increasing chaff rate. It is also observed that, the Hamming Distance between the sketches of a pair of correlated flows is expected to be less than a threshold as long as $maxDelay$ and $chaffRate$ don't exceed a certain value.

6) Online Computation of Packet-Timing Sketches:

In this subsection we present an efficient algorithm to simultaneously compute packet-timing sketches of all observed flows. The algorithm computes sketches continuously in a cumulative way. Therefore, the sketch of a flow with respect to the packets received so far is available at any desired time even if the flow is still active. The essence of the algorithm is to cumulatively compute Equation (1) by updating linear transform coefficients ($C_F(i)$) for each captured packet. Since $V_F(t)$ indicates the number of packets that flow F has in time-slot t, Equation (1) essentially accumulates the random basis vector values for the time-slots that the observed packets are transmitted. More formally, for each packet p of flow F, the algorithm updates transform coefficients as follows:

$$C_F(i) \leftarrow C_F(i) + B_i(t_p) \tag{9}$$

where t_p indicates the time-slot which packet p is transmitted. The pseudo-code for the algorithm performing this procedure for all flows is given below. The algorithm runs on the packet stream Φ and take three inputs namely the epoch (T_{Epoch}), length of the time-slots (L_{TS}), and the number of transform coefficients$-$*equivalently the length of binary sketches*$-(k)$ for each flow. The subroutines $getFlow()$ and $getTimeStamp()$ extracts the flow information that the packet belongs to and the time stamp from a a packet respectively. The algorithm might compute the random basis vector elements "$B_i(t)$" online whenever needed. However, since packets arrive roughly in chronological order, $B_i(t)$ values can be pre-computed for a sliding window of time and stored in a cache repository for practical purposes.

Algorithm 1 $MaintainSketches(T_{Epoch}, L_{TS}, k)$

```
for all packet P captured on stream Φ do
    F ← getFlow(P) {Determine the flow of P}
    t = ⌊(getTimeStamp(P) − T_Epoch) / L_TS⌋ {Determine current time slot}
    for i = 1 to k do
        C_F(i) ← C_F(i) + B_i(t)
    end for
end for
```

The above algorithm continuously maintains an integer-array sketch for each active flow. If the binary sketch of a flow is required at any time, the algorithm simply computes it from the signs of the current value of integer-array sketch of that flow as explained in the previous subsection.

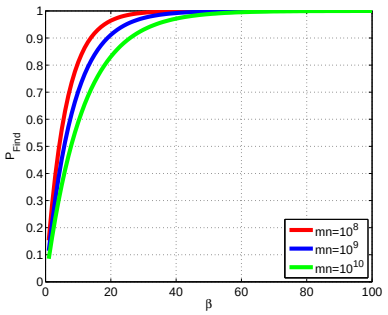


Figure 3. Probability of the proposed search algorithm finding a correlated flow-pair among m ingress and n egress flows for different number of repetitions (β). The sketch length is $k = 512$ and the threshold is $th = 71$.

The above algorithm simply updates k coefficients of the corresponding flow for each packet. Since k is constant (typically $k = 512$), the algorithm runs in linear time in the number of packets. Also, the algorithm requires linear space in the number of active flows since a constant length integer array is maintained for each active flow.

B. Efficient Search for Correlated Flows

In this subsection, we present an approximate search algorithm based on binary sketches, which finds all correlated flows in $O(n + \sqrt{nm})$ time, with high probability. The algorithm is similar to the algorithm proposed in [12], where the authors search a large audio database using robust audio hashes. The algorithm steps are given below:

• We first pick α random bit locations on binary sketches, where $2^\alpha = \sqrt{nm}$. We refer the bits on these random locations of a flow's sketch as the subsketch of that flow. More formally, let $b_1, b_2, ..., b_\alpha$ be these random bit locations, where k is the sketch length and $\alpha \leq k$. Then $[S_F(b_1), S_F(b_2), ..., S_F(b_\alpha)]$ is the subsketch of flow F.

• Then for n ingress flows, we construct a lookup table, which is used to access an ingress flow from its subsketch. Note that the size of the lookup tables is $2^\alpha = \sqrt{nm}$, since a subsketch can have 2^α possible values.

• Following that, for each egress flow F_e, we compute the Hamming Distance between the binary sketch of F_e and the binary sketches of ingress flows whose subsketch is the same as the subsketch of F_e. Notice that, we use the lookup table constructed in the previous step to find the ingress flows with the matching subsketches. If any of the Hamming Distances is smaller than the threshold th, F_e and that particular ingress flow are declared as correlated.

• Finally, in case we miss comparing some of the correlated ingress/egress flow pairs, we repeat the above steps β times, each time with a different subsketch bit locations.

Recall that n and m are the number of ingress and egress flows respectively. Hence, the proposed search algorithm computes the lookup table in $O(n)$ time. After that, it compares each of the m egress flows with on average $n/2^\alpha$ flows, since on average $1/2^{\alpha^{th}}$ of the ingress flows matches the subsketch of a given egress flow. And the algorithm repeats this β times. As a result, the proposed search algorithm runs in $O((n + mn/2^\alpha)\beta)$ time in the expectation sense. Since we set $2^\alpha = \sqrt{nm}$, and β is typically a small constant, the proposed algorithm's run time is $O(n + \sqrt{nm})$.

The proposed search algorithm can find correlated flow pairs only if their subsketches match in at least one of the β repetitions. To find the probability of that event, let P_{match} denote the probability of two flows, whose sketches are closer than th, having the same subsketch. Then we can write that, $P_{match} \geq \prod_{i=0}^{\alpha-1} \frac{k-th-i}{k-i}$, where k is the sketch length. Using this, we can write the probability of the proposed search algorithm finding an ingress/egress flow pair, the Hamming Distance between whose sketches are smaller than th, in β repetitions as $P_{find} = 1 - (1 - P_{match})^\beta \geq 1 - \left(1 - \prod_{i=0}^{\alpha-1} \frac{k-th-i}{k-i}\right)^\beta$. It is clear that P_{find} approaches to 1 as β increases. To give an example, in Figure 3, we plot P_{find}, for different mn and β, where we set $2^\alpha = \sqrt{mn}$, where the sketch length is $k = 512$ and the threshold is $th = 71$. Threshold was set to allow 10^{-4} false positive rate as will be explained in Section IV. It is observed that P_{find} climbs rapidly with β, and when $\beta \approx 50$, the proposed search algorithm is expected to find almost all flows whose sketches are closer than th.

IV. Experiments and Results

To demonstrate the efficacy of the proposed stepping-stone detection method we set up an experiment, which measures how successfully the correlated flows can be detected under a fixed false alarm probability.

Obtaining Correlated Flows: In most cases, stepping-stone attacks are carried over an interactive protocol, such as SSH. Therefore, to obtain correlated flows, we first captured 100 real SSH flows, at our network's border. We refer these flows as the **original flows**. We observed that the original flows transmitted 2.4 packets per second on average. Then for each of these original flows, we obtained a **perturbed flow**, by delaying packets and introducing chaff. Despite the delay and chaff, a perturbed flow is considered to be correlated to the flow it originated from. To obtain a perturbed flow from an original flow, first we delayed each packet of the original flow by a random amount chosen uniformly from the interval $[0, maxDelay]$. Then, we introduced chaff packets to both the original flow and the perturbed flow at random times, where the ratio of the number of introduced chaff packets to the number of original packets was determined by $chaffRate$.

Searching for Correlated Flows: Once we obtained these 100 correlated flow pairs, we blended them into the real network network trace, which was previously captured at our network's border during a typical weekday. Then, we ran the proposed stepping-stone detection technique on the blended trace as if it was a real-time network traffic. The proposed technique continuously maintained the sketches of the active

flows in the network trace as packets were captured. To deal with terminated flows, we checked whether the flows were still active once in every minute. A flow and it's sketch were erased if it was idle for more than 60 seconds. Finally, to identify the correlated flows in the trace, we performed the proposed correlated flow search at every $\Delta = 10$ seconds. We believe $\Delta = 10$ is reasonable because, it is very hard for an attacker to start and finish her attack within 10 seconds. During the search, we used 16-bit subsketches and and 50 lookup tables, such that $\alpha = 16$ and $\beta = 50$ as explained in Section III-B . With those specific α and β values, we were able to perform the search in $O(n + \sqrt{nm})$ time for up to $n = 10^5$ active ingress and $m = 10^5$ active egress flows, since $(2^{16})^2 \approx 10^{10}$. We declared a pair of flows as correlated, when we found out in any of these periodic search processes that the Hamming Distance between the binary sketches of those flows were below a threshold th. Finally, to quantify the performance of the proposed scheme, we measured the detection rate as the ratio of the number of correctly identified correlated flows to the number of all correlated flows blended into the trace.

Selecting the Threshold: A proper setting of the detection threshold th was crucial for our experiments. Like all detection methods based on thresholds, the lower the threshold the higher the false alarm rate and the higher the threshold the lower the detection rate. In our experiments, we picked the detection threshold, which yielded the fixed 10^{-4} false alarm rate. We believe that, 10^{-4} false alarm rate is acceptable in most cases. To determine a such threshold, one needs to estimate the probability distribution of the Hamming Distances between binary sketches uncorrelated flows. For this purpose, we first empirically computed the probability distribution of Hamming Distances between the binary sketches of 5000 uncorrelated SSH flow pairs captured at our network's border. During that process, we made sure that the flows were concurrent, by shifting them in time. This enabled us to produce a more realistic estimate, since a stepping-stone detection scheme searches concurrent flows for correlated pairs. Once we computed the empirical distribution, we fitted a Gaussian on the empirical distribution. Then we picked the threshold value th such that, the integral of the fitted Gaussian from $-\infty$ to th was equal to 10^{-4}, such that $\int_{-\infty}^{th} f_G(h)\, dh = 10^{-4}$. We plot the computed empirical distribution, fitted Gassian and the selected threshold value for 512-bit sketches in Figure 4, where we used 500 millisecond time-slots.

Notice that, the distribution of Hamming Distances between the sketches of uncorrelated flows should ideally be a Binomial distribution with $p = 0.5$. However, the empirical distribution deviated from the ideal as observed in Figure 4. The reason is that, even though the flows were uncorrelated, their packet-count vector representations show some coarse similarities. For instance, since we used quite long time slots (i.e. 500 milliseconds), there's a good chance that a pair

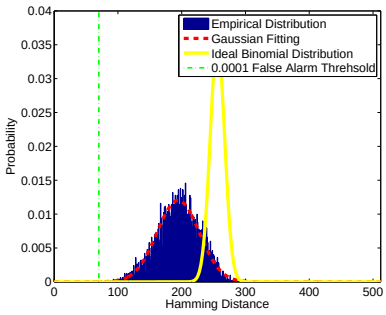


Figure 4. The empirical probability distribution of the Hamming Distances between the binary sketches of uncorrelated flows (durations between 10 and 40 seconds) and the Gaussian probability distribution function approximating the empirical distribution. The vertical line marks the threshold value (i.e. $th = 71$) for false alarm rate of 10^{-4}.

of uncorrelated flows share many common time slots that they are both active in, although they usually have different number of packets within a given time-slot.

Real-Time Exclusions in the Correlated Flow Search: In our experiments we excluded some flows in the correlated search process, since they had no potential to be a part of a stepping-stone attack. Our concern was not about the computational efficiency but rather about the detection performance, since such flows might introduce extra false positives. In order to be real-time, we identified and excluded such flows in real time as well. In our experiments, we excluded the following cases:

- ***Flows With Reverse Directions:*** In most cases, two hosts communicate with each other through a pair of flows (one for each direction), where one flow's source IP and port is other flow's destination IP and port. Although, such a pair of flows usually have similar packet timings to each other (i.e. TCP packets and their ACK's), they are not part of a stepping stone attack.
- ***Too Short Flows:*** A successful stepping-stone attack is expected to last long enough to allow the attacker exchange sufficient information with the victim over an interactive protocol. Hence, too short flows are highly unlikely to be a part of a stepping-stone attack and therefore excluded in the comparison process. To identify short flows, we used the number of time-slots in which the flow was active, such that, we excluded a flow in the search process if it was active in less than 20 time-slots.
- ***Flows With Insufficient Packet-Timing Information:*** In some cases, a flow might have no distinctive packet-timing information, such as when downloading a file or streaming a video etc. Usually such flows all have constant number of packets at each time-slot and therefore appear to be correlated with each other. Hence, such flows should be excluded in the search process and treated separately. To detect such flows in real-time, we simply checked the fraction of the time-slots in which a flow is active during its lifetime. We observed that, most of the flows having

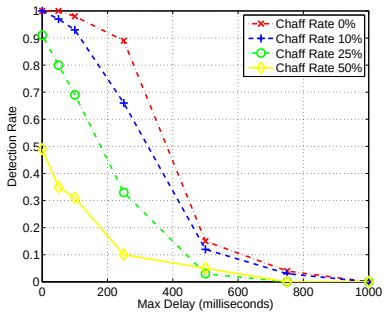


Figure 5. Detection rates for different $chaffRate$ and $maxDelay$ values. The time slot length is 500 milliseconds and the packet-timing summaries are 512 bit binary strings. As expected, increasing chaff rates and delays reduces the detection rate.

insufficient packet-timing information were active more than 90% of the time-slots.

Detection Rate Results: In our experiments, we computed the detection rate of the proposed scheme under various $maxDelay$ and $chaffRate$. Figure 5 plots these measured detection rates, where we used 500 millisecond long time-slots and 512-bit sketches. The duration of the correlated flows were 60 seconds. It is observed that the proposed scheme resists delays and chaff packets to some extent. For instance, the algorithm detected 95% of the correlated flow pairs when the $maxDelay = 100$ milliseconds and $chaffRate = 0.1$. However, further increasing chaff rates and delays decreases the detection rate, as expected since, the packet-count vectors of correlated flows deviate from each other as chaff rate and max delay increase.

Effect of Time-Slot Length: Time-slot length is an important parameter, which sets a trade-off between resistance to delays and ability to distinguish between uncorrelated flows. If the time-slot length was set too short, then even very slight packet delays would result in packets drifting into subsequent time-slots. As a result, packet-count vectors, and therefore flow sketches, would change drastically even under a slight packet delays. On the other hand, making the time-slots too long would result in the packet-count vectors of some uncorrelated flows being similar to each other. Our experiments suggest that, 500 millisecond long time-slots are an appropriate choice for our setting. Figures 5 and 6(a) depict that, using shorter time-slots (i.e. 300 milliseconds) weakens the resistance to the packet delays since detection rate is observed to decrease more rapidly for increased $maxDelay$ in Figrue 6(a) than in Figure 5. On the other hand, using longer time-slots i.e. 1000 milliseconds) makes the sketches of uncorrelated flows similar to each other. As a result, the 10^{-4} false alarm rate threshold becomes smaller and therefore the detection performance decreases again, as observed in Figure 6(b).

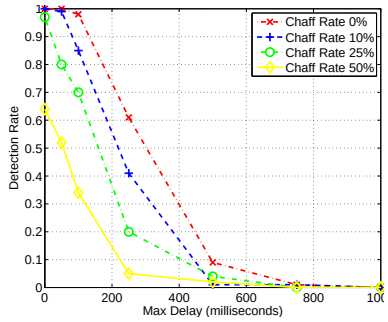
Effect of Sketch Length: Using short sketches is very important for the proposed scheme to be scalable. However, using too short sketches makes it harder to distinguish between correlated and uncorrelated flows. The reason is that, shorter sketches contains less information about the packet-timings of flows, thereby yielding increased probability of the sketches of two uncorrelated flows being similar to each other. More formally, when shorter sketches are used, the distribution of the Hamming Distances between the binary sketches of uncorrelated flows (Figure 4) will have higher variance, thereby pushing the 10^{-4} false alarm rate threshold towards zero and hence decreasing the detection performance. The improvement of the detection rate with the increased sketch length is observed in Figures 7(a), 5 and 7(b), where we used 256-bit, 512-bit and 1024-bit sketches, respectively.

Effect of Correlated Flows' Duration: It is expected that, the detection rate increases as the duration of the correlated flows increases. That is because, longer flows have more packets and therefore their linear transform coefficients have higher magnitudes. As a result, same amount of distortion on packet-count vectors has less impact on the integer-array sketches of longer flows, as discussed in Section III-A5. Hence, sketches of longer flows are more resistant to timing perturbations. This can be observed in Figures 8(a), 5 and 8(b), where we present the detection rates for correlated flow durations 30, 60 and 90 seconds, respectively. Consequently one can say that, the longer an attacker stays on-line, the more likely the proposed scheme detects her attack.

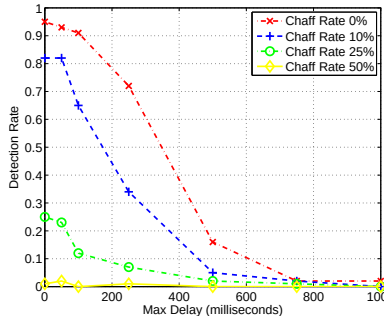
V. Limitations And Potential Solutions

High Chaff Rates: Although, stepping-stone attacks are constrained by a maximum tolerable delay, in theory there is no such constraint on chaff. That is, an attacker can introduce as many chaff packets as she would like (as long as the network is able to handle) in order to disrupt the observed correlation between her flows. However, in most cases, sending packets at an unusually high-rate might expose the attack instead of concealing it. Nevertheless, such cases are potential problems, not only for the proposed scheme but for all timing-based flow correlation schemes (i.e. link padding algorithms in the context of anonymity networks [13][14][15]). To mitigate such cases, one needs to identify flows with unusually high packet-rates and treat them separately. If these cases are relatively rare, one can afford to employ very complex algorithms to process them. One possible solution is that, flows with unusually high packet-rates can be tested by an active flow correlation scheme [16] [17][18], which marks an ingress flow by perturbing packet timings in a certain manner, and checks if any of the egress flows contain the embedded mark.

Flow Splitting/Merging: To evade detection, an attacker might employ more complex strategies, such as flow splitting or flow merging. In flow splitting, an attacker creates multiple egress flows sharing outgoing attack packets, instead of using only one egress flow. Or similarly, in flow merging, an

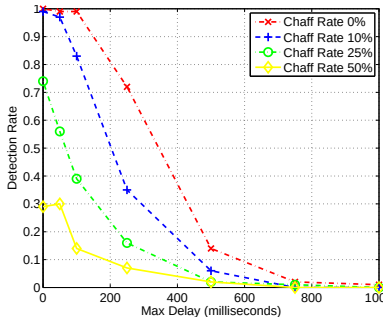


(a) Time-slot Length = 300 msec

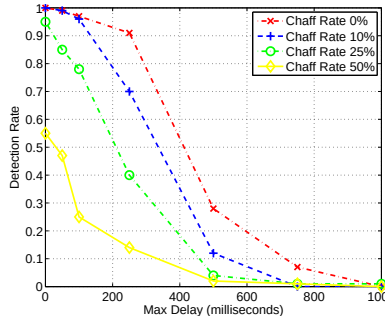


(b) Time-slot Length = 1000 msec

Figure 6. Effect of time-slot length, for 60 second long correlated flows and 512 bit binary sketches. Setting the time-slot length = 500 msec (Figure 5) performs better than both 300 and 1000 msec.

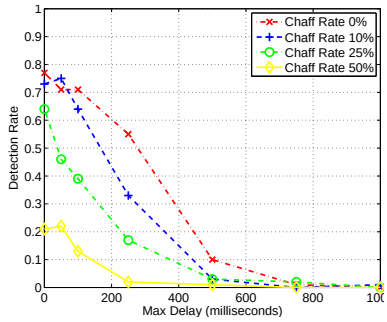


(a) Binary Sketch Length = 256 bits

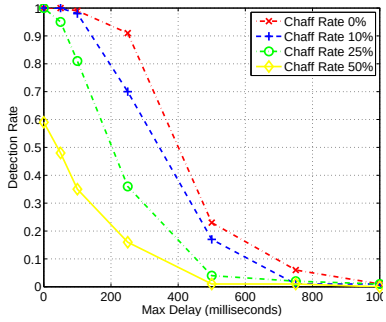


(b) Binary Sketch Length = 1024 bits

Figure 7. Effect of sketch length for 60 second long correlated flows and = 500 msec long time-slots. It is observed that, the detection rate increases with the increased sketch length.



(a) Correlated Flow Duration = 30 sec



(b) Correlated Flow Duration = 90 sec

Figure 8. Effect of correlated flow duration for = 500 msec long time-slots and 512 bit binary sketches. It is observed that, increasing flow duration increases the detection rate.

attacker sends attack packets to the monitored intermediate node over multiple ingress flows and merge them into one egress flow. To detect such cases, a stepping-stone detection scheme should compare superpositions of several ingress flows with superpositions of several egress flows for possible correlations. Unfortunately, such kind of search will be very costly since every possible combination of the flows have to be tested. However, due to the linear transformation employed by the proposed scheme, the integer-array sketch of the superposition of a set of flows is equal to the summation of the sketches of those flows. This linearity property enables quick and efficient computation of the binary sketch of any given combination of flows. Hence, one can potentially combine this linearity property with an appropriate optimization technique, such as dynamic programming, gradient search etc., to efficiently search for a small set of ingress flows and egress flows which are collectively correlated with each other. We leave the exploration of this issue as a future work.

VI. Related Work

Stepping-Stone Detection: Staniford-Chen and Heberlein proposed perhaps the first technique for the stepping stone detection problem in [19]. The basic idea proposed was to correlate two flows based on payload thumbprints. Zhang and Paxson proposed a flow correlation algorithm based on occurrence times of packet bursts and silence moments called ON/OFF periods in [6]. The assumption is that correlated flows switch from the OFF state to the ON state at similar times. Yoda and Etoh proposed a similar timing based algorithm in [8], where correlation was defined over sequence number vs. time curves of flows. The idea was that the curves of correlated flows should be close to each other. Another timing based algorithm was proposed by He and Tong in [20], where the authors formulate the stepping stone detection problem as nonparametric hypothesis testing. In [21],Wang *et. al.* use inter-packet delay information to compute the correlation between two flows.

The techniques explained above are all robust against packet delays and jitter to some extent. However, when attackers insert chaff packets into flows, the correlation mechanisms used by these schemes start to break down. In [11], Donoho *et. al.* argue that attackers have maximum tolerable delay constraints and the correlation between stepping stone flows can be detected regardless of chaff packets if flows last long enough. Similarly, under a maximum tolerable delay constraint, Blum *et. al.* present confidence bounds on the stepping stone detection problem in [7]. Their algorithm is based on the difference of the number of packets between two flows at a given time. This difference is expected to be low for correlated flows even if there are a few chaff packets. In [9] Zhang *et. al.* propose a packet-timing comparison algorithm with special focus on random delays and chaff. However, their algorithm compares

two flows in linear time in the number of packets. In [22] Strayer *et. al.* proposed a State-Space algorithm similar to the one used to identify wireless receiver/transmitter pairs. The idea is to maintain a conversation probability matrix (CPM) which shows the correlation between each flow pair based on the probability that the observed packet is caused a previously received packet on another flow. In [10], Wu *et. al.* tried to improve the chaff resistance properties of [7]. However, they assume that the chaff is introduced only one of the flows of a correlated flow pair. A similar timing-based flow correlation method is presented in [23], where authors identify Skype related correlated flows. In [24], Coskun and Memon proposed a technique to efficiently identify relay nodes in real time. However, their technique cannot identify the actual relaying flows. As a completely different approach, in [16], [17] and [18] authors propose an active stepping-stone detection scheme based on watermarking. The basic idea is to mark the timing information of ingress flows, and search for the traces of those marks in the egress flows. A similar watermarking technique is proposed in [25] to track VoIP calls.

Data Stream Sketching: In general, data sketching techniques can be viewed as linear projections of an input stream on appropriate basis functions [1]. They are widely used to answer efficient queries on streaming data such as rangesum, heavy hitters, quantiles, inner product etc.,[2][26][3].

Robust Multimedia Hashing: The proposed technique shows a resemblance to robust hashing schemes. They both represent an input signal by a short array (robust hash), which is resistant to small perturbations on the input. In the context of multimedia signal processing, robust hash functions are often used to identify and authenticate multimedia contents (audio, video and image) in the presence of perceptually preserving modifications such as compression, minor filtering etc. [27, 28, 12].

VII. Conclusion

In this paper, we presented a novel stepping-stone detection scheme based on an online algorithm, which continuously maintains sketches of network flows' packet-timing information from a stream of captured packets at the border of a network. These sketches are then used to efficiently identify correlated flows, since the correlated flows have similar sketches. The proposed scheme computes flow sketches very efficiently by a streaming algorithm, which performs a few arithmetic operations for each packet. In addition, the sketches of a pair of correlated flows remain similar, even if the flows encounter various timing perturbations, thereby allowing the proposed scheme detect the correlated flows even under delays, jitter, chaff, etc to some extent. Finally, using the fact that correlated flows have similar sketches, the proposed scheme identifies correlated ingress/egress flow pairs among n ingress and m egress flows in $O(n+\sqrt{nm})$ time, as compared to known techniques, which requires $O(nm)$.

Our experiments verify that the proposed scheme is resistant to chaff and packet delays to some extent. However, there is still room for improvement. We observed in the experiments that, when we use longer time-slots in order to be more resistant to packet delays, the sketches of uncorrelated flows start to exhibit some similarities, thereby negatively affecting the detection performance. Shorter time-slots, on the other hand, have limited resistance to packet delays. To combine the advantages of both sides and potentially improve the resistance to packet delays, we plan to use randomly varying time-slot lengths. Meanwhile, we used a fixed threshold when deciding if two flows are correlated or not. However, using a different threshold for each pair of flows, regarding several features, such as flow durations, number of packets etc, will potentially improve the detection performance. We leave the exploration of these improvements as future work.

References

[1] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, pp. 29–38, 2004.

[2] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*. London, UK: Springer-Verlag, 2002, pp. 693–703.

[3] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, "How to summarize the universe: Dynamic maintenance of quantiles," in *In VLDB*, 2002, pp. 454–465.

[4] D. Schnackenberg, H. Holliday, R. Smith, K. Djahandari, and D. Sterne, "Cooperative intrusion traceback and response architecture (citra)," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2001.

[5] D. Schnackenberg, K. Djahandari, and D. Steme, "Infrastructure for intrusion detection and responsen," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2000.

[6] Y. Zhang and V. Paxson, "Detecting stepping stones," in *Proceedings of the 9th USENIX Security Symposium*, August 2000, pp. 171–184.

[7] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: Algorithms and confidence bounds," in *Conference of Recent Advance in Intrusion Detection (RAID), Sophia Antipolis, French Riviera, France*, September 2004.

[8] K. Yoda and H. Etoh, "Finding a connection chain for tracing intruders," in *F. Guppens, Y. Deswarte, D. Gollamann, and M. Waidner, editors, 6th European Symposisum on Research in Computer Security - ES-*

ORICS 2000 LNCS -1985, Toulouse, France, October 2000.

[9] L. Zhang, A. Persaud, A. Johnson, and Y. Guan, "Stepping stone attack attribution in non-cooperative ip networks," in *roceedings of the 25th IEEE International Performance Computing and Communications Conference (IPCCC 2006)*, April 2006.

[10] H.-C. Wu and S.-H. S. Huang, "Detecting stepping-stone with chaff perturbations," in *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, 2007, pp. 85–90.

[11] D. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, "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.

[12] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system with an efficient search strategy," *Journal of New Music Research*, vol. 32, no. 2, pp. 211–221, 2003.

[13] X. Fu, B. Graham, R. Bettati, and W. Zhao, "On effectiveness of link padding for statistical traffic analysis attacks," in *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003, p. 340.

[14] P. Venkitasubramaniam and L. Tong, "Anonymous networking with minimum latency in multihop networks," in *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008, pp. 18–32.

[15] W. Wang, M. Motani, and V. Srinivasan, "Dependent link padding algorithms for low latency anonymity systems," in *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 323–332.

[16] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 20–29.

[17] Y. J. Pyun, Y. H. Park, X. Wang, D. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 634–642.

[18] A. Houmansadr, N. Kiyavash, and N. Borisov, "Rainbow: A robust and invisible non-blind watermark for network flows," in *NDSS'09: 16th Annual Network and Distributed System Security Symposium*, 2009.

[19] S. Staniford-Chen and L. Heberlein, "Holding intruders accountable on the internet," in *Proc. IEEE Symposium on Security and Privacy, Oakland, CA*, May 1995, p. 3949.

[20] T. He and L. Tong, "A signal processing perspective of stepping-stone detection," in *Proc. of IEEE CISS'06, Princeton, NJ*, 2006.

[21] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-packet delay based correlation for tracing encrypted connections through stepping stone," in *Proceedings of the 7th European Symposium on Research in Computer Securit, (ESORICS*, 2002, pp. 244–263.

[22] W. T. Strayer, C. E. Jones, I. Castineyra, J. B. Levin, and R. R. Hain, "An integrated architecture for attack attribution," in *BBN Technologies, Tech. Rep. BBN REPORT-8384*, December 2003.

[23] K. Suh, D. Figueiredo, J. Kurose, and D. Towsley, "Characterizing and detecting skype-relayed traffic," in *Proc. of Infocom*, 2006.

[24] B. Coskun and N. Memon, "Efficient detection of delay-constrained relay nodes," in *ACSAC'07: 23rd Annual Computer Security Applications Conference, Miami Beach, FL*, Dec 2007.

[25] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, 2005, pp. 81–91.

[26] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Surfing wavelets on streams: One-pass summaries for approximate aggregate queries," in *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, 2001, pp. 79–88.

[27] R. Venkatesan, S. Koon, M. Jakubowski, and P. Moulin, "Robust image hashing," in *Proc. IEEE Int. Conf. Image Processing*, 2000.

[28] J. Fridrich, "Robust bit extraction from images," in *ICMCS '99, Florence, Italy*, June 1999.