Compromising Anonymous Communication Systems Using Blind Source Separation

Ye Zhu
Cleveland State University
and
Riccardo Bettati
Texas A&M University

We propose a class of anonymity attacks to both wired and wireless anonymity networks. These attacks are based on the blind source separation algorithms widely used to recover individual signals from mixtures of signals in statistical signal processing. Since the philosophy behind the design of current anonymity networks is to mix traffic or to hide in crowds, the proposed anonymity attacks are very effective.

The flow separation attack proposed for wired anonymity networks can separate the traffic in a mix network. Our experiments show that this attack is effective and scalable. By combining the flow separation method with frequency spectrum matching, a passive attacker can derive the traffic map of the mix network. We use a non-trivial network to show that the combined attack works.

The proposed anonymity attacks for wireless networks can identify nodes in fully anonymized wireless networks using collections of very simple sensors. Based on time series of counts of anonymous packets provided by the sensors, we estimate the number of nodes with the use of Principal Component Analysis. We then proceed to separate the collected packet data into traffic flows that, with help of the spatial diversity in the available sensors, can be used to estimate the location of the wireless nodes. Our simulation experiments indicate that the estimators show high accuracy and high confidence for anonymized TCP traffic. Additional experiments indicate that the estimators perform very well in anonymous wireless networks that use traffic padding.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms: Security, Algorithms

Additional Key Words and Phrases: Anonymous Communication, Location Privacy, Blind Source Separation

Author's address: Y. Zhu, Department of Electrical and Computer Engineering, Cleveland State University, 2121 Euclid Ave, SH 433, Cleveland, OH 44115; email: y.zhu61@csuohio.edu; R. Bettati, Department of Computer Science, Texas A&M University, College Station, TX 77843; email: bettati@cs.tamu.edu. This work is partially supported by faculty research development grant (FRD) from Cleveland State University.

1. INTRODUCTION

One of the key premises in anonymity systems has been that large crowds protect the anonymity of individuals. Anonymity measures rely on users being able to "hide in a crowd", and the larger the crowd, the easier it is for the individual to hide. This tenet has been underlying early formulations of anonymity metrics (such as the *anonymity set size* in [Kesdogan et al. 1998]) and implicitly most newer metrics as well, such as [Serjantov and Danezis 2002] and [Diaz et al. 2002]. Empirical results support this assumption, given that larger numbers of participants in a distributed scenario naturally decrease the signal-to-noise ratio of the footprint of any individual.

In this paper we propose data *pre-conditioning* methods that allow to *partition* the set of participants, and therefore significantly increase the footprint of individuals. We formulate these pre-conditioning methods as a class of attacks, and we apply them to on both wired and wireless anonymity networks. We use Blind Source Separation (BSS), a methodology from statistical signal processing to recover unobserved "source" signals from a set of observed mixtures of the signals. These attacks can be launched against *wired* anonymous communication networks to un-mix traffic and thus affect traditional anonymity such as sender anonymity, receiver anonymity or sender/receiver anonymity. These attacks can also be launched against *wireless* anonymous communication networks to isolate and identify participants and to compromise location privacy.

1.1 Flow Separation Attack on Wired Anonymous Communication Networks

Since Chaum [Chaum 1981] pioneered the basic idea of anonymous communication with the use *mixes*, researchers have developed various mix-based anonymity systems for different applications. Mix networks anonymize communication by perturbing the traffic in (a) the payload domain (through encryption), (b) in the route domain (through re-routing) and (c) in the timing domain (through aggregation of batching and link padding).

In this paper we discuss *flow separation*, which aims at separating (as opposed to identifying) flows inside a network, based on aggregate traffic information only. Once flows have been successfully separated, and therefore single flows or small groups of flows identified, subsequent frequency spectrum matching [Zhu et al. 2004] or time domain cross-correlation [Levine et al. 2004] can then easily determine additional information about the flow, such as the path taken, the ingress and egress points into and out of the observed network.

Once flows have been separated, information about the *traffic content* of the flows can be inferred as well. As a practical example, Dusi *et al.* [Dusi et al. 2008] describe the difficulty of classifying traffic over SSH tunnels when multiple flows are multiplexed over the same tunnel. Later in this paper we will illustrate how flow separation yields representations of flows that highly correlate to the existing flows. After such highly accurate separation of flows, the classification mechanisms described by Dusi *et al.* can be applied effectively on aggregates of tunneled flows.

We use experiments to show that flow separation attacks are effective for both single mixes and mix networks. We also show that further attacks can make use of the information about the separated flows and so be very effective in reducing anonymity. We analyze the effect of multicast/broadcast traffic on the flow separation attack. In contrast to intuition, our analysis and experiments show that the presence of multicast/broadcast traffic significantly helps the attacker to more precisely separate the flows. We discuss the possible use of flow separation attacks in other anonymity network settings and pros and cons of counter-measures.

1.2 Location Privacy Attack on Wireless Anonymous Communication Networks

With the increasing popularity of 802.11 style wireless networks (WLANs), both in infrastructure and in ad-hoc mode, *location privacy* issues in such networks and in ubiquitous computing environments in general have received great attention. Much recent work has focused on the identification of location privacy risks associated with the use of WLANs and on the implications of weak location privacy (e.g., [Cuellar et al. 2004]). *Locating* a node in a wireless network typically requires first to *identify* the node (where some identifier is associated with the node, without necessarily disclosing the node's user identity,) before proceeding to the geographic location proper. In densely populated networks, node location is difficult without prior identification, since it is impossible to properly attributed traffic to nodes and so to keep the nodes apart. In sparsely populated networks, the identification step is trivial and can be skipped altogether.

A number of schemes exist to prevent simple node identification at MAC layer or above through appropriate encryption and the use of one-time MAC address [Gruteser and Grunwald 2003] or broadcast-only communication [Kong and Hong 2003]. More sophisticated node identification approaches rely on interactions with services and access points [Beresford and Stajano 2003] and may be countered by schemes such as path perturbation [Hoh and Gruteser 2005], in which nodes report appropriately modified locations whenever they are close to other nodes, with the goal to confuse the location tracker. Once nodes have been identified, the location proper can be performed with the help of propagation analysis [Castro et al. 2001], directional information [Malhotra et al. 2005], or signal strength analysis [Bahl and Padmanabhan 2000]. A system that counters signal-strength based location through manipulation of sender signal strength is described in [Cai et al. 2005].

Appropriate pre-conditioning of collected traffic data using flow separation allows an attacker to compromise the location privacy in a densely populated, *perfectly anonymized* wireless network. The traffic data could be collected by very simple sensors, which only need to monitor packets at MAC level or above, do not require directional capabilities, do not need to distinguish packets or relate network packets to senders or receivers, only require coarse time synchronization support, and require only low-bandwidth links for inter-sensor communication. (We don't need support for signal-strength measurement on the sensors either.) Such collections of sensors could be realized by a number of WLAN users that collude and exchange information, or by a separate infrastructure of sensor nodes, such as a sensor network. Given these limited required capabilities, we use the sensors to count packets over intervals of given length, and to forward the resulting time series of packet counts for analysis to some central location. No information is available about how many nodes are present and sending in the area, and the anonymity measures in the WLAN prevent the sensors from distinguishing packets sent from different nodes.

Our experimental evaluations using a widely accepted packet-level network simulator (ns-2) indicate that (a) the proposed algorithms estimate the node density with high accuracy and that (b) they estimate node locations with both high accuracy and high confidence. The majority of experiments is performed with the intent to simulate naturally occurring (i.e. TCP) traffic over typical anonymizing wireless networks, such as ANODR [Kong and Hong 2003]. In order to show the effectiveness of the approach, we also simulate a network that uses constant-rate padding of traffic on all nodes. Our experiments indicate that traffic padding is largely useless in this setting: it has no impact on the effectiveness of our estimators for both node-density and node-location.

We consider these results significant, since they indicate that it is impossible to maintain location privacy in 802.11-style networks against colluding WLAN users or networks of sensors that use simple off-the-shelf technology. Our experiments show that crowds are unable to hide individuals in WLAN settings. BSS algorithms can easily and effectively separate packets from different senders, based on packet-count time series only.

1.3 Statistical Methods for Flow Separation

The pre-conditioning methods described in this paper rely on Blind Source Separation (BSS) [Jutten and Herault 1991], which was originally developed to solve variations of the *cocktail party problem*, where the goal is to extract one speaker's voice signal given a mixtures of voices, presumably at a cocktail party. BSS algorithms solve the problem by taking advantage of the independence between voices from different speakers. We will show how BSS can be used in wired networks to "un-mix" traffic flows Similarly, in wireless networks, we can use BSS algorithms to separate traffic from different wireless nodes. The separated traffic is not in a form that can be directly associated with any sender node. However, we take advantage of spatial diversity in the collected data to reconstruct the path of a flow through a mix network, or the the sender location in a wireless network, based on the separated traffic.

While the number of "mixed flows" through a wired mix are known (assuming one knows the number of ports of the mix router,) the number of senders in a wireless network is typically unknown. In [Zhu and Bettati 2007] we describe how we use Principal Component Analysis (PCA) to estimate the number of sending nodes.

1.4 Structure of the Paper

The remainder of this paper is organized as follows: Section 2 reviews the related work. In Section 3, we introduce the blind source separation algorithms. In Section 4, we introduce the flow separation attack for the wired anonymity networks. Section 5 describes the attacks to compromise location privacy in wireless anonymity networks. In Sections 6 and 7, we use ns-2 simulation experiments to show the effectiveness of the flow separation attack. We evaluate the flow separation attack against a non-trivial mix network in Section 8. Section 9 evaluates the performance of location privacy attacks for wireless anonymity networks. Section 10 discusses countermeasures for flow separation attack and further attacks on location privacy. We conclude this paper in Section 11.

2. RELATED WORK

2.1 Traditional Wired Anonymity Networks and Anonymity Attacks

Reviving Chaum's idea to use mixes to reroute messages, Helsingius [Helsingius 1996] implemented the first Internet anonymous remailer, which is a single application proxy and replaces the original email's source address with the remailer's address. Eric Hughes and Hal Finney [Parekh 1996] built the cypherpunk remailer, a distributed mix network with reply functions and encryption. Gülcü and Tsudik [Gulcu and Tsudik 1996] developed a relatively complete anonymous email system, called Babel. Möller and Cottrell [Moeller and Cottrell 2000] developed Mixmaster, which counters a global passive attack by using message padding. It counters trickle and flood attacks [Gulcu and Tsudik 1996; Serjantov et al. 2002] by using a pool batching strategy. Danezis, Dingledine and Mathewson [Danezis et al. 2003] developed mixminion. Mixminion's design considers a large set of attacks that researchers have found [Serjantov et al. 2002; Berthold and Langos 2002; Berthold et al. 2000; Raymond 2001].

Low-latency anonymity systems have been developed for flow-based traffic in the Internet. A typical example is Tor [Dingledine et al. 2004], the second-generation onion router, developed for circuit-based low-latency anonymous communication.

Recently, *ad hoc* anonymizing networks have been discovered to hide the communication to command-and-control nodes in botnets: The *Storm* [Porras et al. 2007] botnet, for example, is assumed to use multiple layers of Nginx proxies to hide the command-and-control infrastructure from view. An additional layer of Nginx nodes then act as reverse proxies and uses fast-flux to hide the location of the command server despite using hard URLs for the bot nodes when "calling home" to the controller. The Storm command-and-control system is an example of effective server hiding, made possible by redirection mechanisms in combination with jurisdictional boundaries. As botnets evolve, it is to be expected that they will develop more sophisticated anonymization protocols, which will take advantage of the very large number of bot nodes available.

Exploiting the premise that communication patterns are difficult to efficiently hide, a large amount of work has addressed the effectiveness of message-based anonymity attacks [Serjantov et al. 2002], a class of flow-based anonymity attacks have been proposed in the literature. Examples are intersection attacks [Danezis and Serjantov 2004], timing attacks [Levine et al. 2004], Danezis's attack on continuous mixes [Danezis 2004], and the flow correlation attack [Zhu et al. 2004]. The timing attack [Levine et al. 2004] uses time domain cross-correlation to match flows given the packet timestamps of the flow. Danezis's attack on the continuous mix [Danezis 2004] takes advantage of the fact that packets are independently delayed at the mix, and the effect of the mix can be modeled as a convolution on the packet times. The flow correlation attack [Zhu et al. 2004] employs statistical methods to detect TCP flows in aggregate traffic. The flow correlation attack can achieve high detection rates for all the mixes described in [Serjantov et al. 2002] and for continuous mixes. The flow separation attack proposed in this paper belongs to the flow-based anonymity attacks.

2.2 Wireless Anonymity Networks and Location Privacy

Following the realization that serious privacy issues are at stake when location information is accessible in many pervasive applications and wireless networks (see, for example, the work of the IETF Working Group on Geographic Location/Privacy (geopriv) [Cuellar et al. 2004]), several communication systems to preserve anonymity and location privacy in ad-hoc and infrastructure-based wireless networks have been proposed. ANODR [Kong and Hong 2003] protects route anonymity by an onion-based encryption and routing protocol. The data transmission in ANODR is based on broadcast, and identity disclosure is prevented by the use of broadcast MAC addresses. MASK [Zhang et al. 2005] also provides a solution for anonymous routing in wireless ad-hoc network. In particular, forging packets and inserting dummy packets are proposed to counter traffic analysis. For infrastructure-based networks, Gruteser et al. [Gruteser and Grunwald 2003] proposed the use of disposable MAC addresses in order to prevent tracking of mobile hosts.

Numerous papers have been published on locating wireless nodes. Many of them are based on the characteristic of physical signals, such as Received Signal Strength (RSS) [Bahl and Padmanabhan 2000], Angle of Arrival (AOA) [Niculescu and Nath 2004], and Time of Arrival (TOA) [Ismail Guvenc and Dedeoglu 2003]. Complex processing methods on collected data are needed to deal with the physical signal's non-linearity, noise, and the complex correlations caused by multi-path effects, interference, and absorption. Elnahrawy *et al.* [Elnahrawy et al. 2004] point out a number of fundamental limits associated with the use of signal strength and claim that these limits are unlikely to be transcended.

Senders can easily counter location estimation attacks based on signal-strength by modulating the transmission power. This has been proposed in Whisper [Cai et al. 2005]. Location privacy attacks using Angle-of-Arrival data assume that sensors have directional capabilities, which adds greatly to the cost of the sensor network. One objective of this section of the paper is to illustrate how appropriate pre-conditioning of the collected traffic data renders most of current anonymity methods for wireless networks, such as encryption, MAC address hiding, signal power fluctuations, link padding, and others are ineffective in 802.11-style setting. For this, we show that we need not make use of information that can be hidden by anonymity measures, such as header data, or sender or receiver information, or packet data, or signal strength, or directional information.

In general, schemes that rely on physical-level, analog signals require large volumes of data to be transfered over the wireless sensor network for further analysis. In comparison, the schemes proposed in this paper rely on highly aggregated packet-count data which can be easily propagated across a low-bandwidth infrastructure. Spatial and temporal redundancy of the packet-count data from different sensors can be exploited to further reduce traffic volume by using appropriate compression methods.

3. BLIND SOURCE SEPARATION

Blind Source Separation (BSS) is a methodology in statistical signal processing to recover unobserved "source" signals from a set of observed mixtures of the signals. The separation is called "blind" to emphasize that the source signals are not observed and that the mixture is a black box to the observer. While no knowledge is available about the mixture, in many cases it can be safely assumed that source signals are independent. A very nice introduction to the statistical principles behind BSS is given in [Cardoso 1998]: In its simplest form, the blind source separation model assumes n independent signals $F_1(t), \cdots, F_n(t)$ and n observations of mixture $O_1(t), \cdots, O_n(t)$ where $O_i(t) = \sum_{j=1}^{n} a_{ij} F_j(t)$. The goal of BSS is to reconstruct the source signals $F_j(t)$ using only the observed data $O_i(t)$, the assumption of independence among the signals $F_j(t)$. Given the observations $O_i(t)$, BSS techniques estimate the signals $F_j(t)$ by maximizing the independence between the estimated signals. The common methods employed in blind source separation are minimization of mutual information [Comon 1994; He et al. 2000], maximization of nongaussianity [Hyvarinen 1999; Hyvarinen and Oja 1997] and maximization of likelihood [Gaeta and Lacoume 1990; Pham et al. 1992].

In our experiments, we use a powerful BSS algorithm proposed in [Cruces and Cichocki 2003]. This algorithm can jointly optimize several statistics of the same order, and it combines advantages of other effective techniques such as Fast-ICA [Hyvarinen and Oja 1997]and SOBI [Belouchrani et al. 1997]. The algorithm has been shown to perform well when the amount of data available is small.

4. FLOW SEPARATION IN MIX NETWORKS

4.1 Network Model and Threat Model

Mix and Mix Network A mix is a relay device for anonymous communication. A single-mix network can achieve some level of communication anonymity: The sender of a message attaches the receiver address to a packet and encrypts it using the mix's public key. Upon receiving a packet, the mix decrypts the packet using its private key. Different from an ordinary router, a mix usually will not relay the received packet immediately. Rather, it will attempt to perturb the flows through the mix in order to foil an attacker's effort to link incoming and outgoing packets or flows. It does this, typically, in three ways: First, it re-encrypts the packet to foil attacks that attempt to match packets in the payload data domain. Then, it *re-routes* the packet to foil correlation attacks that rely on route traceback. Finally, it perturbs the flows in the time domain through *batching*, *reordering*, and *link padding*. Batching collects several packets and then sends them out in a *batch*. The order of packets may be altered as well. Both these batching techniques are important in order to prevent timing-based attacks. Different batching and reordering strategies are summarized in [Serjantov et al. 2002] and [Zhu et al. 2004].

Most low-latency anonymity systems do not employ batching, reordering, and padding strategies. For example, Onion Router [Goldschlag et al. 1999], Crowds [Reiter and Rubin 1998], Morphmix [Rennhard and Plattner 2002], P5 [Sherwood et al. 2002], and Tor [Dingledine et al. 2004] do not use any batching and reordering techniques.

A network may consist of multiple mixes that are inter-connected by a network such as the Internet. A mix network may provide enhanced anonymity, as payload packets may go through several mixes so that if one mix is compromised, anonymity can still be maintained. In the following we will use the term *mix* to mean either

single-node mixes, or mix cascades, or networks of mixes.
Threat Model We assume a passive adversary, whose capabilities are summarized as follows:

(1) The adversary passively observes a number of input and output links of a mix or among a collection of mixes, collects the packet arrival and departure times, and analyzes them. This type of attack can be easily staged on wired and wireless links [Howard 1998] by a variety of agents, such as governments or malicious ISPs.
(2) For simplicity of discussion, we assume a *partially global* adversary, i.e. an adversary that has observation points on a subset of links between mixes in the mix network. While this assumption seems overly strong, it is not: An attacker with access to only a small number of points will naturally aggregate mixes for which it has no observation points into *super-mixes*.
(3) The adversary cannot correlate (based on packet timing, content, or size) an individual packet on an input link to another packet on an output link based on content and packet size. This is prevented by encryption and packet padding, respectively.
(4) We focus on mixes that operate as simple proxies. In other words, no batching or reordering is used. Link padding (with dummy packets) is not used either. This follows the practice of some existing mix networks, such as Tor [Dingledine et al. 2004], and other obfuscation mechanisms, such as SSH tunnels and the cascaded Nginx proxies in the Storm botnet.
(5) Finally, the adversary aims to recover the *a-priori* unknown individual flows from the aggregate of indistinguishable traffic. Once a flow or a number of flows have been recovered, the attacker can proceed to either classify the content of the flow, or to compute the path of the flows, or the end-to-end communication matrix, or other measures of interest.

4.2 Flow Separation

In this section, we will first define the problem in the context of blind source separation and then describe how to apply the flow separation method in a wired mix network.

4.2.1 *Flow Separation as a Blind Source Separation Problem.* We define a *flow* to be a series of packets that are exchanged between a pair of hosts. Such a flow can be a single TCP or UDP connection, or can be carried by (striped over) several connections, both UDP or TCP. In this paper we limit ourselves to flows whose packets all flow along a single path. In other words, while packets from different flows can take different paths, all the packets from a single flow follow the same path. We define an *aggregate flow* at the *link-level* to be the sum of the packets (belonging to different flows) on the link. We define the aggregate flow at *mix-level* to be the sum of packets through the same input and output port of a mix. Unless specified otherwise, the word "flow" in the remaining of this paper means "mix-level aggregate flow" for brevity.

We will show in this paper that, for the attacker who tries to break the anonymity of a mix, it is very helpful to *separate* the flows through the mix based on the observation of the traffic. The separation of the flows through the mix can recover the traffic pattern of individual flows or small groups thereof. This fine-granularity information can be used to increase the effectiveness of further analysis, such as the frequency spectrum matching attack described in Section 4.2.2 or the time domain cross-correlation attack [Levine et al. 2004] for flow reconstruction, or application characterization over encrypted tunnels [Dusi et al. 2008], or other flow confidentiality situations.

In this paper, we are interested in the traffic pattern carried in the time series of packet counts during a sequence of identical-length intervals of length T. For example, in Figure 1, the attacker can get a time series $O_1 = [o_1^1, o_2^1, \cdots, o_n^1]$ of packet counts by observing the link between Sender S_1 and the mix. We use n to denote the *sample size* in this paper. The attacker's objective is to recover the packet count time series $F_i = [f_1^i, f_2^i, \cdots, f_n^i]$ *for each flow*. For the simplest case, we assume that (a) there is no congestion in mix and that (b) the time series can be synchronized. (We will relax both assumptions in later sections.) In the example of Figure 1, the time series F_1 is contained in both time series O_1 and O_3 i.e. $O_1 = F_1 + F_2$, $O_3 = F_1 + F_3$. In general, for a mix with j input ports, k output ports and m mix-level aggregate flows, we can rewrite the problem in vector-matrix notation,

$$\begin{pmatrix} O_1 \\ O_2 \\ \vdots \\ O_{j+k} \end{pmatrix} = \mathbf{A}_{(j+k)\times m} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_m \end{pmatrix} \tag{1}$$

where $\mathbf{A_{(j+k)\times m}}$ is called *mixing matrix* in the BSS problem. Since the aggregate flows through the mix are independent from each other – they come from different sources – we can use any of the methods mentioned in Section 3 to solve the problem. Even the flows from a single host, such as F_1 and F_2, can be regarded as independent as they follow different paths and are controlled by different sockets. This independence assumption is of course only valid as long as Sender S_1 is not heavily overloaded, since otherwise one flow would influence the other.

In the following, we need to keep in mind that flow separation often is not able to separate individual flows. Rather, mix-level aggregates flows that share the links at the observation points form what we call the *minimum separable unit*.

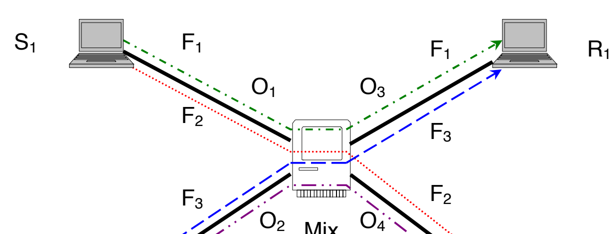


Fig. 1. An Example for Flow Model

Basic BSS algorithms require the number of observations to be larger than or equal to the number of independent components. For flow separation, this means that $j + k \geq m$, where j and k denote the number of observations at the input and output of the mix, respectively, and m denotes the number of flows. Advanced blind source separation algorithm [Hyvarinen and Inki 2002; Hyarinen et al. 1999] target over-complete bases problems and can be used for the case where $m > j + k$. But they usually require that m, the number of independent flows, be known. Since all the mix traffic is encrypted, and all packets padded to the same length, it is hard for the attacker to estimate m. We assume that $m = j + k$. The cost of the assumption is that some independent flows can not be separated, that is, they remain mixed. We will see that this is not a severe constraint, in particular not in mix networks where flows that remain mixed in some separations can be separated using separation results in neighboring mixes.

Unless there is multicast or broadcast traffic through the mix, the $j + k$ observations will have some redundancy, because the summation of all the observations on the input ports are equal to the summation of all the observations on the output ports. In other words, the row vectors of the mixing matrix are linearly dependent. Again, the cost of the redundancy is that some independent flows are not separated.

The flow estimation generated by blind source separation algorithms is usually a scaled version of the actual flow (of its time series, actually). Sometimes, the estimated flow may be of different sign than the actual flow. Scaling does not affect the frequency components of the time series, and so frequency matching can be used to further analyze the generated data.

Furthermore, since the elements of the estimated mixing matrix are not binary, it is not straightforward to tell the direction of each aggregate flow. Some heuristic approach can be used, but we leave this to further research.

When monitoring points can be placed at some locations inside the network, separation results at different locations can be compared, and common separated components identified if such components exist. This can be used to reconstruct the path of flows, or simply to strengthen the confidence in the results of the flow separation step.

4.2.2 *Frequency Matching Across Neighboring Mixes.* The result of flow separation is a set of mix-level aggregate flows that have been determined to traverse the mix. Each separated aggregate flow is identified by a time series of packet counts. As pointed out earlier, some of these aggregate flows may represent multiple actual flows that could not be separated. Aggregate flows can be further separated by comparing BSS results across multiple mixes.

Frequency spectrum matching has shown to be particularly effective to further analyze the traffic. The rationale for the use of frequency matching is four-fold: First, the dynamics of a flow, especially a TCP flow [Huang et al. 2001], is characterized by its periodicities. By matching the frequency spectrum of a known flow with the frequency spectrums of estimated flows obtained by blind source separation techniques, we can identify the known flow with high accuracy. Second, frequency matching removes the zero-frequency component, and so easily eliminates ambiguities introduced by the scaling in the estimated time series. Third, frequency spectrum matching can also be applied on the mix-level aggregate flows, since the different frequency components in each individual flows can characterize the aggregate flow. Fourth, the low frequency components of traffic are often not affected by congestion as they traverse multiple switches and mixes. This is particularly the case for TCP traffic, where the frequency components are largely defined by the behavior at the end hosts. In summary, frequency spectrum analysis has excellent prerequisites to be highly effective.

Even if no information is available about individual flows, the attacker can easily determine if there is communication between two neighboring mixes. Matching the estimated aggregate flows across neighboring mixes can give attackers more information, such as how many aggregate flows are traversing the next mix. In a mix network, an aggregate flow through a mix may split into aggregate flows of smaller size, multiplex with other aggregate flows, or do both. By matching the estimated aggregate flows on mixes along estimated paths, and by comparing the results against those of other mixes, the attacker can detect such splitting and multiplexing. Based on the information gathered, and if sufficient monitoring points are available in the network, the attacker can eventually get a detailed map of traffic in a mix network, *without having access to information about any individual flow.* In Section 8, we show a traffic map of flows in a large network, which was obtained from BSS followed by aggregate flow matching.

5. COMPROMISE LOCATION PRIVACY IN WIRELESS ANONYMITY NETWORKS

Wireless anonymity networks naturally offer themselves very well to a BSS analysis. In a wireless setting with passive sensors, the packet flows originating at the senders give rise to the unobserved signals, and the packets overheard by the sensors produce the observed signals. Finally, the wireless medium acts as mix matrix. BSS can therefore be used to separate senders and to study how this supports follow-up attacks, such as localization of senders.

5.1 Network Model and Threat Model

In the following we assume a passive attacker who is interested in determining the number of nodes in the network (*node density*) and the geographic location of the nodes (*node location*). Disclosure of node location may aid privacy attacks, but may be used in other settings as well. For example low-cost intrusion detection schemes for ad-hoc networks can perform node density estimations at deployment time and determine if active intruders are present. If so, location estimations can support the localization of the intruder.

Network Model We assume a set of wireless nodes (simply called *nodes* in the following) that communicate over an ad-hoc WLAN using an 802.11-style MAC protocol. We assume that all communications are *perfectly anonymized*: For example, all communications are broadcast-based so that the anonymity attacker cannot identify the source and destination of a MAC frame [Kong and Hong 2003; Zhu et al. 2004]. Similarly, MAC addresses can be recycled [Gruteser and Grunwald 2003] to achieve the same effect. We also assume that all the packets inside the wireless network are encrypted. As a result, no information is divulged to external observers either through packet data or header content. In addition, we assume that nodes are able to manipulate signal power (e.g. [Cai et al. 2005]) so as to render any observed signal strength information difficult to use.

Threat Model We derive the thread model from that described for the wired case in Section 4.1, and adapt it to the case of anonymized wireless systems, similar to [Kong et al. 2003]. We assume that the communication between nodes is observed by a network of low-cost sensors scattered around a field. The sensors can be either WLAN receivers of a set of colluding users in the area, or they can be deployed as a separate sensor network infrastructure. The attacker collects packet timing information from the sensors in the field for analysis.

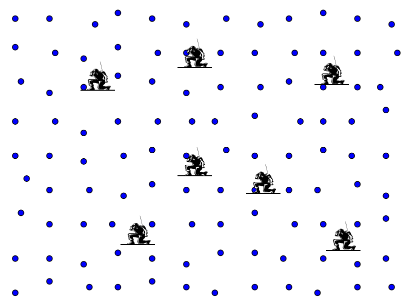


Fig. 2. Threat Model

We summarize the capabilities of the passive attacker as follows:

(1) Sensor nodes have off-the-shelf 802.11 receivers.
(2) No signal strength information is available.
(3) No directional information is available.
(4) Time synchronization across sensor nodes is insufficient to allow for signal-propagation based location estimation.
(5) A sensor cannot associate a packet with a sender or receiver node.
(6) The location of each sensor is known.
(7) The sensor are scattered randomly, but uniformly, over the networked area of interest, as shown in Figure 2.
(8) The communication between sensors does not interfere with the communication between wireless nodes.

5.2 Data Collection and Pre-processing

The data collected from each sensor is a time series of counts of the packets "overheard" by the sensor. (While sensors cannot decrypt the packets or even associate packets with a mobile node, they can mark the time when a packet is received, and so *count* the number of packets received over any interval.) We use the time series $S_i = [s_i^1, s_i^2, \cdots, s_i^l]$ to denote the series of packet counts detected by Sensor i during a sequence of intervals of length T each. Since there may be several wireless nodes in the field, the packet counts on each sensor may contain packets from several wireless nodes. Similarly, the same packet may be counted by multiple sensors.

As for any data gathering application on sensor networks, power consumption and bandwidth limitations are important design issues. Only packet *counts* are collected from the sensor, and so the resulting amount of data is significantly less than from collecting, say the time-stamp of each packet. In addition, we can use data compression or coding schemes designed for sensor networks such as MEGA [von Rickenbach and Wattenhofer 2004] to exploit any remaining spatial redundancy across neighboring nodes or temporal redundancy at individual nodes.

5.3 Node Location Estimation

In a network where all packets are perfectly anonymized, the estimation of the location of sender nodes has only *aggregated* packet data available, since packets sent by different nodes cannot be distinguished. In the following we describe how we use Blind Source Separation to *de-aggregate* the packet-count time series collected at a group of sensors into an estimation of the *per-node* packet-count time series M_j of Sender Node j. Based on the estimated per-node time series we then use a proximity-based scheme to estimate the location of nodes.

5.3.1 *Blind Source Separation.* While the goal of BSS in this context is to reconstruct the original signals M_i (the time series of packet counts sent by individual nodes), in practice the separated signals (we call these *components*) are sometimes only loosely related to the original signals. We categorize these separated components into three types: In the first case, the component is correlated to some signal M_i. We call this type of component *individual* component. In the second case a component may be correlated to an *aggregate* of signals from several nodes. This happens when the packets of more than two wireless nodes can be “heard” by all the sensors. In such a case, the BSS algorithm would not be able to fully separate the signal mixture into the individual components. Rather, while some components can be successfully separated, others remain aggregated. In the third case, components may represent noise signals. Noise in our case can be caused by packet collisions that prevent some sensors from “hearing” some packets. Noise can also emerge as artifact from generating the packet timing sequences. For example a packet may be counted in the i^{th} interval for some sensor while for some other sensor the same packet may be counted in the $(i+1)^{th}$ interval due to transmission delay or imperfect synchronization. For brevity, we call the second type *aggregate* component and the third type *noise* component.

5.3.2 *Node Location Estimation Algorithm.* An algorithm to estimate the location of sender nodes would consist of three steps: In a first step we partition the network area into a set of *mini areas.* In order to maximize both spatial resolution and spatial diversity in the following steps we should choose the size of the mini areas to be small but sufficiently large so that most subareas contain at least one sensor. We group the nodes in each $c \times c$ neighboring set of mini areas into *sensor block* as shown in Figure 3[1]. Neighboring blocks are overlapping, and as a result sensors generally belong to several sensor blocks. (For the case of a quadratic blocks, most sensors belong to c^2 blocks.) For each block of sensors we sequentially apply a BSS algorithm to recover the packet traces of mobile nodes in the sensing range. As a result of this block-by-block separation step, we are left with a large set of components, as described in Section 5.3.1. Many of these components are either aggregates or noise components. In a second step we eliminate the latter by

[1]In general, mini-areas and sensor blocks can be irregularly shaped. In Figure 3 we draw them to be quadratic.

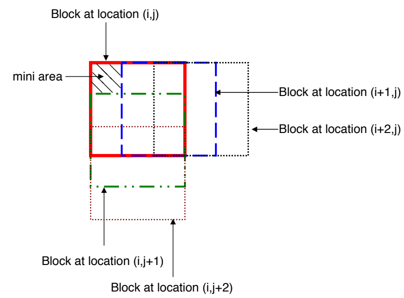


Fig. 3. Sensor Blocks

identifying components that appear in several blocks. This is achieved by identifying clusters of similar components across all blocks. This clustering step generates a set of components that have been detected by several blocks of sensors, and so are likely to be similar to the original signals.

At this point the senders have been identified, and the attacker can proceed to extract additional information. In the following, we focus on localization of the senders. In the third and last step, we therefore estimate the location of the senders by intersecting the sensing ranges for all blocks that have separated components that are highly correlated to the original signals.

We describe the three steps (separation, clustering, intersection) in more detail:

Separation Step For each sensor block we apply BSS to recover a set of s *components* as described in Section 5.3.1 for a sensor block with s sensors. We use R_i^j to represent the j^{th} recovered component from the i^{th} sensor block.

Clustering Step We eliminate those components that are likely to be noise or aggregate components. For this we use the following heuristic: If a component represents a real signal, the same component has likely been detected and separated in at least a similar form by more than one sensor block. In contrast, a component that was generated because of some interference or other artifacts has likely been generated by a single block only.

Based on this heuristic we identify clusters of similar components by using the cross correlation coefficient as measure for similarity, and define the distance between two components as follows:

$$D(R_i^p, R_j^q) = 1 - \left\| corr(R_i^p, R_j^q) \right\| \quad , \tag{2}$$

where R_i^p denotes the p^{th} component recovered from the i^{th} sensor block B_i, and $corr(X, Y)$ denotes the correlation coefficient of components X and Y. (We use the absolute value of the cross correlation because the separated components may be of different sign than the actual time series.) As a result, the highly correlated (similar) components will cluster together. Figure 4 uses a two-dimensional representation to further illustrate the rationale for the clustering approach in this step. As shown in this figure, the individual components form clusters. The aggregate components on the other hand scatter in-between these clusters. The noise components are distant both from each other and from the other components.

We select the center components $R_1, \cdots, R_K$ of the K largest resulting clusters, where K is the estimated number of nodes in the area. The value for K is either known *a priori* or is estimated using a highly accurate method based on Principal

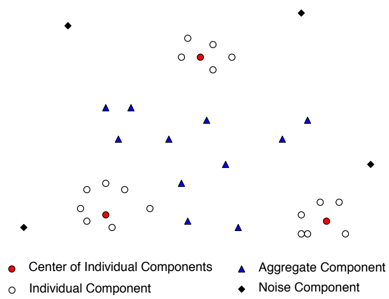


Fig. 4. Visualization of Distance between Separated Components

Component Analysis, which we describe in [Zhu and Bettati 2007] in detail. We note that it is highly unlikely that either aggregate or noise components are selected as center components: (a) Aggregate components are unlikely in the center of clusters, and (b) noise is local to a small group of sensors at best, and gives rise to small clusters. As a result, the K selected center components will be highly correlated to the packet count times series $M_1, \cdots, M_K$ of the nodes in the network.

Intersection Step We locate a sending node by intersecting the sensing ranges of blocks that are likely to "hear" the node. For this we select sensor blocks that have components that are closely correlated with the likely time series of the nodes in the area. The rationale is that for the sensors in a sensor block to hear a node, they must have sensed a signal that is at least similar to the signal generated by the node. This means that sensor blocks with components that correlate with any of the K center components are likely to hear a sending node. We therefore determine the likely location of a node by geographically intersecting the sensing areas of the sensors in those sensor blocks that have highly correlated component with a center component determined in the preceding clustering step.

A straightforward correlation of the components gives poor results, for two reasons: First, sensor blocks in the immediate neighborhood of a sender node often display insufficient spacial diversity to successfully separate the component representing the sender node packet time series. They must therefore be discarded. Second, simply correlating the components of the sensor block with a particular center component may lead to too many false positives because geographically very distant components may occasionally correlate with a particular center component. For example, correlated components may appear along the the multi-hop path of a connection in an *ad-hoc* network, and so give rise to geographically distant sensor blocks that are erroneously classified as being able to hear the node. In such cases, the geographical area intersection approach fails to locate the node. We address these problems by borrowing techniques and terminology from image processing.

For each center component R_k, we generate a "image matrix" IMG_k of correlation coefficients (the "intensity") as follows: each entry in the matrix $IMG_k(i,j)$ represents the maximum correlation between the center component R_k and all c^2 components of the block, say B_u, at location (i,j):

$$IMG_k(i,j) = \max_{1 \le p \le c^2} (\|corr(R_u^p, R_k)\|$$

We then apply edge detection to partition the matrix into geographically contiguous regions, each with components that correlate with the center component R_k. We discard the regions that are not sufficiently correlated with the center component. (A detailed discussion of the threshold selection in this step is given in [Zhu and Bettati 2007].)

We intersect the sensing area of sensor blocks in the remaining regions as follows: Sort the points $IMG_k(i,j)$ in the remaining regions in order of decreasing intensity. Starting with the highest-intensity point, add subsequent points by intersecting their sensing range. Stop when you either run out of points or the new point's sensing area is disjoint from the computed intersection area, causing the new intersection area to disappear. The resulting intersection area is the suspected area of location of a node.

6. EVALUATION OF FLOW SEPARATION ATTACK ON A SINGLE MIX WITH DIFFERENT COMBINATIONS OF TRAFFIC

Whenever we have access to monitors at the boundary of some network, we treat the network as a *single* mix (which we call *super mix*), independently of whether the network contains one or more mixes. Whenever we can plant additional monitors inside the network, we are *de-facto* partitioning the single-node mix view into that of a mix *network*. In this section we evaluate the single-mix case, and we will elaborate the results on the network case in Section 8.

6.1 Experiment Setup

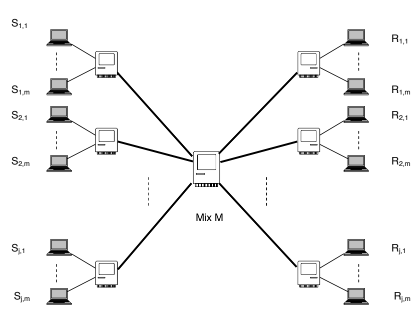


Fig. 5. Experiment Setup for Single Mix

Figure 5 shows the experimental network setup with a single mix. We use ns-2 to simulate the network. The links in the figure are all of $10Mbit/s$ bandwidth and $10ms$ delay[2] if not specified otherwise. The mix under study has two input ports and two output ports and four aggregate flows passing through the mix, as shown in Figure 1. We study mixes with more than two ports in Section 7. Unless specified otherwise, we use time observation intervals of $32s$ length and sample interval of $10ms$ length, resulting in time series of size $n = 3200$. Similar results were obtained for shorter observations as well.

[2]Senders and receivers can be at a large distance from the mix, potentially connecting through several routers and switches.

6.2 Metrics

In the following, we will use two metrics to evaluate the accuracy of the flow separation. The metrics compare the separated flows with the actual flows in the time domain and the frequency domain, respectively.

We use *mean square error (MSE)* to match separated and actual flows in the time domain as follows: Let $F_A = [f_1^A, f_2^A, \cdots, f_n^A]$ represent the time series of the actual flow and $F_B = [f_1^B, f_2^B, \cdots, f_n^B]$ represent the time series estimated by the BSS algorithm. To match the time series F_A with F_B, we first scale and lift F_B so that both series have the same mean and variance.

$$F'_B = \frac{std(F_A)}{std(F_B)} \cdot (F_B - mean(F_B) \cdot [1, 1, \cdots, 1]) + mean(F_A) \cdot [1, 1, \cdots, 1] \quad , \tag{3}$$

where $std(F)$ and $mean(F)$ denote the standard deviation and average of the time series F, respectively. The *mean square error* is defined as follows:

$$\varepsilon_{A,B} = \frac{\|F_A - F'_B\|^2}{n} \quad . \tag{4}$$

Since the times series F_B can also be a flipped version of F_A, we need to match F_A with $-F_B$ as well.

We use a second metric to evaluate how well the separated Flow F_B matches the actual flows in the frequency domain. (We describe in Section 4.2.2 how the correlation of actual and separated flows in the frequency domain is used when we analyze systems with multiple mixes.) To evaluate the performance in frequency domain, we use the second metric called *frequency spectrum matching degree*. We define the matching degree between F_B and F_A to be the absolute value of correlation between the spectrum of the separated flow F_B and the spectrum of the actual flow F_A.

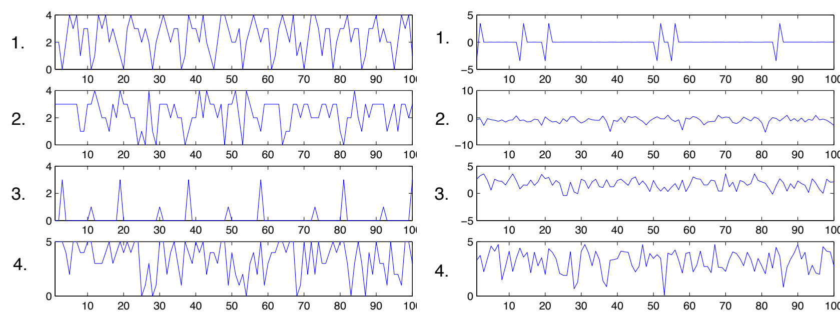
6.3 Different Types of Traffic

In this experiment, four aggregate flows, including one FTP flow, one sequence of HTTP requests, and two on/off UDP flows, are passing through the mix. The parameters for the flows are as follows: Flow 1: FTP flow, with round trip time around $80ms$. Flow 2: UDP-1 flow, on/off traffic, with burst rate $2500kbit/s$, average burst time $13ms$ and average idle time $6ms$. Flow 3: HTTP flows, with average page size 2048 byte. Flow 4: UDP-2, on/off traffic with burst rate $4000kbit/s$, average burst time $12ms$ and average idle time $5ms$. All the random parameters for the flows are exponentially distributed. The flows traverse the mix as shown in Figure 1.

Figure 6 shows portions of the actual times series (Figure 6(a)) and of the separated time series (Figure 6(b)). From the figures, it is apparent that the flipped version of the actual flow 3 (HTTP flows) is contained in the separated flow 2. We also observe the resemblance between actual flow 1 (FTP flow) and separated flow 4. Separated flow 1 is clearly not close to any actual flows. This is caused by the redundancy contained in the observations, as described in Section 4.2.1.

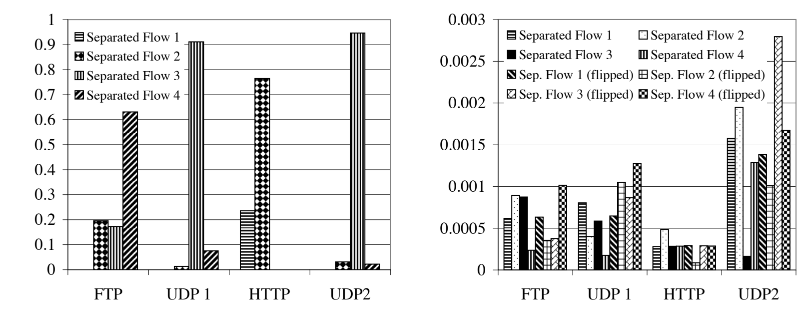
Figure 7 shows the separation accuracy using the two metrics defined earlier. We note in Figure 7(b) that both the separated flow and its flipped time series is compared against the actual flows. Both metrics can identify the FTP flow, the HTTP flows, and flow UDP-2. But the two metrics disagree on flow UDP-1:

while Flow UDP-1 correlates well in the frequency domain with separated Flow 3 (indicated by the highest frequency spectrum matching degree for Flow UDP-1 in Figure 7(a),) in the time domain it best matches with separated Flow 4(flipped) (indicated by the lowest MSE for Flow UDP-1 in Figure 7(a)). This is because of the redundancy in the observations, and the two UDP flows can not be separated. MSE fails for this case since it is designed for one-to-one flow matching while frequency spectrum matching is more suitable for matching of flows against aggregates. The latter case is more common in the context of flow separation, where often aggregates of flows cannot be completely separated.



(a) Actual Flows (b) Separated Flows

Fig. 6. Example of Flow Separation for Different Types of Traffic



(a) Frequency Spectrum Matching Degree (b) MSE

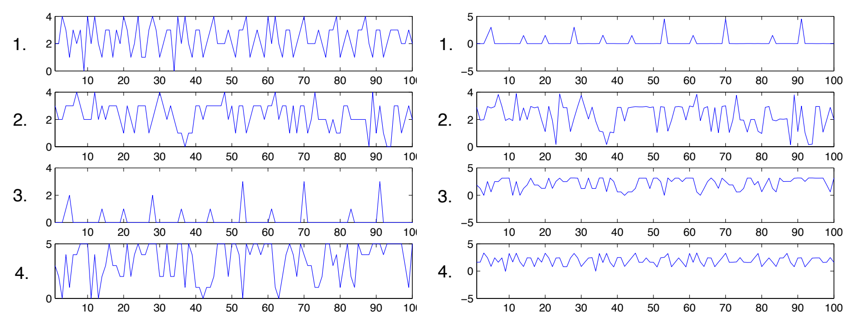
Fig. 7. Performance of Flow Separation for Different Types of Traffic

6.4 Different Types of Traffic with Multicast Flow

In this experiment, the Flow UDP-1 in the previous experiment is multicast to both output ports.

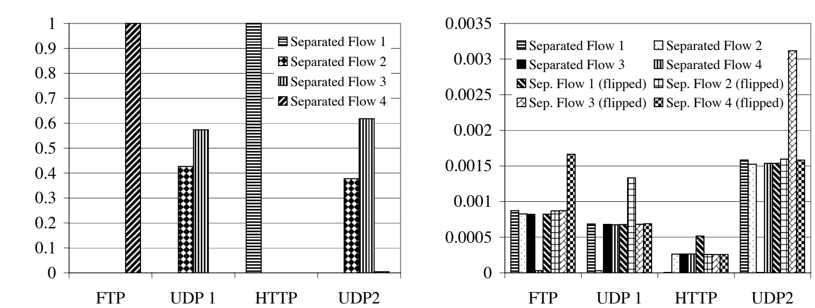
Portions of the actual flows and the estimated flows are shown in Figure 8. We observe the correspondence between the actual flows and estimated flows easily. In comparison with the previous experiment, we conclude that multicast flows can help the flow separation, as they eliminate redundant observations.

The MSE performance results in Figure 9 show that the flows are successfully identified. Frequency spectrum matching successfully determine the FTP and HTTP flows, but does not perform well on the UDP flows. This is because the two UDP flows have approximately similar periods, and the periodical behavior is not strong for exponential on/off traffic.



(a) Actual Flows (b) Separated Flows

Fig. 8. Example of Flow Separation for Different Types of Traffic (with Multicast Traffic)



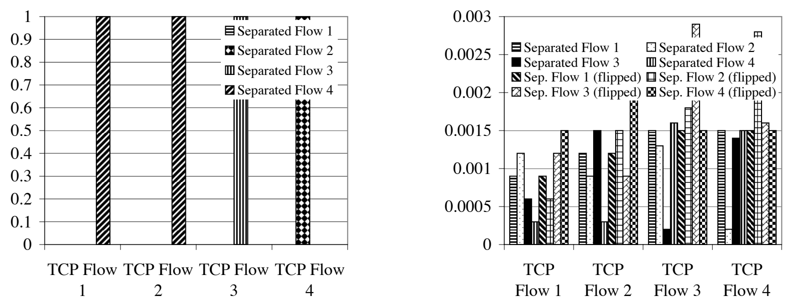
(a) Frequency Spectrum Matching Degree (b) MSE

Fig. 9. Performance of Flow Separation for Different Types of Traffic (with Multicast Traffic)

6.5 TCP-Only Traffic

Since most of the traffic in today's network is TCP, we focus on TCP traffic in the next series of experiments. All the flows in this experiment are FTP flows. To distinguish the flows, we vary the link delays between the sender and mix, with S_1 having $10ms$ link delay to the mix, and S_2 having $15ms$ delay[3].

Figure 10 shows the flow separation performance. Since there is no multicast traffic, the redundancy in observations results that TCP Flow 1 and TCP Flow 2 are still mixed. In the final result, the flows are identified successfully by the frequency spectrum matching method.



(a) Frequency Spectrum Matching Degree (b) MSE

Fig. 10. Performance of Flow Separation for TCP-Only Traffic (without Multicast Traffic)

6.6 TCP-Only Traffic with Multicast Flow

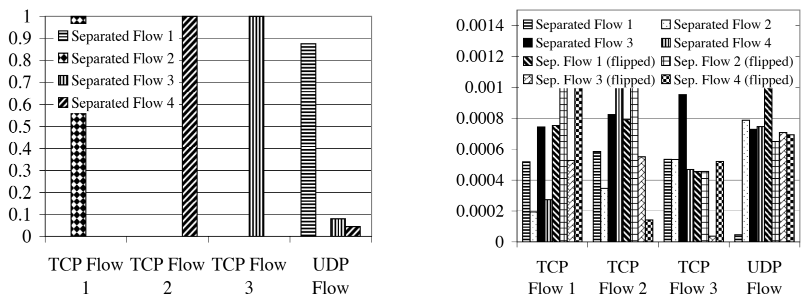
In this experiment, we change one FTP flow in the previous experiment to a multicast UDP flow. The UDP flow is exponential on/off traffic with the same parameters as UDP-1 of the experiment in Section 6.3.

Figure 11 shows the flow separation performance. Similarly to the effect of multicast flows on different types of traffic, the four flows are now separated completely. We also observe that the frequency spectrum method identifies the FTP flows successfully. The performance for the exponential on/off UDP flow is not as good as for FTP flows, however, because the frequency signature of exponential traffic is very weak.

7. EVALUATION OF SCALABILITY OF FLOW SEPARATION

In order to gain insight into how flow separation performs as systems scale, we evaluate the flow separation performance with respect to (a) increasing numbers of flows in the mix-level aggregate flows (the number of aggregate flows remains constant), (b) increasing numbers of mix-level aggregate flows, and (c) increasing numbers of ports per mix.

[3]The difference in link delays gives rise to different round-trip times for the different TCP connections. As we show in the experiments in Section 8.3, BSS is just as effective in separating TCP flows when link delays are identical, and therefore the round-trip times are very similar.



(a) Frequency Spectrum Matching Degree

(b) MSE

Fig. 11. Performance of Flow Separation for TCP-Only Traffic (with Multicast Traffic)

7.1 Experiment Setup and Metrics

In this series of experiment, the network setup is as shown in Figure 5: The setup consists of $m \times j$ sender hosts and $m \times j$ receivers hosts. The flows from the senders get routed through j sender-side routers (mixes) to Mix M, which forwards the traffic to the receiver-side routers (mixes) and finally to the receivers nodes. We note that there is no need to investigate mixes with unequal number of input and output ports(i.e. $j \neq k$), since BSS does not distinguish input from output anyway. All the flows are FTP flows. We vary the round-trip times by adding $5ms$ delay incrementally to the link connected to each sender.

In this series of experiment, we limit ourselves to frequency spectrum matching results as the performance metrics, mainly because they are more suitable for redundant observations when there is no multicast traffic. When there are more than one flows in one mix-level aggregate flow, we will use the frequency spectrum of the actual mix-level aggregate flow to match with the separated mix-level aggregate flows. We will show only the maximum matching degree for each actual flow.

7.2 Scalability: Size of Aggregate Flows

We first leave the number of observations constant and increase the number of FTP flows in each mix-level aggregate flow by adjusting m. The mix under study still has two input ports and two output ports. The directions of the aggregate flows are still as show in Figure 1, except that each aggregate now contains $\frac{m}{2}$ flows.

Figure 12 shows the performance of the flow separation for different aggregate sizes. We use $< m, B, T >$ to represent different experiments, where B denotes the bandwidth of each link in the experiment, and T denotes the sample interval.

First, we observe that for the experiments with link bandwidth $10Mbit/s$ and sample interval $10ms$, the performance decreases when the aggregate size increases. This is because the TCP flows tends to “fill the link bandwidth” when there is enough data to send. When the number of FTP flows increases, congestion happens, and the time series of individual flows get perturbed as they traverse the mix. In addition, packets get dropped, which perturbs the time series even more. The perturbation caused by congestion and by the packet drops degrades the separation performance.

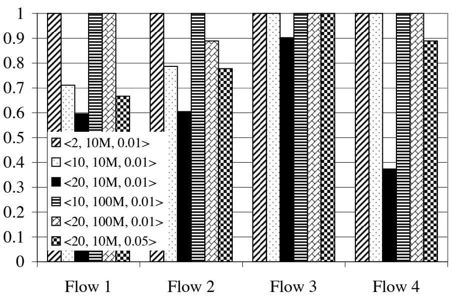


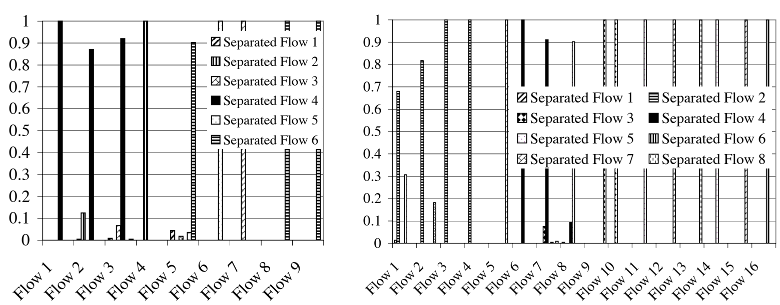
Fig. 12. Frequency Spectrum Matching Degree for Different Size of Aggregate Flows

Second, we observe that if we increase the bandwidth from $10Mbit/s$ to $100Mbit/s$, the performance significantly increases for the same aggregate size and sample interval. Obviously, less congestion causes better performance.

Third, for the same aggregate size and link bandwidth, increasing sample intervals increases the performance. Larger sampling intervals reduce the boundary effects caused by non-perfect sampling and by lack of synchronization. As a result, signal to noise ratio increases, and flow separation performance increases as well.

7.3 Scalability: Number of Aggregate Flows and Number of Flows

In this set of experiments, we change the number of mix ports and the number of the flows through the mix. We describe three typical experiments here.



(a) 3 × 3 mix, 9 Flows

(b) 4 × 4 mix, 16 Flows

Fig. 13. Frequency Spectrum Matching Degree

In a series of experiments on six aggregate flows through a 3×3 mix we observe that the frequency spectrum of the separated flow matches that of the original flows nearly perfectly. We also observe that performance of the flow separation remains good when the number of ports increases as long as the ratio of number of aggregate flows to number of observations (ports) remains constant.

Figures 13(a) and 13(b) show the performance of experiments with 9 mix-level aggregate flows through a 3×3 mix and 16 mix-level aggregate flows through a 4×4 mix, respectively. Our experiments show that when the number of aggregate

Flows	Path	Parameters	Throughput (packets/s)
1	$S_1 \rightarrow M_1' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_5' \rightarrow R_1$	FTP	106.125
2	$S_2 \rightarrow M_1' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_7' \rightarrow R_5$	FTP	100.791
3	$S_3 \rightarrow M_2' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_6' \rightarrow R_3$	FTP	95.936
4	$S_4 \rightarrow M_2' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_8' \rightarrow R_7$	FTP	91.541
5	$S_5 \rightarrow M_3' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_5' \rightarrow R_2$	FTP	87.531
6	$S_6 \rightarrow M_3' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_7' \rightarrow R_6$	FTP	83.858
7	$S_7 \rightarrow M_4' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_6' \rightarrow R_4$	FTP	80.483
8	$S_8 \rightarrow M_4' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_8' \rightarrow R_8$	FTP	77.357
9	$\rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_{10} \rightarrow$	Pareto	319.317
10	$\rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_9 \rightarrow$	Pareto	318.558
11	$\rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_{10} \rightarrow$	Pareto	321.806
12	$\rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_9 \rightarrow$	Pareto	323.36

Table I. Flow Configuration

flow increases, fewer flows can be separated. In other words, the number of the flows that remain mixed increases. This is because the ratio of aggregate flows to observations increases, and the BSS algorithm needs more observations. Meanwhile, the frequency spectrum matching degree remains high. This indicates that the flows have been correctly separated.

8. EVALUATION OF FLOW SEPARATION ATTACK FOR MIX NETWORKS

Flow separation can also be useful in mix *networks* with either local or partially global passive attackers. The attacker can perform flow separation between the available monitoring points, and then use the separation results to gather further information about the traffic. As an example of this approach, in the following we show a set of experiments where the attacker correlates the separated aggregate flows to first derive the path taken by the flows and then to generate the traffic map of the flows in the network. We point out that the attacker can infer this information *without having access to traffic information about any individual flow.*

8.1 Experiment Setup

Figure 14 shows the network setup in this experiment. A total of 8 FTP flows from senders on the left side are traversing the mix network. To distinguish these 8 FTP flows, we incrementally add a $5ms$ delay to links connected to each sender. In Section 8.3 we illustrate how we also get excellent results with identical delays on links, and describe why round-trip times on connections have little effect on separation performance. To simulate the cross traffic in the mix network, four larger aggregates of flows are added to the mix network. In order to reflect the self-similar nature of the network traffic [Park and Willinger 1999], the high-volume cross traffic is Pareto distributed. The configuration of the flows is shown in Table I.

In the center of the mix network, the traffic volume ratio between link-level aggregate traffic and each individual flow from senders is at least $7:1$. We assume that the attacker can observe links connected to Mix $M_1, M_2, \cdots, M_{12}$. Thus, a flow originating from S_1 can take 2^6 possible paths.

To measure the accuracy of the flow path reconstruction based on flow separation, we use an entropy-based metric as follows: Suppose we are interested in flow F_x. The attacker can suspect the flow F_x taking a path P_i with probability p_i based on the information gathered from the flow separation in the mix network. Assuming there are h possible paths that can be suspected as the path taken by Flow F_x, we

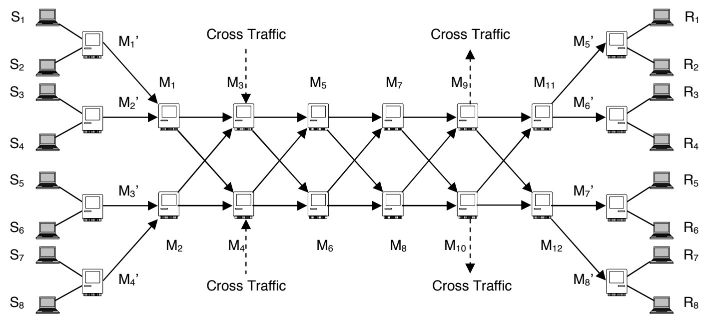


Fig. 14. Experiment Setup of Mix Network

define the flow path *obfuscation degree* D as

$$D = -\sum_{i=1}^{h} p_i \log_2 p_i \quad . \tag{5}$$

Suppose a flow originated from S_1 in Figure 14 is suspected to use each of 2^6 possible paths with equal probability. Then the path obfuscation degree D is 6 bit.

8.2 Performance

Figure 15 shows the mean value of the cross correlation using frequency spectrum matching among the first four FTP flows and separated flows recovered from Mix $1-12$. The cross-correlation values less than 0.1 are marked as white. We note that the cross-correlation values between separated flows recovered from the same mix are also marked as white. This includes the cross-correlation (auto-correlation) for the same separated flow or FTP flow.

From the cross-correlation map shown in Figure 15, we can derive the traffic direction in the mix network by linking the traffic flows with dark pixels in the map.

Figure 16 shows a dynamic-programming algorithm to detect a flow, say F_x in the network based on flow separation attack and frequency spectrum matching method. The main idea behind the algorithm is to first use the aggregate flow F_{tmp}, which is determined to be on the path previously to match the separated flows on the neighboring mixes. The threshold *threshold_1* is used to determine the Candidate array which includes the separated flows that have some components of the identified aggregate flow F_{tmp}. Then we match the flow F_x with the separated flows in the Candidate array to determine the most closely matching flow on the next hop. The process continues until the correlation is too weak, which is determined by the threshold *threshold_2*. Thresholds *threshold_1* and *threshold_2* can be determined by online learning based either on data collected by attacker or on some heuristic setting.

This algorithm is very effective. While the obfuscation degrees of all flows F_1 to F_8 in the experiment have a value of 6 bit before the attack (i.e., all possible paths are equally likely in the 6 mix stages), the obfuscation degree after the attack

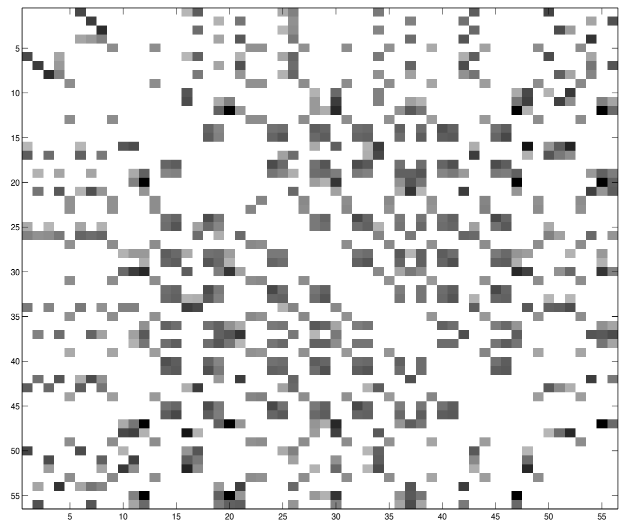


Fig. 15. Mean Value of Cross Correlation between Four FTP flows and Estimated Flows

F_{tmp}=F_x
M_{tmp}=M_x
while (mix M_{tmp} is not a dead-end) do {
    empty Candidate array
    for each mix M_i connected to M_{tmp} {
        for each flow F'_y separated by flow separation attack on M_i {
            matching(F_{tmp}, F'_y)=Cross-correlation coefficient of the frequency spectrums of F_{tmp} and F'_y
            if matching(F_{tmp}, F'_y)> threshold_1
                record (F'_y, M_i) into array Candidate
        }
    }
    find the element (F'_{max}, M_{max}) in candidate array, so that
    matching(F_x, F'_{max}) ≥matching(F_x, F'_y), for any F'_y in Candidate array
    if matching(F_x, F'_{max}) <threshold_2
        break
    F_{tmp}=F'_{max}
    M_{tmp}=M_{max}
    record M_{max} as a mix on the flow path
}

Fig. 16. Flow Detection Algorithm

drops to zero bit for all flows except for flows F_5 and F_7, where the the obfuscation degree drops to 0.5 bit. We set the thresholds *threshold_1* to zero and *threshold_2* to 0.1 heuristically. The result is based on the observations of 32 seconds of traffic.

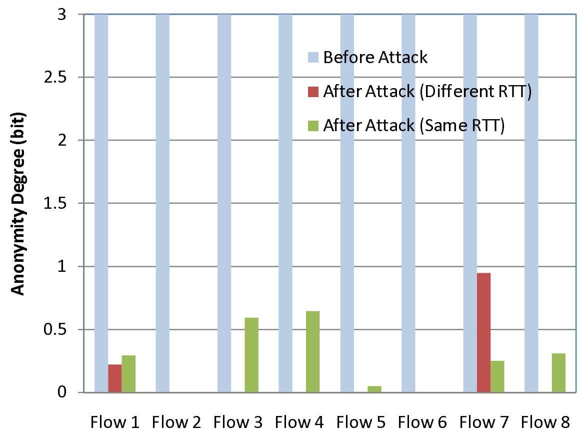


Fig. 17. Anonymity Degree

Our data indicates that similar results can be obtained with significantly smaller observation intervals.

8.3 Identical Round-Trip Time TCP Flows

In this set of experiments, we illustrate that the level of variance in the round-trip time does not affect performance of traffic flow separation. For example, an immediate (and, as it turns out, naive) approach for a countermeasure would be to make the round trip times of traffic flows identical in order to increase dependence between traffic flows, and in turn to render flow separation ineffective. In the following we will show that flow separation remains effective for TCP flow even in the case of identical link delays (and therefore similar round-trip times). We reuse the experiment setup used in the previous set of experiments but for a slight modification: The link delays of all links connected to senders are set to 5ms instead of having the delays increase from link to link. As a result the round trip times are very similar for all the eight traffic flows.

We measure the receiver anonymity by using the anonymity degree defined in [Serjantov and Danezis 2002] [4]:

$$D_{rec} = \sum_{i=1}^{n} p_i \log_2(p_i) \tag{6}$$

where p_i represents the probability that the ith receiver is identified as the flow of interest, and n is the number of possible receivers.

Figure 17 illustrates the performance of flow separation for the case of different RTTs and similar RTTs. We observe that separation performance for flows of similar RTTs and different RTTs is comparable. To investigate the cause for this comparable performance, we correlate the original traffic flows in both the identical-RTT experiments and the different-RTT experiments, as shown in Figure 18. We make two observations: (a) During the first 10 seconds, the correlation across the

[4] Sender anonymity is defined similarly.

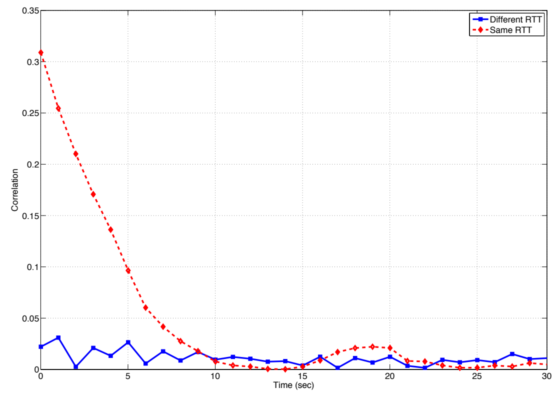


Fig. 18. Correlation of Original Flows

traffic flows with similar RTTs is much higher than the correlation across traffic flows of different RTTs. (b) After 10 seconds, the correlation among traffic flows of similar RTT is comparable with that of traffic flows of different RTTs. This decrease in dependence of traffic flows with similar RTTs explains the comparable performance in flow separation as shown in Figure 17.

9. EVALUATION OF PRIVACY ATTACKS ON WIRELESS NETWORKS

Having illustrated the effectiveness of BSS to separate traffic flows in wired networks, we proceed to evaluate its effectiveness to separate, identify, and locate senders in anonymized wireless networks. Similarly to the previous experiments, we perform a series of simulations in the ns-2 network simulator.

9.1 Experiment Setup

In the following experiments we simulate a field with a 1600m $\times$ 1600m square area. We assume that the sensors are arranged in a grid, and the distance between two neighboring sensors in the sensor grid is 50m. To eliminate boundary effects, the location of the wireless node is restricted to a 1000m $\times$ 1000m *center area* of the simulated field. The wireless network interfaces of both wireless nodes and sensors are modeled according to the commercial Lucent WaveLan radio interface, which has a nominal radio range of 250m. For the sensors the transmission function is disabled, so that they can only eavesdrop on the traffic. All simulations have a duration of 200 seconds. The packet count data is sampled with a sample interval of 50ms. We place 20 wireless nodes inside the center area. There are 36 randomly generated TCP flows in the wireless ad-hoc network. To make sure that every wireless node sends packets, every wireless node has at least one TCP flow that originates from it. The size of the mini area is 50m $\times$ 50m, so that each area contains one sensor, and each sensor block is a 3 by 3 array of sensors.

9.2 Effectiveness of Location Estimation

Performance Metrics As described in Section 5.3.2, the output of the location estimation algorithm is the suspected area of location of a node. To evaluate the performance according to the suspected area, we quantize the whole area using 5m $\times$ 5m tiles. The suspected area is represented by a set of points inside the suspected area, each point representing the corner of the corresponding tile. Two metrics are used to evaluate the area: One is the *mean error distance* between the points inside the suspected area and the actual location of a wireless node. The other is the *standard deviation of the error distance* between the points inside the suspected area and the actual location of a wireless node. The first one measure the *accuracy* of the detection algorithm and the second measures the *precision* of the detection algorithm. If we cast the evaluation of the estimation algorithm in terms of evaluating a statistical estimator, the accuracy corresponds to the *bias* of the estimator and the precision corresponds to the *variance* of the estimator.

To evaluate the intermediate result of the clustering step in Section 5.3.2, we compare the K selected center components with the actual packet count time series of corresponding wireless nodes. The metrics used for comparison is the absolute value of the cross-correlation. We use absolute value here to account for the possibility that the separated component is of different sign than the time series.

Performance We run our algorithm on three different types of node arrangements: *grid* arrangement, *random* arrangement and *clustered* arrangement. Examples of typical results of our location algorithm are shown in Figure 19. Please note that in Figure 19(c) two relatively large suspected areas are removed to prevent overlapping with other suspected areas. The two larger suspected areas are caused by the two pairs of closely located nodes near point A and point B, respectively, in Figure 19(c). These closely located nodes cannot be differentiated by the sensor grid, so the number of actual differentiable nodes is less than the number of nodes we know.

To quantitatively evaluate our algorithm, we run simulations for both random and clustered arrangements. For clustered arrangements, the locations of wireless nodes are generated by a Gaussian distribution with standard deviation of 100m for both x and y axes. The mean of the distribution is chosen to arrange the wireless nodes around the center of the field. The results for clustered arrangements indicate that sparse arrangement of sensor grids yield good results as well. In Figure 19(c), for example, only about 300 sensors are needed to localize the nodes.

Figure 20 shows the accuracy and precision resulting from the experiment, which lead us to the following five observations:

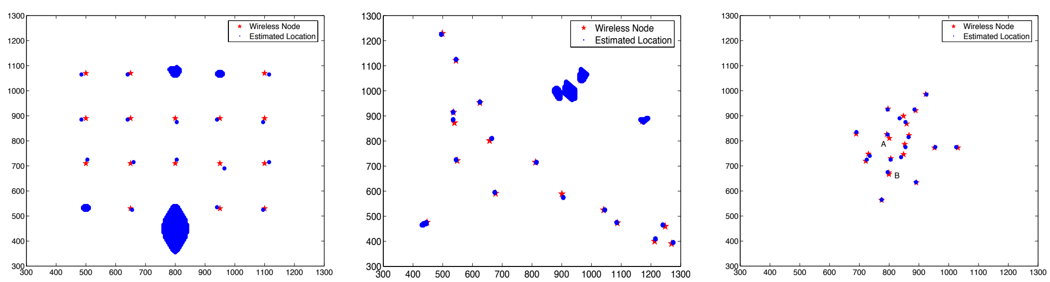
(i) The algorithm accurately locates the nodes. For random arrangements, node location estimations have an error of less than 30m in more than 90% of the cases (more than 85% for clustered arrangements).

(ii) The algorithm precisely locate nodes. For random arrangements, more than 90% of the detections have error standard deviation less than 20m (more than 85% for clustered arrangements).

(iii) As to be expected, the performance of the algorithm degrades when nodes are clustered. This is because of three reasons: *(a)* Clustered arrangements tend to have wireless nodes located close enough so that the sensor grid can not differentiate them. *(b)* For clustered arrangements, increased contention for the wireless

medium causes increased dependency between packet sending time series from different wireless nodes, thus affecting the performance of BSS which relies on the independence of underlying signals. *(c)* For clustered topologies, sensor blocks may overhear packets from more nodes than the number of sensor in the block. In such cases BSS runs into the so-called *overcomplete base* problem where some components can not be separated. In summary, effect (a) is equivalent to "thinning out" the sensor network. In addition, both effects (b) and (c) cause the performance of blind source separation to degrade.
(iv) In some cases the location estimation algorithm performs significantly worse for random arrangements than for clustered arrangements. While this appears counter-intuitive at first, it is easy to explain: Since clustered nodes are tending to be close to each other, their signals cover a smaller region than the nodes in random arrangements. Thus, large errors for clustered arrangements are unlikely.
(v) As a sanity check, we correlate the selected center components with actual time series. Figure 21 shows the correlation: we observe that the selected center components are highly correlated to the actual time series.

We also evaluate our approach based on topology. The 95% confidence intervals for per-topology mean error distance are from 15 to 59 and from 22 to 43 for random topologies and clustered topologies, respectively.

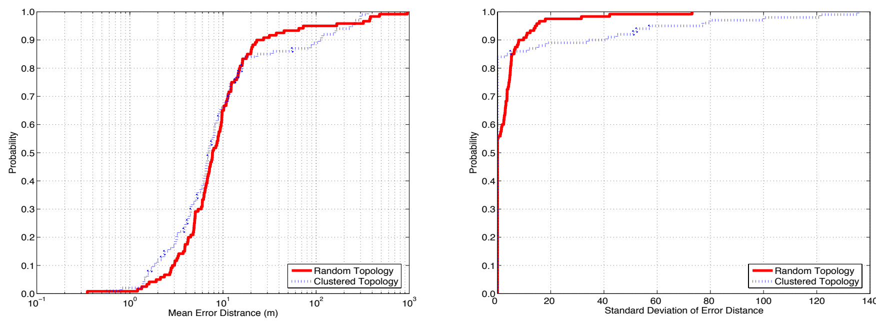


(a) Grid Topology (b) Random Topology (c) Clustered Topology

Fig. 19. Location Estimation on Different Topologies (Note: Two larger suspected areas are removed from Figure (c) to prevent overlapping.)

Constant-Rate Traffic In this experiment we evaluate our location detection algorithm against a network with *constant-rate link padding*. In such a network, packets are sent at constant intervals, and dummy packets are added whenever the link would idle. In the experiment, each wireless node sends constant-rate UDP packets to one of its neighbors. We use the grid topology of Figure 19(a). The choice of neighbor is made so that two loops are formed, with the outside nodes forming an outer loop and the inner nodes an inner loop. The packet sending rate of each wireless node is 40 packet/s and the bandwidth utilization is about 80%. The goal of this setup is to evaluate an arrangement that is as uniform as possible, thus making the separation and location problem maximally hard.

The results of this experiment are shown in Figure 22. We observe that the location detection algorithm is also effective against constant-rate traffic and heavy



(a) Mean Error Distance

(b) Standard Deviation of Error Distance

Fig. 20. Performance of Location Estimation Algorithm (Empirical CDF)

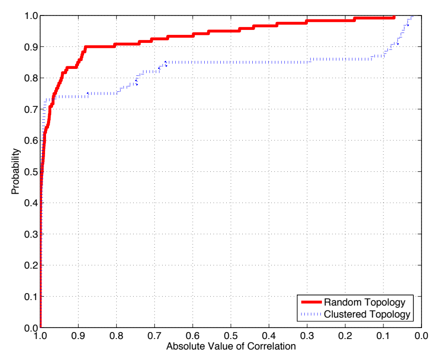


Fig. 21. Absolute Value of Correlation with Actual Time Series (Empirical CDF)

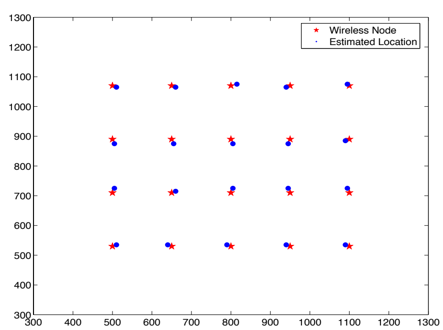


Fig. 22. Location Estimation Result (Constant Rate Traffic)

traffic. While the flows are constant-rate at sender application level, they are perturbed by the 802.11 MAC protocol, which adds enough timing signature to the flows, and so helps to separate the traffic. This experiment also illustrates that traffic padding at network layer or above is largely ineffective. A MAC level traffic padding scheme that consider both the media control protocol and bandwidth efficiency is needed.

10. DISCUSSION

10.1 Countermeasures to Flow Separation Attack

The countermeasures to flow separation attack are intuitive.

—Padding of the links to render the observations obtained by the passive attacker non-distinguishable, or at least mostly redundant.
—Use of pool-mix batching strategies. Pool mixes delay packets according to some probability distribution. If the delay probabilities are sufficiently small, the aggregate flows at the output ports can differ significantly from the aggregate flows at the input ports. This adds noise to the passive attacker's observations and can degrade the performance of flow separation attacks. The cost of such an approach is increased packet transfer latency and lower throughput, especially for TCP traffic.
—Increase of the dependency among flows by adding dependent dummy traffic flows to the mix-level aggregate flows.
—Padding of aggregate flows to render the distribution of the packet counts Gaussian. Most BSS algorithms fail when the signals mixed are Gaussian distributed. Some classes of BSS algorithm, however, make use of the time structure of the signals, and can still separate the flows, e.g., [Tong et al. 1991; Molgedey and Schuster 1994].

In general, BSS algorithms coping with noisy delayed signals and over-complete base problems are still active research topics in BSS research. Flow separation attacks will be more powerful when more advanced such algorithms become available.

10.2 Further Attacks on Wireless Networks

The mechanisms used to estimate density and location information can be used to infer additional information about the wireless nodes as well:

Traditional sender/receiver/route anonymity: For a given intensity image IMG_k, we can apply an edge detection algorithm to reveal the sender/receiver relationship as well as information about the communication path. The result of an example attack is shown in Figure 23. From the intensity image shown in Figure 23(a) we can observe the relationship between the sender and the receiver. A contour of the route taken by a flow is shown in 23(b). (The locations of the sender and receiver are marked with a star.)

Identity privacy: If *a priori* information is available about a wireless user, such as a model for communication or motion, the identity can be derived by correlating the K separated center components with the available models.

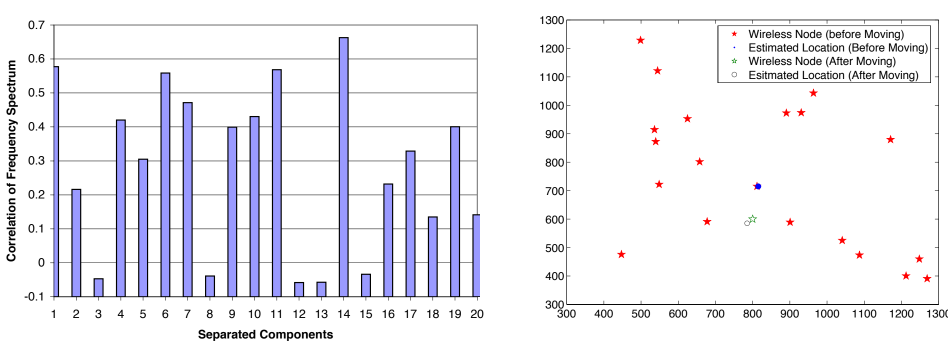
Motion privacy: To detect the motion of the wireless nodes in a field, we periodically compare the selected components by correlating them in the time domain and by tracking their estimated locations. When routes change, and time correlation disappears, one can make use of spatial correlation until routes stabilize again.

(a) Intensity Image IMG_k (b) Edge Detection

Fig. 23. Sender/Receiver/Route Anonymity Attack

Suppose the attacker finds out that component R_i^t at some time t is highly correlated to component $R_j^{t+\delta}$ at some later time $t+\delta$, and the location of component R_i^t and $R_j^{t+\delta}$ is estimated to be $area_i^t$ and $area_j^{t+\delta}$ respectively. The attacker can infer that a node has moved from $area_i^t$ to $area_j^{t+\delta}$. From the analysis of node location privacy, we can get the location of k nodes at time t and $t+\delta$. Under the assumption that routing does not change dramatically from time t to $t+\delta$, we can find the correspondence between k signals at time t and k signals at time $t+\delta$ through correlation. When routes do change from time t to time $t+\delta$, the timing behavior of flows can change as well, due to new contention situations or different path lengths. If routing does change, we can take advantage of correlation in the space domain: since the speed of nodes is limited, for small values of δ the location of a node can not vary indeterminately.

Figure 24 shows the result of an experiment for the simple case of a single moving node. Figure 24(a) shows the correlation between the spectrum of the center component corresponding to the moving node before the move and the spectrum of all the center components after the move. The correlation is maximum between the spectrum of the center component corresponding to the moving node before the move and the spectrum of the center component (Component 14) corresponding to the moving node after the move. Figure 24(b) shows the motion estimation based on the two center components with maximum correlation.



(a) Spectrum Correlation (b) Motion Estimation

Fig. 24. Motion Privacy Detection

11. CONCLUSION

In this paper, we proposed anonymity attacks to both wired and wireless anonymity networks. These attacks are based on the blind source separation algorithms widely used to recover individual signals from mixtures of signals. Since the philosophy behind the design of current anonymity networks is to mix traffic or to hide in crowds, the proposed anonymity attacks are very effective.

We show in a series of experiments that flow correlation is effective in separating flows about which no *a priori* information is available. When several observation points are available throughout a network, we show that separation re-creates the flow information at sufficiently accurate level so that frequency-spectrum correlation across observation points can re-construct the path of the (unknown) flows. In the same way, we show how flow separation can also be used to simply recover the traffic map of the anonymity network. We discuss the possible usage of flow separation in different flow confidentiality settings, such as ssh tunnels or anonymity network settings, and we elaborate on criteria for its countermeasures.

Users of wireless networks are particularly exposed to privacy attacks (i.e. attacks that either identify the user or his presence, or that identify the location of the user) since the communication medium is readily available for passive tapping. We show that traditional schemes for anonymous communication in wireless settings, such as masking of MAC addresses and link padding with dummy traffic, are largely ineffective against statistical timing analysis of network traffic. For example, we show how blind source separation allows us to identify the flows from different senders based on the same traces of aggregate packet counts. Similarly, principal component analysis of traces of aggregate packet counts leads to accurate estimations of the number of nodes in a dense setting. We show in our experiments that the location estimation of nodes is accurate (typically significantly below the distance between any two sensors). These results indicate how effective these attacks are in separating traffic from different senders, and so identifying the presence of the senders.

The fact that the proposed schemes require from the sensors only the capability to receive and count 802.11 packets indicates that one should be able to deploy similar schemes for intrusion detection in ad-hoc networks, for example: The ad-hoc nodes could easily collect the data necessary to identify active intruders and to pin-point their location.

The poor performance of link-padding based anonymity protocols in wireless settings is due to a large part to the underlying carrier-sensing based MAC protocols, which perturbs the originally padded traffic, and so renders it susceptible to blind source separation attacks. With privacy of users in mind, it may be time to re-evaluate the use of carrier-sensing based versus scheduling based MAC protocols and how to trade-off privacy versus efficiency in such systems.

REFERENCES

Bahl, P. and Padmanabhan, V. N. 2000. Radar: An in-building rf-based user location and tracking system. In *INFOCOM.* Tel Aviv, Israel, 775–784.

Belouchrani, A., Abed-Meraim, K., Cardoso, J.-F., and Moulines, E. 1997. A blind source separation technique using second order statistics. *IEEE Transactions on Signal Processing 45,* 2, 434–444.

Beresford, A. R. and Stajano, F. 2003. Location privacy in pervasive computing. *IEEE Pervasive Computing 2,* 1, 46–55.

Berthold, O. and Langos, H. 2002. Dummy traffic against long term intersection attacks. In *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*. San Francisco, CA, 110–128.

Berthold, O., Pfitzmann, A., and Standtke, R. 2000. The disadvantages of free MIX routes and how to overcome them. In *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Berkeley, CA, 30–45.

Cai, J., You, H., Lu, B., Pooch, U., and Mi, L. 2005. Whisper - a lightweight anonymous communication mechanism in wireless ad-hoc networks. In *Proc. of Internation Conference on Wireless Networks (ICWN 05)*. Las Vegas, NV, 482–488.

Cardoso, J.-F. 1998. Blind signal separation: Statistical principles. *9,* 10, 2009–2025. Special issue on blind identification and estimation.

Castro, P., Chiu, P., Kremenek, T., and Muntz, R. R. 2001. A probabilistic room location service for wireless networked environments. In *Proc. of the 3rd International Conference on Ubiquitous Computing*. Atlanta, GA, 18–34.

Chaum, D. L. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24,* 2, 84–90.

Comon, P. 1994. Independent component analysis, a new concept? *Signal Process. 36,* 3, 287–314.

Cruces, S. and Cichocki, A. 2003. Combining blind source extraction with joint approximate diagonalization: Thin algorithms for ICA. In *Proc. of the Fourth Symposium on Independent Component Analysis and Blind Signal Separation*. Nara, Japan, 463–468.

Cuellar, J. R., John B. Morris, J., Mulligan, D. K., Peterson, J., and Polk, J. M. 2004. RFC 3693: Geopriv requirements. Available: http://www.ietf.org/rfc/rfc36-93.txt [Accessed July 2005].

Danezis, G. 2004. The traffic analysis of continuous-time mixes. In *Proc. of Privacy Enhancing Technologies Workshop (PET 2004)*. Toronto, Canada, 35–50.

Danezis, G., Dingledine, R., and Mathewson, N. 2003. Mixminion: Design of a type iii anonymous remailer protocol. In *Proc. of the 2003 IEEE Symposium on Security and Privacy*. Oakland, CA, 2–15.

Danezis, G. and Serjantov, A. 2004. Statistical disclosure or intersection attacks on anonymity systems. In *Proc. of 6th Information Hiding Workshop (IH 2004)*. Toronto, Canada, 293–308.

Diaz, C., Seys, S., Claessens, J., and Preneel, B. 2002. Towards measuring anonymity. In *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*. San Francisco, CA, 54–68.

Dingledine, R., Mathewson, N., and Syverson, P. 2004. Tor: The second-generation onion router. In *Proc. of the 13th USENIX Security Symposium*. San Diego, CA, 303–320.

Dusi, M., Gringoli, F., and Salgarelli, L. 2008. A preliminary look at the privacy of ssh tunnels. *Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on*, 1–7.

Elnahrawy, E., Li, X., and Martin, R. P. 2004. The limits of localization using signal strength: A comparative study. In *IEEE SECON 2004*. Santa Clara, California, 406–414.

Gaeta, M. and Lacoume, J.-L. 1990. Source separation without prior knowledge: The maximum likelihood solution. In *Proc. of EUSIPCO'90*. Barcelona, Spain, 621–624.

Goldschlag, D., Reed, M., and Syverson, P. 1999. Onion routing for anonymous and private internet connections. *Communications of the ACM (USA) 42,* 2, 39–41.

Gruteser, M. and Grunwald, D. 2003. Enhancing location privacy in wireless lan through disposable interface identifiers: A quantitative analysis. In *Proc. of the 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*. San Diego, CA, 46–55.

Gulcu, C. and Tsudik, G. 1996. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*. IEEE, 2–16.

He, Z., Yang, L., Liu, J., Lu, Z., He, C., and Shi, Y. 2000. Blind source separation using clustering-based multivariate density estimation algorithm. *IEEE Trans. on Signal Processing 48,* 2, 575–579.

HELSINGIUS, J. 1996. Press release: Johan Helsingius closes his internet remailer. Available: http://www.penet.fi/press-english.html [Accessed June 2001].

HOH, B. AND GRUTESER, M. 2005. Protecting location privacy through path confusion. In *IEEE/CreateNet Intl. Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*. Athens, Greece, 194–205.

HOWARD, J. D. 1998. An analysis of security incidents on the internet 1989-1995. Ph.D. thesis, Pittsburgh, PA, USA.

HUANG, P., FELDMANN, A., AND WILLINGER, W. 2001. A non-instrusive, wavelet-based approach to detecting network performance problems. In *Proc. of Internet Measurement Workshop*. San Francisco, CA, 213–227.

HYARINEN, A., CRISTESCU, R., AND OJA, E. 1999. A fast algorithm for estimating overcomplete ICA bases for image windows. In *Proc. Int. Joint Conf. on Neural Networks*. Washington, DC, 894–899.

HYVARINEN, A. 1999. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks 10,* 3, 626–634.

HYVARINEN, A. AND INKI, M. 2002. Estimating overcomplete independent component bases for image windows. *J. Math. Imaging Vis. 17,* 2, 139–152.

HYVARINEN, A. AND OJA, E. 1997. A fast fixed-point algorithm for independent component analysis. *Neural Comput. 9,* 7, 1483–1492.

ISMAIL GUVENC, CHAOUKI T. ABDALLAH, R. J. AND DEDEOGLU, O. 2003. Enhancements to rss based indoor tracking systems using kalman filters. In *GSPx and International Signal Processing Conference*. Dallas, TX.

JUTTEN, C. AND HERAULT, J. 1991. Blind separation of sources, part 1: An adaptive algorithm based on neuromimetic architecture. *Signal Process. 24,* 1, 1–10.

KESDOGAN, D., EGNER, J., AND BUESCHKES, R. 1998. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proc. of Information Hiding Workshop (IH 1998)*. Portland, OR, 83–98.

KONG, J., GERLA, M., AND HONG, X. 2003. A new set of passive routing attacks in mobile ad hoc networks. In *Proc. of IEEE Military Communications Conference (MILCOM'03)*. Boston, MA, 796–801.

KONG, J. AND HONG, X. 2003. ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC-03)*. Annapolis, MD, 291–302.

LEVINE, B. N., REITER, M. K., WANG, C., AND WRIGHT, M. K. 2004. Timing attacks in low-latency mix-based systems. In *Proc. of Financial Cryptography (FC '04)*. Key West, FL, 251–265.

MALHOTRA, N., KRASNIEWSKI, M., YANG, C., BAGCHI, S., AND CHAPPELL, W. 2005. Location estimation in ad hoc networks with directional antennas. *icdcs 00*, 633–642.

MOELLER, U. AND COTTRELL, L. 2000. Mixmaster Protocol — Version 2. Unfinished draft.

MOLGEDEY, L. AND SCHUSTER, H. G. 1994. Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters 72,* 23 (June), 3634–3637.

NICULESCU, D. AND NATH, B. 2004. Vor base stations for indoor 802.11 positioning. In *Proc. of the 10th Annual International Conference on Mobile Computing and Networking*. Philadelphia, PA, 58–69.

PAREKH, S. 1996. Prospects for remailers - where is anonymity heading on the internet. Available: http://www.firstmonday.dk/issues/issue2/remailers/ [Accessed June 2001].

PARK, K. AND WILLINGER, W. 1999. Self-similar network traffic: An overview.

PHAM, D.-T., GARRAT, P., AND JUTTEN, C. 1992. Separation of a mixture of independent sources through a maximum likelihood approach. In *Proc. of EUSIPCO'92*. Brussels, Belgium, 771–774.

PORRAS, P., SAIDI, H., AND YEGNESWARAN, V. 2007. A multi-perspective analysis of the storm (peacomm) worm. Tech. rep., SRI International, Computer Science Laboratory, Computer Science Laboratory, 333 Ravenswood Avenue, Menlo Park, CA 94025. October.

RAYMOND, J. 2001. Traffic analysis: Protocols, attacks, design issues and open problems. In *Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. LNCS, vol. 2009. Springer-Verlag, 10–29.

REITER, M. K. AND RUBIN, A. D. 1998. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur. 1,* 1, 66–92.

RENNHARD, M. AND PLATTNER, B. 2002. Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*. ACM Press, New York, NY, USA, 91–102.

SERJANTOV, A. AND DANEZIS, G. 2002. Towards an information theoretic metric for anonymity. In *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*. San Francisco, CA, 41–53.

SERJANTOV, A., DINGLEDINE, R., AND SYVERSON, P. 2002. From a trickle to a flood: Active attacks on several mix types. In *Proc. of Information Hiding Workshop (IH 2002)*. Noordwijkerhout, The Netherlands, 36–52.

SHERWOOD, R., BHATTACHARJEE, B., AND SRINIVASAN, A. 2002. p^5: A protocol for scalable anonymous communication. In *Proc. of the 2002 IEEE Symposium on Security and Privacy*. Oakland, CA, 58–70.

TONG, L., WEN LIU, R., SOON, V. C., AND HUANG, Y.-F. 1991. Indeterminacy and identifiability of blind identification. *Circuits and Systems, IEEE Transactions on 38,* 5, 499–509.

VON RICKENBACH, P. AND WATTENHOFER, R. 2004. Gathering correlated data in sensor networks. In *Proc. of the 2004 Joint Workshop on Foundations of Mobile Computing*. Philadelphia, PA, 60–66.

ZHANG, Y., LIU, W., AND LOU, W. 2005. Anonymous communications in mobile ad hoc networks. In *INFOCOM*. 1940–1951.

ZHU, B., WAN, Z., KANKANHALLI, M. S., BAO, F., AND DENG, R. H. 2004. Anonymous secure routing in mobile ad-hoc networks. In *Proc. of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*. Tampa, FL, 102–108.

ZHU, Y. AND BETTATI, R. 2007. Compromising confidentiality in wireless network using sensors with limited information. In *IEEE International Conference on Distributed Computing Systems (ICDCS 2007)*. Toronto, CANADA.

ZHU, Y., FU, X., GRAHAM, B., BETTATI, R., AND ZHAO, W. 2004. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*. LNCS, vol. 3424. 207–225.

Received July 2008