Strengthening Forensic Investigations of Child Pornography on P2P Networks

Marc Liberatore* Brian Neil Levine* Clay Shields△

*Dept. of Computer Science, Univ. of Massachusetts Amherst, {*liberato,brian*}*@cs.umass.edu*

△ Dept. of Computer Science, Georgetown Univ., *clay@cs.georgetown.edu*

ABSTRACT

Measurements of the Internet for law enforcement purposes must be forensically valid. We examine the problems inherent in using various network- and application-level identifiers in the context of forensic measurement, as exemplified in the policing of peer-to-peer file sharing networks for sexually exploitative imagery of children (child pornography, or CP). First, we present a characterization of measurements of these networks, including large-scale measurements performed in the law enforcement context. We then show how the identifiers in these measurements can be unreliable, and propose the tagging *of remote machines. Our proposed tagging method marks remote machines by providing them with application- or system-level data that is valid, but covertly has meaning to investigators. This tagging allows investigators to link network observations with physical evidence in a legal, forensically strong, and valid manner. We present a detailed model and analysis of our method, show how tagging can be used in several specific applications, discuss the general applicability of our method, and detail why the tags are strong evidence of criminal intent and participation in a crime.*

1. INTRODUCTION

The most popular resource for the criminal acquisition and distribution of images and video of child pornography is peer-to-peer (p2p) networks, including BitTorrent and Gnutella.[1] Law enforcement (LE) have both an easy and difficult time policing these networks. On the one hand, it is easy to identify millions of IP address trafficking in known child pornography (CP), as we demonstrate in Section 3. On the other hand, this success falls short in several ways. IP addresses and application identifiers are the foundation of all current criminal network investigations. Yet, IP addresses do not distinguish multiple physical machines behind a NAT box. Similarly, when a mobile user moves among many IP addresses, it is difficult to link such activities to that single user.

The value of evidence is the critical difference between forensics and related security research in incident response and privacy; moreover, methods and legal procedures for collecting data differentiate network forensics from simple network measurement. Making guesses or inferences may be suitable for discovering the limits of privacy or advancing incident response, and it may generate an investigative lead, but it will not advance a legal case. IP addresses are an excellent example of the low-value evidence that is in standard use by criminal forensic practitioners — a recent, scathing report by the National Academy of Sciences [18] calls for a scientific overhaul of forensics, including digital forensics.

In this paper, we introduce new techniques that draw a bright line between the measurement or surveillance of these networks and collection of forensically valid evidence from them. Validating the evidence collected during a network investigation is difficult because remote users do not maintain a unique and unmodifiable identifier that can be recovered upon seizure of the machine with a warrant. We propose a novel method of subtly tagging a remote computer over the network to create such an identifier. Our approach is an advance over previous methods of gathering information about a remote computer that rely on statistical characterizations, including clock skew [14] or radiometrics [2]. These past characterizations vary with environmental factors such as temperature or attack [6], leading to both false positives and false negatives, and crucially, lack the ability to link together sequential observation by independent observers. Moreover, we detail why our approach, which is akin to marking bills, is legal.

For this work, we built a system to gather evidence of possession of child pornography on a p2p network, and it is in use by law enforcement in 50 U.S. states who have gathered data for us over a five-month period of time. The system's data has been used to obtain hundreds of warrants, which has the standard of probable cause, and we show why it is limited to that case. We characterize these measurements in order to motivate our tagging techniques. When found on a machine during

[1]Past studies have found that 28% of possessors of child pornograhpy had images of children younger than 3 years old; and that 16% of investigations of CP possession ended with discovery of persons who directly victimized children [21].

a forensic exam, our tags are strong evidence that the machine corresponds to observations of a peer made over the network. Unlike statistical characterization methods, our method has very strong privacy properties: the results can be recovered by investigators only after a search warrant is obtained from a judge, and tags observed by third parties are meaningless. Our careful design and analysis also demonstrates that false positive probabilities can be driven to near zero. The tradeoff is that our challenge is to make sure they are retained by the target, to be later discovered during an examination.

Specifically, we make several contributions:

- We present the results of five months of investigations into Internet crime, performed with the help of law enforcement. Our focus is on sharing of child pornography on peer-to-peer networks. We show that identifying those trading in CP is simple, and that such trafficking is unfortunately common, with millions of distinct IP addresses participating.
- We analyze the strength of digital evidence relied on by investigators in these crimes, demonstrating that these techniques are insufficient beyond probable cause for stationary IP addresses. Moreover, such techniques are insufficient for demonstrating intent and do not work well for mobile users.
- We propose a novel method of strengthening network investigations of criminal activity called tagging. We analyze its design and demonstrate that the chances of false positives can be made insignificant with relatively low overhead.
- Finally, we will show how these tags can be used in several specific applications (including BitTorrent and DNS) discuss the general applicability of our method, and detail why the tags are strong evidence of intent and participation in a crime.

We begin with a statement of the problem, and a description of the relevant attacker models. We then present an empirical analysis of measurements collected on the Gnutella network and among BitTorrent peers, with a focus on the evidentiary value of these measurements. We then present tagging, our proposed mechanism for improving such evidence. We follow with a discussion of the issues of law and the forensic context — readers unfamiliar with this topic may wish to start with Section 5 before reading the main body of this paper. This work is a significant extension to our prior work [15].

2. PROBLEM AND ATTACKER MODEL

In this section, we briefly discuss the motivating problem for our work: network investigations of criminal activity, and forensic validation of the evidence of these crimes. We briefly discuss the investigative process, and the problem that forensic validation poses. We also present the relevant attacker models. In later sections, we present a set of characterizations that empirically show the scope and importance of the problems we identify here and present a more exact description of our proposed solution.

2.1 Problem Statement

When investigating Internet crimes such as trafficking in child pornography, the general approach of law enforcement is as follows: an investigator issues queries for likely child pornography (CP) and gathers results. Some results are chosen for further investigation, and the investigator uses the court system to compel an ISP to reveal a physical location that corresponds to the potential source of network traffic that provided the query results. The location is searched, a machine and its accompanying storage media is physically seized, and the media is examined for evidence of the possession or distribution of CP. We describe the various legal restrictions that US investigators operate under in Section 5; these restrictions influence our design decisions.

Our interest lies in effectively identifying the correct end system. In particular, can investigators strongly link network measurements with user behavior and intent? Our goals are twofold: First, we aim to evaluate the quality of the procedures currently used to perform these measurements. We present results of our evaluation in Section 3; in summary, we show that the current procedure of using IP addresses and certain protocol-specific identifiers can fail to identify a unique system in many circumstances.

Thus, our second goal is to improve the quality of evidence and the range of tools available to investigators. In particular, we propose the use of *tagging*. The general mechanism of tagging is to insert per-observation bit patterns — tags — into stable storage media belonging to a suspect during the course of the network-based investigation. These tags can later be recovered from the storage media following a legal seizure, not unlike marked bills might be recovered after an undercover transaction involving stolen property or illegal drugs. These tags can then be used to both link the observations with the media, and to show a pattern of behavior, and thus, intent, on the part of the suspect.

2.2 Attacker Models

We have two actors in our scenario, and we define assumptions for both. We define the **investigator's attacker model** as follows: An investigator of a given p2p system: *(i)* seeks to identify users of the protocol in possession of, or distributing, child pornography — typically, an IP address within their jurisdiction is the endpoint of the network investigation; *(ii)* must work within the protocol, and can not rely upon policy violations, criminal activity, or privilege escalation to gather

evidence; *(iii)* may consider indirect evidence to generate leads, but must have direct evidence to succeed (i.e., seeks a direct network-level connection to a remote user's system). A **criminal's attacker model** and goals are markedly different. A criminal: *(i)* will actively attempt to acquire new CP; *(ii)* may redistribute and advertise possession of CP; *(iii)* may actively manipulate the protocol, violate policy, or engage post-facto anti-forensics in order to hide their activities. Clearly, a criminal actively attempting to hide their trail will be harder to catch. As noted above, there are a wide range of societally acceptable outcomes, depending upon the interactions of these investigator and criminal knowledge, permissible conduct for investigators, and tolerance of criminality.

Given that we allow the criminal to erase evidence from their own machine, why do we expect our techniques to work at all? There are several answers. First, unlike most mechanisms in security, most forensic mechanisms are not subject to *catastrophic failure*: even if one person can and does erase evidence, that does not imply that everyone can do it, nor does it mean that one person can erase it for everyone else. And it is still worth catching those that do not erase evidence. In contrast, if there exists a security exploit in Windows, then one user can comprise every Internet-accessible Windows machine.

Secondly, these crimes are not committed by persons with great savvy — the quantitative proof is the measurement we present in Section 3: we identified 4,674,419 IP addresses sharing known images of known child pornography. These images were based on a database of known hash values and any one of these persons could have flipped a single bit of their image and not be caught, yet more than four million did not.

Finally, our methods are designed to tag mechanisms that improve performance when left enabled. While application developers are savvy, we don't believe that they have an interest in protecting persons trafficking in child pornography and are not willing to degrade their own network performance to do so. While the RIAA and MPAA are active in copyright enforcement, those civil torts require a much lower standard of evidence (a preponderance of evidence) than criminal prosecutions (requiring evidence beyond a reasonable doubt); our tagging techniques would be overkill. Moreover, since the tags are impossible to trace back to investigators, these developers will have no sense of whether they are being used. They would need to turn off all writes to disk by the entire application and operating system to ensure they are not, which is possible but an unlikely scenario.

3. AN EMPIRICAL STUDY OF CRIME AND IDENTIFIERS ON P2P NETWORKS

Our goal in this section is to demonstrate the low value of IP addresses and application-level IDs used in network investigations. This low value is the result of widespread use of DHCP and mobile networking as well as the presence of botnets, and our results in this section provide some quantification of this problem. Our empirical results are based on our five-month measurement study of child pornography file sharing on p2p networks. Our data was collected using a tool we wrote for monitoring and investigating sharing of child pornography on Gnutella networks. As a consequence of our efforts, our tool *RoundUp* has been adopted recently as the standard for for p2p investigations by the US Internet Crimes Against Children (ICAC) Task Force; ICAC is a collection of law enforcement agencies from all 50 states. Data from each detective's actual investigation of child pornography trafficking on Gnutella are stored in a centralized system under police control. We analyzed an anonymized version of the data.

From Oct 1, 2009 to March 1, 2010, our system collected measurements of 4.6 million IP addresses using 799,557 GUIDs. A GUID is Gnutella's application-level identifier that is chosen at random during installation. In all, almost 19,000 distinct child pornography files were observed by our law enforcement (LE) colleagues on the Gnutella network. These files are identified by hash value (not filename), and they are checked manually at least once by law enforcement.

While Gnutella is not the most popular p2p program, our statistics above show it is popular with child pornography (CP) file sharers. To ensure our results are true for other p2p networks, in this section we also characterize BitTorrent measurement data collected by Menasche et al. [16]. While our study focused on CP, the data collected by Menasche et al. measured non-contraband content on BitTorrent. That study measured torrent activity between August 2008 to March 2009, including the IP address and BitTorrent PeerID of participants. Our conclusions about evidence on Gnutella GUIIDs and IP addresses are validated by observing the same results for BitTorrent PeerIDs and IP addresses, as we describe below. We begin with a summary of the success and limitations of the current investigative approach.

3.1 The Current Investigative Approach

As we detail in Section 5, data collected during network investigations are used only as a stepping stone to obtain legal authority to search a physical location for evidence of a crime.

Success of existing approach. In the course of a search, storage media are examined for CP and evidence of intent, such as cached search terms. The presence of this evidence is used to create a case for criminal possession of CP. This methodology has been used successfully in hundreds of investigations in the US.

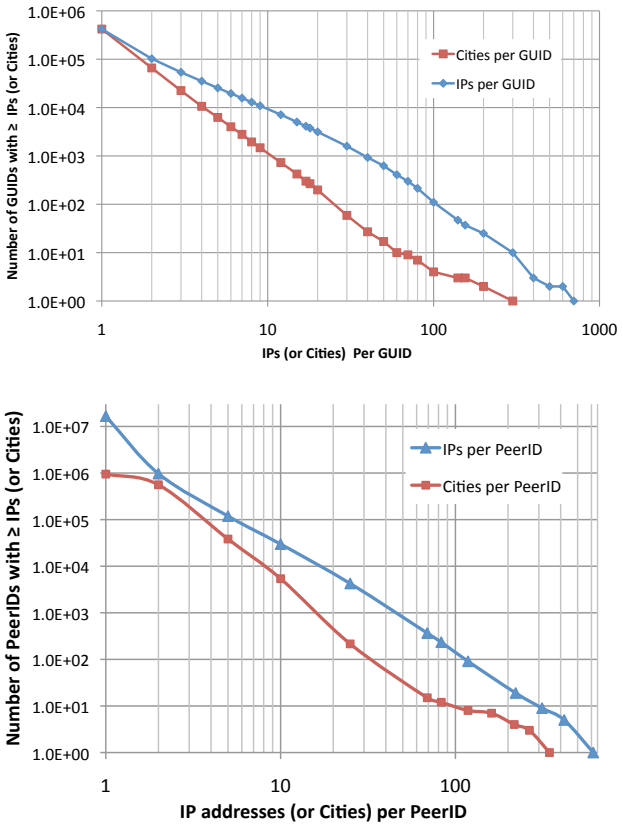

Figure 1: CCDF of the number of cities and IPs each ID was associated with. (Top) Gnutella (Bottom) BitTorrent

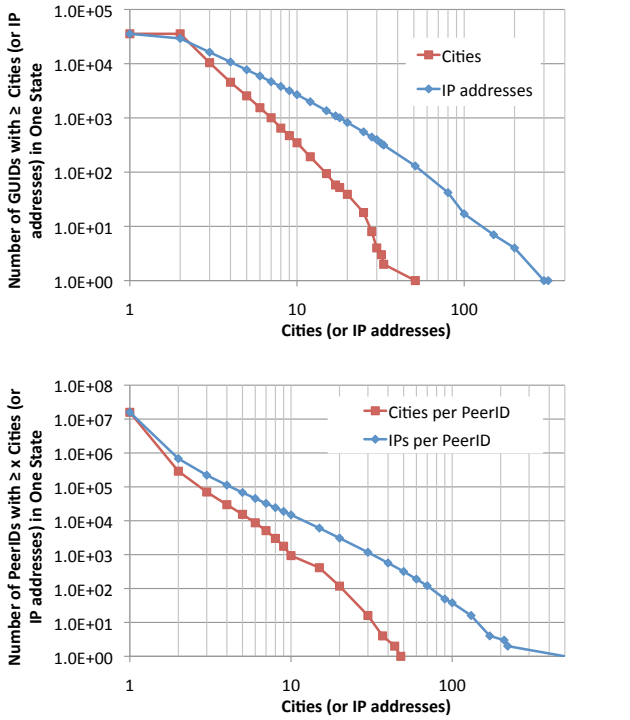

Figure 2: CCDF of cities and IPs each ID was associated with, given that all IPs used by the ID map to cities in a small geographical region. (Top) Gnutella (Bottom) BitTorrent

Limitations of existing approach. There are three key limitations of the current approach. *(i)* When network investigations lead to search warrants, it is evidence found from the search that is used as the basis for criminal prosecution. In fact, there is often no connection between what is observed on the network and what is found in the search: users may delete files, or install new client software. As a result, it is challenging to prosecute for distribution of CP in the case that some CP is found during a warrant-based search, but it is not the same files that were requested and downloaded by LE from that peer. *(ii)* Positively identifying a seized machine as the same one that was investigated remotely is a challenge. Circumstances such as network address translation, DHCP lease times, and mobile interfaces can cause a mismatch. Similarly, many file sharing applications do not provide a stable unique identifier for the user. For example, BitTorrent does not require fixed *peer identifiers* (PeerIDs), and Gnutella does not ensure each client's self-assigned *Globally Unique ID* (GUID) is, in fact, globally unique. *(iii)* Intent is part of the definition of criminal CP possession and distribution. Intent can be demonstrated legally in several ways [9]. Unfortunately, a key form of evidence of intent on p2p networks — sharing on the network over a long period of time — cannot be demonstrated in court. An even greater challenge is to definitively show that the same person is responsible for using multiple GUIDs or multiple IPs over time, particularly over open APs in cafes or campuses.

It can be challenging to find the evidence of a crime: The subject of investigation might hide it within the system with encryption or steganography, or might keep the material on a removable storage device that is physically concealed. In cases where the investigator does not locate the material that was seen as being available, it is not clear whether the wrong system was seized or if the material simply hasn't been discovered. A reliable indication that the correct system was seized, as we propose in the Section 4, can help resolve this dilemma and the others above.

3.2 Identity and Intent in P2P Networks

Data collected during network investigation can be used for two distinct, dependent purposes — but are not currently. First, measurements can establish the *identity* of a suspect. By identity, we mean a network or application identifier that can ultimately be linked

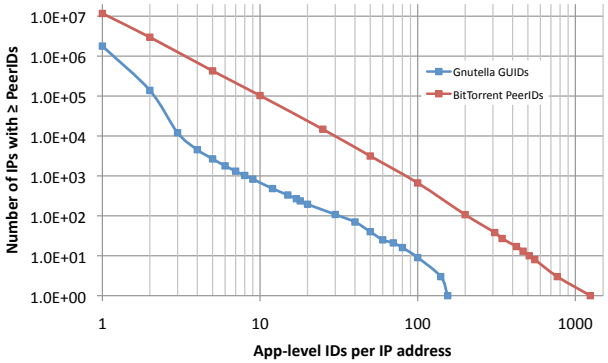

Figure 3: CCDF of the number of IDs observed per IP address.

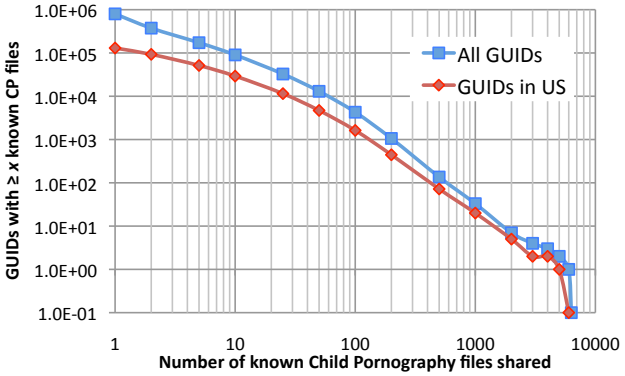

Figure 4: CCDF of the number of distinct pieces of known CP possessed per ID.

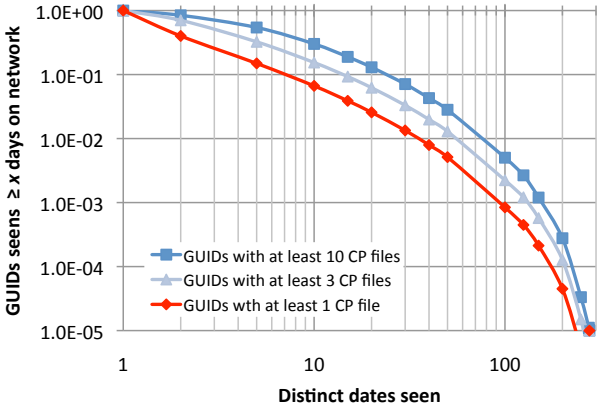

Figure 5: CCDF of the distinct days an ID was seen online.

to an individual at a given time and place. Second, measurements can be used to establish *intent* to commit a crime: a user might accidentally download a single CP file, but if they have a large and growing collection over the course of months, it is highly unlikely to be accidental. Discovering *intent* requires consistency of identifiers over time, a property that we observe does not always hold.

Generally, current investigators instead use network identifiers such as IP addresses only to obtain a search warrant. As we discuss in Section 5, IP addresses are generally regarded as meeting the standard of probable cause — good enough for a search, but not convincing enough for prosecution. As we show here, IP addresses can vary significantly over time for other, fixed identifiers, so this level of skepticism is warranted. GUIDs and other application-level identifiers suffer from similar credibility problems.

The top plot in Fig. 1 demonstrates how application IDs can fail as a unique identifier. The figure plots the number of IP addresses associated with each ID in the data. For Gnutella, this consists of GUIDs that have been identified as trafficking in child pornography (via file SHA1 values) by our law enforcement partners. In our data, 9,315 GUIDs were each associated with 10 or more IP addresses. A separate line plots the same data by geographic location as mapped by the commercial MaxMind geolocation service, which maps IPs to cities. The bottom plot shows similar statistics in BitTorrent.

One particular GUID was observed in 329 cities around the world using 398 IP addresses. We found this GUID was sharing exactly one file. This file (identified by hash value) was found throughout the network with many different filenames. We assert this GUID is actually a botnet that responds to queries for any `x` on the network with `x.mpg`, always sharing the same content, likely malware. The existence of this GUID shows the difficulty in assuming that GUIDs are unique identifiers for corroborating an investigation with a seized machine, as this one appears to be shared by many users. Similarly, there are more than 10 PeerIDs that map to IP addresses found in more than 100 cities. These observations point to a weakness in such IDs that may skew all data points collected, but in a non-obvious way. For researchers, this discrepancy may be overlooked, but in the legal context it is troublesome.

Another problem is posed by mobile users. Fig. 2 isolates IDs that report from 2 or more IP addresses all located one state or region of a country. Since these IDs have IP addresses that map to only one geographic area, we assert that it is most likely one real user, as botnets and misconfigurations are unlikely to be contained to a geographical area. It is unclear to use if these users are sharing their ID with friends, or driving around using various open WiFi access points. This data suggests either that these identifiers are weak, or that users actively move around to avoid detection; either motivates our tagging solution.

Fig. 3 demonstrates the related problem of relying on a peer's IP address as a unique identifier of a specific computer. The figure plots the number of IDs observed

per IP address. For example, 2,663 IP addresses were each linked to at least 5 different GUIDs in our database. It is not clear whether the GUIDs represent five or more different users behind one NAT box or if one user is responsible for all activity from that IP using five GUIDs. In BitTorrent, this problem is worse since PeerIDs can be generated per torrent, which makes it impossible to link the download of a particular torrent with a specific installation on a computer, or even a specific user on a single computer.

Figs. 4 and 5 show statistics on the scope of problem, demonstrating that many users are observed repeatedly over long periods of time. Users with larger collections (some have thousands of files) tend to stay on the network longer. With current techniques it is unknown if some of the GUIDs observed once are actually the same user.

4. REMOTE DEVICE TAGGING

Measurements that attempt to tie observations together using information provided by an application-level protocol may be flawed. Identifiers turn out to be neither unique nor consistent identifiers in all cases. We propose a novel mechanism to remotely tag a device that is under investigation. We begin by presenting the tagging process and follow with an analytical model of the process. We then show several tagging opportunities that exist in BitTorrent software such as Vuze, as well as opportunities in the DNS system. We end with a discussion of the increase in investigative power that tagging provides.

4.1 The Tagging Process

We propose the tagging of remote machines by investigators, to leave a record of an observation on the remote machine for later recovery during warranted search. We envision the general process as follows:

- Investigators discover a *vector* for tags: we define a vector as a set of bits embedded in a protocol that can be set within the bounds of the protocol by the investigator and sent to a remote machine under investigation. Further, these bits or some determinate function of them must be stored by the remote machine on non-volatile media. As we detail in Section 4.3, BitTorrent peers will ask each other their application name and peer ID, and there are minimal restrictions on these values; these values may be stored in a file at the target. These fields can function as tags to uniquely identify the remote machine. No unauthorized access to the target's machine is required; tags are inserted through normal functioning of a system.
- When directly connecting to a remote machine during an investigation, investigators use an appropriate vector to tag the machine. Tags are selected in such a way that their meaning is not obvious and to minimize the likelihood of collision. The investigator records the tags used to so that they can be validated when recovered.

  One method of selecting tags is to take a hash value of text representing specific details of the investigation. This hash can be provided as part of the search warrant request to a judge to commit the investigator to one or more values. Law enforcement organizations could release publicly the root of a Merkle tree of all hash values used for a specific time period.
- Upon issuance of a warrant by a magistrate, investigators seize a machine and look for known tags on it. These tags may be found in the expected place, or recovery may require more advanced forensic techniques such as file carving. Tags that are recovered from a seized machine validate that it is a specific system that was investigated over the network. Because recovery requires a magistrate-approved warrant and because the meaning of the tags is hidden, our approach has robust privacy properties.

There are two ways in which retrieving tags from a machine can fail. False negatives occur when tags that were placed by an investigator are unrecoverable, due to deliberate user action, log rotation, cache eviction, and so on. In these cases, the tags will not be available as evidence. False positives occur when investigators recover tags that they did not actually place — in essence, they recover incorrect evidence. From a legal perspective, this problem is much more serious. We examine this problem in detail below.

4.2 Modeling False Positives

Since tagging assumes the machine is recovered with a search warrant, it only makes sense to leverage tags when they are clear evidence. In other words, they are not useful as probable cause since the investigators must have that to get the search warrant. Therefore, how certain are tags as evidence, in other words, what is their false positive (FP) rate? False positives occur when a machine that was never tagged appears to investigators to store known tags. The analysis we carry out to determine false positives applies equally to the scenario where an adversary places tags on a third-party victim's machine in an attempt to frame the victim; we assume the attacker don't know which tags are used by investigators, an easy assumption to satisfy in practice.

Model assumptions. Assume investigators tag target machines with an n-bit tag each time they are observed on the network (called a *session*), and they keep a database of T entries. The number of entries is exactly the space of all tags that have ever been (or will be) assigned for a distinct taggable event. Each entry will

include other essential information about the investigation: the name of the investigator, the date, the tagged IP address, etc. Here we set $T = 2^{\frac{n}{f}}$, and therefore the chance that a recovered tag (that was not placed by investigators) is a false positive is $T/2^n$. We assume $f > 1$, since when $f = 1$ the chances of a false positive is 1. We discuss how f affects performance below, and in fact it is one of two variables that must be decided ahead of time.

In the analysis below, we assume a log file is recovered from a seized machine, and that (unbeknownst to investigators) the machine has never been tagged. We let L be the number of candidate tags that are discovered.

Large tags. The simple case for tagging is when n is very large; in that case, it is easy to make it improbable that a tag found on seized machine falsely matches a tag in the database. The chances that one or more of L candidate tags match stored values in the database is

$$\begin{aligned} Pr\{\text{False positive}\} &= 1 - Pr\{\text{no candidate matches}\} \\ &= 1 - (1 - \frac{2^{n/f}}{2^n})^L \end{aligned} \quad (1)$$

However, the maximum value of n is not chosen by the investigator; it is a given from the exact software that is being leveraged for tagging. For example, if $L = 2000$ and $n \leq 32$, the chances of a false positive is greater than 3%, which is most likely too high.

Unfortunately, in some situations, the tag size n is limited and we require a low false positive rate. For example, in Section 4.3 we discuss the use of the varying portion of a CIDR block as tags stored in a BitTorrent peer cache. To overcome this limitation, we have the investigator generate and use many *subtags* per session. Subtags are generated by splitting n-bit tags into k equal-length parts. An investigator then subtags a remote machine k times in a session.

There are two scenarios that we must consider.

- (Case A) The subtags are stored on the target machine in a *preserved order* that can be recovered.
- (Case B) The subtags are stored on the target machine in an *unordered set* that prevents ordered recovery. For the unordered case, we offer two solutions: (B1) tagging the target k times each session; and (B2) allocating space in the subtags to denote the order for recovery of the full tag, which we show below is a better solution.

We derive the false positive rates of the three approaches below and then compare their performance.

Case A: Order Preserved: Concatenated subtags. In this case, we assume subtags are written to a sequential log file, and that investigators can reconstruct the original tag by assembling subtags in the order they are recovered from the log file. It may be that other data is inserted into the target's log between subtags, which can result in false positives. Here, we model the most conservative case: we show the number of false positives given that none of the L candidate subtags were placed by investigators.

When the machine is recovered, the investigator will accept the machine as tagged only if k of the L subtags, when concatenated, appear in the database of T assigned tags. The problem is that investigators must try every combination of $\binom{L}{k}$ found, which creates a large number of potential false positives. Here, $T = 2^{\frac{n}{f}}$ as before. The false positive rate of the concatenated tags is

$$\begin{aligned} Pr\{\text{False positive}\} &= 1 - Pr\{\text{no full tag matches}\} \\ &\leq 1 - \left(1 - \binom{L}{k}\frac{1}{2^n}\right)^{2^{\frac{n}{f}}} \end{aligned} \quad (2)$$

Note that this is a conservative upper bound, not an equality, as we have elided the inclusion-exclusion terms.

Case B1: Unpreserved Order: Multiple subtags. In this case, the target machine does not store tags in a preserved order. In this solution to the problem, we have investigators tag the machine with k subtags that share the same database. Therefore, there is a limit of $T = 2^{\frac{2}{fk}}$ tags that can be assigned.

The false positive rate for the case of k subtags of $\lfloor n/k \rfloor$-bits each is given by a Binomial distribution.

$$\begin{aligned} Pr\{\text{F.P.}\} &= Pr\{k \text{ or more of } L \text{ subtags match}\} \\ &= 1 - \sum_{i=0}^{k-1} \binom{L}{i} (2^{\frac{n}{fk} - \frac{n}{k}})^i (1 - (2^{\frac{n}{fk} - \frac{n}{k}}))^{L-i} \end{aligned} \quad (3)$$

Eq. 3 quantifies the tradeoff between using one tag of n bits and k subtags of one bit each, and all cases in between.

Case B2: Unpreserved Order: labeled subtags. When the tagged machine stores the subtags in an unordered set, a better solution is to give each subtag its own database; to do that, we need to reserve $\log_2 k$ bits in each subtag to denote which sub-database it is in. In that case, each subtag has length $r = \lfloor n/k \rfloor - \lceil \log_2 k \rceil$ bits, and we have $T = 2^{rk/f}$ tags possible. To determine the false positive probability, we assume that the L candidate tags are equally divided among the k subtag databases. Therefore, there are $(L/k)^k$ candidate full tags to evaluate; each must not match an assigned value.

In this case, the false positive probability is modification of Eq. 1:

$$\begin{aligned} Pr\{\text{F.P.}\} &= 1 - Pr\{\text{none of the } (\tfrac{L}{k})^k \text{ subtags match}\} \\ &= 1 - \left(1 - \frac{2^{rk/f}}{2^{rk}}\right)^{(\frac{L}{k})^k} \end{aligned} \quad (4)$$

As we discuss in Section 4.3.2, specific ranges of IP addresses such as CIDR blocks can be used as tags. In

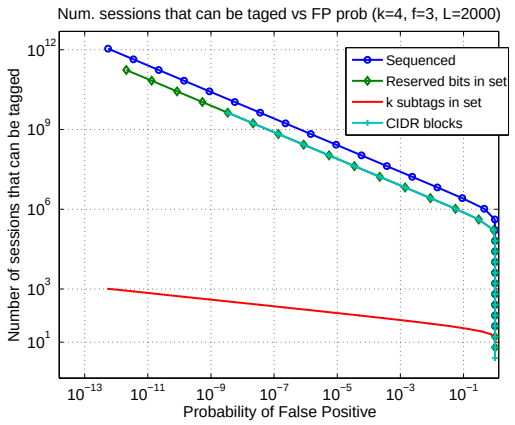

Figure 6: The relationship between false positive rate and sessions that can be tagged.

this case, the fixed prefix of the CIDR block serves in place of the $\log_2 k$ bits that would otherwise be reserved to denote the sub-database. The no-cost nature of these prefix bits explains the improved performance of this method in the comparison below.

4.2.1 *Sessions Taggable by Each Method*

To compare these three methods, we assume the investigator is given the maximum length of each subtag, has a maximum value of L, and has a desired FP rate. Her job is to select f and k such that the false positive rate is achieved and the number of sessions that can be tagged is maximized. In general, higher f and k values lower the FP rate but reduce the number of sessions that can be tagged. We assume the goal is to minimize k, since if more tags are required to be stored, it is more likely that in general that the tags may be removed by standard operation of the machine.

Accordingly, we evaluate three questions: *(i)* For a desired FP rate, what is the minimal value of f? *(ii)* How does the number of sessions that can be tagged vary with f? *(iii)* What is an acceptable FP rate?

We explore these questions in several ways. In all graphs we let $k = 4, f = 3$, and $L = 2000$ as examples. First, we show a quantitative comparison of the false positive rates for the three techniques in Fig 7. In all cases, the probability of a false positive decreases exponentially with n and is less than 1 in 10^6 when subtags are less than 22 bits.

If the same false positive rate is desired when allowable subtags are smaller, then we can increase k or f . How should an investigator set k and f? We offer the following simple algorithm. (1) Select a desired false positive rate ρ. (2) Set $k = 1$ and let n/k be the subtag length. (3) Calculate f and then the maximum number of taggable sessions, T, using the following formulas. (4) If T is too small, increase k and goto Step 2.

We let ρ be the desired FP rate. For Case A, we solve for T and a minimal value of f as

$$f = n/\log_2 \left(\frac{\log(1-\rho)}{\log\left(1 - \frac{\binom{L}{k}}{2^n}\right)} \right) \tag{5}$$

since $T = 2^{n/f}$,

$$T = (\log(1-\rho))/(\log\left(1 - \frac{\binom{L}{k}}{2^n}\right)) \tag{6}$$

For Case B2, we have

$$f = rk/\left(\log_2(1-(1-\rho)^{1/(\frac{L}{k})^k}) + rk\right) \tag{7}$$

since $T = 2^{rk/f}$,

$$T = (1-(1-\rho)^{1/(\frac{L}{k})^k})2^{rk} \tag{8}$$

The remaining question is what level of FP is appropriate? Historically, law enforcement around the US have made about $A = 2000$ arrests per year. We use this number as an example, however in practice, we can keep one set of tables per version of each file sharing client. To set ρ, we assume that all the arrests are mistakes, and let $\rho = 0.1A$ such that the expected number of arrests that have a false positive tag is 0.1. If a more conservative estimate is desired, Chernoff bounds can be used to ensure that even values above the expected mean occur with very low probability.

4.3 Available Covert Channels

For tagging to be practicable and of maximum value, several conditions must hold. First, a machine under investigation must be able to receive data from a remote source. Second, that data must be stored in a fashion that can be retrieved by investigators if the machine is physically seized. Third, investigators must be able to manipulate this data in such a way that it is specific to a single investigation — re-using tags dilutes or nullifies their evidentiary value, violating the assumptions we make in our security analysis. The information will only be recovered when legal authorization by means of a search warrant so allows. Here, we discuss the general manner by which such opportunities can be found and present several tagging channels

4.3.1 *Discovering available channels*

In our experience, tags ultimately reside in one of two places: log (or audit) files, and cache files. In many applications and on many operating systems, a variety of log files record both regular and exceptional events. Log files exist for audit and debugging reasons, and as such often include most of all of the salient details of triggering events. Cache files exist to improve performance or reliability of systems. For example, most p2p

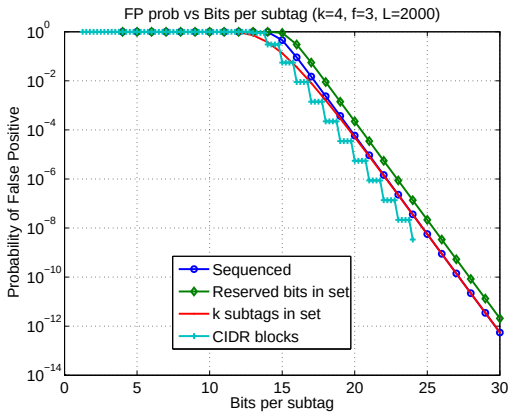

Figure 7: The false positive rate as the number of bits per subtag increases.

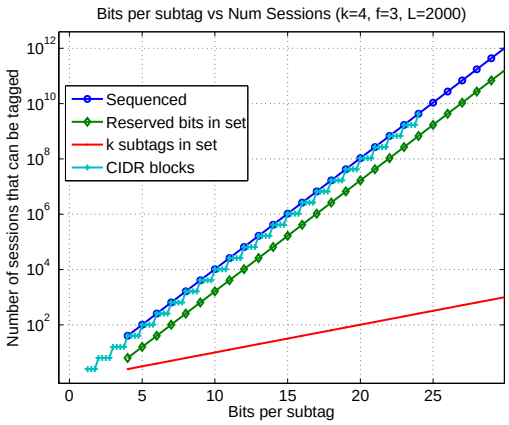

Figure 8: The no. of sessions that can be tagged as the number of bits per subtag increases.

applications store data about remote peers for several purposes: to allow for decentralized operation and bootstrapping; to enable efficient load distribution; or to enable optimizations such as tit-for-tat (or the outright banning of other peers). In both cases, the removal of the taggable files would be detrimental to the performance of the system that creates them, and therefore these are the best candidates, as we discuss later. For example, turning off DNS caching will definitely degrade performance.

There are several ways in which tagging opportunities can be discovered. We used a manual, ad hoc process to discover the tagging opportunities we describe below. We conjecture that both static and dynamic analysis techniques could be applied to an application to find candidate fields for tagging and we believe this is an interesting problem for future work. For example, such techniques could tag data read from network sockets, and calculate (or observe) when the data is written to disk, reporting that fact to an investigator for further verification of its suitability.

4.3.2 *Specific tagging vectors*

Here we give three specific examples of tagging vectors that currently exist in the regular functioning of p2p file sharing software, as well as mentioning other possible opportunities. These opportunities range across the application and operating system, and include both caches and log files. We present these opportunities as proofs-of-concept, and not as fully developed tagging systems. A limitation of our work is that we do not evaluate the churn of these systems quantitatively.

Peer caches. In BitTorrent, peers may actively download and upload a torrent for long periods of time. Files are large, and the culture of BitTorrent users is such that continuing to provide upload bandwidth ("seeding") a torrent is encouraged, while disconnecting immediately after finishing your own download, possibly while throttling your own uploads ("leaching") is discouraged. To maintain state for these active torrents across application restarts and machine power-offs or suspensions, most BitTorrent clients write relevant information to a cache file of some sort. Hence, if these logs are turned off, the performance of BitTorrent will be worse as clients would have to re-discover all peers after application restarts. As we discussed in Section 2.2, removing this functionality from the program is a poor defense.

The μTorrent client stores, per-user-account and per-torrent, the IP addresses and ports of remote peers sharing that torrent. (The same client, rebranded, is also the BitTorrent client distributed by BitTorrent, Inc.) These IPs and ports are stored in a file named `resume.dat`, which is a bencoded[2] dictionary (associative array). This dictionary is keyed by the each active torrent's infohash. An infohash uniquely identifies the content of a torrent. Each value associated with a torrent's infohash is another dictionary. In this dictionary, the key "peers6" encodes 128-bit IPv6 addresses and 16-bit ports of peers. IPv4 addresses are encoded as backwards-compatible IPv6 addresses. Crucially, these addresses and ports can be provided not just from the tracker, but from other peers through the peer-exchange extensions to the BitTorrent protocol. In our observations, these values need not represent reachable or even valid peers to be entered into the peer cache, presumably because well-behaved BitTorrent clients may ignore incoming connections.

In a similar fashion, Vuze stores, per-user-account, a cache file for each active torrent, named `<infohash>.dat`. These bencoded dictionaries contain a key explicitly describing the "tracker_cache", including entries for "tracker_peers" formatted as follows:

[2]Bencoding is a data serialization technique specified by the BitTorrent protocol.

```
[ {  'ip': '83.253.52.14',
     'port': 6886,
     'prot': 1,
     'src': 'Tracker'},
  {  'ip': '87.7.101.196',
     'port': 54650,
     'prot': 1,
     'src': 'PeerExchange'}, ...
]
```

Here, even the source of the remote peer is listed, so we can exclude all non-"PeerExchanged" addresses from consideration when recovering tags, eliminating a potential source of false positives.

In both cases, the address and port serve as a tag, though we have observed that the order of the entries in the peer cache is not preserved in either case. Investigators can use CIDR blocks of address space as tags for IPv4, and equivalent allocation mechanisms in IPv6. Investigators can rent small blocks of address space from many different ISPs to prevent any particular address range from appearing as obviously enforcement-related.

As currently implemented, neither peer caching mechanism requires the investigator to answer BitTorrent protocol messages sent to the addresses that may be stored, so the investigator can insert tags chosen from the IPv6 or v4 address space as appropriate through the peer exchange mechanism. If the implementation were changed to require these remote tags to be valid, the investigator could limit the tags to addresses under their control, running appropriately modified p2p software.

DNS cache entries. By default, μTorrent performs reverse DNS lookups on peer IPs once it has connected to them, and Vuze can be configured to do likewise. In fact, many p2p applications include this feature. By performing this lookup, a p2p application likely causes the host operating system to cache the DNS entry in question. The existence of this entry, and possibly the textual value of the DNS entry itself, serve as a tag.

Other tagging opportunities. Depending upon the p2p system and implementation, there are other targets of opportunity. An obvious potential target is the payload data being transmitted by the p2p users. Most such systems use some sort of hashing scheme to prevent the deliberate poisoning of exchanges with bad data. Still, if this data is ever written to disk, for example as either as a temporary file or through the VM system, traces of it may persist and be recoverable through standard forensic means. Relative to the tag sizes derived earlier, the immense size of even a relatively small file system allocation unit could provide a definitive tagging opportunity.

Vuze log files. Vuze (formerly Azureus) is among the most popular of BitTorrent clients. It creates user-account-specific log files for several purposes, including debugging. One such log file, named `debug.log` (or a variation thereof, when rotated), contains at least two obvious candidates for tagging.

The first arises from the evolving nature of the BitTorrent protocol specification. In particular, the format by which BitTorrent peers are identified, the *peerID*, is not fully specified. To aid developers in discovering and naming new BitTorrent implementations, peerIDs in unrecognized formats are saved to the log. As these peerIDs can be arbitrarily chosen, 120-bit strings, they present an ideal tagging target. Similarly, when peers give longer-form identifiers, as permitted in both the LibTorrent Extension Protocol and the Azureus Messaging Protocol, unknown or mismatched identifiers are written to the log file. Below is an example real entry, demonstrating a large number of available tagging bits:

```
- [2009] Log File Opened for Vuze 4.2.0.2
- [0406 09:16:22] unknown_client [LTEP]:
        "Unknown KG/2.2.2.0" / "KGet/2.2.2"
        [4B4765742F322E322E32],
        Peer ID: 2D4B47323232302D494775533761494E45425245
- [0406 09:22:14] mismatch_id [LTEP]:
        "BitTorrent SDK 2.0.0.0" / "BitTorrent SDK 2.0"
        [426974546F7272656E742053444B20322E30],
        Peer ID: 2D4245323030302D2759514731415950276462 62
```

The second tagging opportunity arises from a more subtle side channel present in the log. In particular events for protocol errors can be triggered at timed intervals, such that the inter-event timing forms a side channel. As a simple example, one second between log entries could represent an encoded `0`, and two second delays a `1`. Past work has shown that channels of this kind are not difficult to implement and that data can be encoded and sent reliably even in the absence of feedback about the time on the receiving system [3, 20]. This is made easier by the fact that BitTorrent files are often large and clients tend to stay on the network for a long period of time [10].

In both cases, these events are logged, but no information is given to the user through the GUI. Both cases clearly allow for tagging, as the log includes information chosen by the remote peer. Further, these tags preserve sequencing information, due to their explicit timestamps.

4.4 Using Tagging to Improve Forensic Investigation

While tagging can be used by one investigator to later identify a particular system, it becomes a more powerful tool for law enforcement when used collaboratively by different investigators. By providing a central repository of tags placed on systems along with a history of when and why the system was tagged, tagging can support a broader and more effective investigation. This approach mirrors the very successful database currently provided by our RoundUp tool, which monitors when a particular user, identified by IP address or p2p client GUID, was seen sharing contraband.

Using the tag database, investigators who seize a sys-

tem can retrieve tags from the system and use them to identify other times the system was seen and tagged online. This can help identify the same system that was seen in with different IP addresses or that was participating in different p2p networks. The ability to recognize the same system in different contexts provides new enforcement opportunities currently unavailable to LE. In particular, it will allow LE to better understand the proclivities of the owner of the system and allow better understanding of the overall community of offenders. For example, if a system is found that has been tagged while sharing contraband frequently or over different networks it could be likely that the user is more likely to be a serious offender than the owner of a system that has fewer tags.

The use of tagging can also help law enforcement understand the scope of the overall problem of CP on p2p networks. While it is possible to count how many users are sharing contraband over time, it is not possible to tell which are users that are reappearing and which are new users coming online. The tags found on seized computer will allow better estimates of the problem by permitting more accurate population estimation with methods similar to the mark-and-release method of estimating animal population in the wild.

5. ISSUES IN FORENSIC MEASUREMENT

Our contributions are fully in the setting of forensic methods rather than standard network measurement techniques. Therefore, in this section, we describe the investigative process used by law enforcement in the United States and discuss the legal requirements that differentiate these investigations from simple observations.

5.1 Legal and Practical Issues

Criminal investigators are limited under the law, and techniques must be carefully designed to comply with applicable law, and to provide valid evidence.

The investigative process. In cases of Internet trafficking of child pornography, the goal of an investigator is to link observed criminal behavior on the network with evidence of that behavior on a specific machine. In the US, the steps an investigator takes are roughly:

1. Issue a search for known terms relating to child pornography (CP).
2. Directly connect to and observe the IP address of a peer possessing CP.
3. Subpoena the peer's ISP to determine the physical location of the peer's IP address.
4. Execute the search warrant, seizing relevant machines and associated storage media.
5. Verify that the storage media contain evidence of the crimes observed on the network.

Only Steps 1 and 2 correspond to typical Internet research measurements, with the remainder corresponding to legal requirements for criminal prosecutions.

All of these steps, however, are informed by various legal requirements and doctrine as follows. First, police can use evidence in *plain view* to start a criminal investigation. As most p2p systems freely advertise shared content among users, this requirement is met; i.e., LE is not wiretapping at a router, they are the end of a TCP connection accepted by the remote peer. Courts demand *probable cause* to allow law enforcement to search for and possibly seize private property (Step 4). The probable cause standard is satisfied by evidence that establishes "a fair probability that contraband or evidence of a crime will be found in a particular place" [1, 11]. Finally, a forensic examination of storage media is conducted by appropriate experts, and the existence of files, identifiers, logs, and other evidence may be used to support a prosecution that seeks to prove guilt beyond a reasonable doubt, a much higher standard than probable cause. Our goal here is to introduce forensic techniques that meet the higher standard.

There are several key considerations for law enforcement that relate to, but differ from, those of the typical disinterested researcher:

Evidentiary standards: Information that does not meet an evidentiary standard of either probable cause for warrants or beyond a reasonable doubt for convictions is merely a lead, and of lesser value. Information collected about a target isn't evidence if it came from a third party. Such information can serve to support search warrants, but not convictions. This distinction between leads and evidence is roughly analogous to the difference between observation studies used to generate hypotheses, and controlled studies used to test them. In the US, a court will not accept evidence that is not acquired through specific definitions of *plain view*, a magistrate-ordered warrant, or similar legal procedures.

Intent is part of the crime: Many crimes include intent as a requirement for conviction. Possession of CP is legal when unintentional, such as if it is unknowingly held in a spam folder. Among other indicia, multiple attempts to download CP can demonstrate intent [9], as could a growing collection, or the presence of organized backups of the CP.

Public-use technology only: *Kyllo v. US (No. 99-8508)* established that prior to obtaining a search warrant, investigators cannot collect information using technology that is not already in general public use. In practice, this means that law enforcement is limited to existing protocols, exploiting vulnerabilities without a warrant to gather evidence can violate *Kyllo*.

6. RELATED WORK

Existing methods of matching traffic to end systems rely heavily on mutable identifiers such as IP addresses. Alternatives available to investigators use statistical properties that also are both protean and easily falsified. For example, a remote peer's clock skew can be measured based on TCP headers [14, 17], but this value is both affected by temperature and easily falsified by modifying the TCP header. Even radiometric identifiers [2] can be altered by a target [6]. To our knowledge, no statistical measure of a remote peer is used by practitioners other than OS fingerprinting using tools like nmap. Our user-centric approach differs from past measurement and characterization of deployed p2p networks [4, 12, 16]. These past works have focused primarily on performance-related metrics, with an eye toward improving typical user experience in file sharing networks such as Gnutella. In contrast, our focus is on forensic investigation and criminal conduct. Finally, our work employs well-known steganographic, watermarking techniques, though we apply them to a novel scenario.

7. CONCLUSION

In this paper, we have made two major contributions. First, we have shown through analysis of large-scale, real-world data sets that both network- and application-level identifiers are not reliable evidence for more than probable cause. Second, we have presented tagging, a mechanisms for the active creation of reliable, verifiable identifiers that easily meet the standard of beyond a reasonable doubt. We have provided a strong theoretical model to demonstrate the applicability of tagging to this problem, and we have presented several distinct avenues for tagging across applications and the operating system.

Our work in forensics exists at the intersection of network measurement and security, and it is informed by our practical experience in this area with our law enforcement colleagues. In general, the best method of approaching real problems is often risk mitigation rather than perfect security, and criminal investigation is no exception. Even though evidence can be erased or lost, catastrophic failure is not present in forensics to the degree that it is in security (no one user can erase all evidence). Moreover, erasing specific data from a machine is much harder than erasing all data, and the latter is still an indication of something. This mode is true for other areas of practical security. For example, despite power monitoring attacks on cryptosystems [7, 13, 19] most people do not use tamper-proof hardware. Although their system is not proof against these attacks, there is still value in defeating most other attackers. Similarly, the Tor privacy network is architected to provide reasonable performance instead of perfect security against known attacks [5]. And the TSA admits it cannot defeat all terrorists, and instead simply mitigates risks [8].

Acknowledgements. We are grateful for the collaboration of Det. Capt. Thomas Kerle of the MA State Police and Corp. Robert Ederly of the PA State Police, as well as over 200 members of US state and federal law enforcement who helped us carry out our measurements. We thank George Bissias and Ramesh Sitaramen for insightful discussions of analyses of tagging techniques.

This work was supported in part by National Institute of Justice Award 2008-CE-CX-K005 and in part by the National Science Foundation awards CNS-0905349 and DUE-0830876. The opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect those of their employers, the U.S. Department of Justice, or the National Science Foundation.

8. REFERENCES

[1] Illinois v. Gates, 462 U.S. 213, 238 (1983).

[2] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless Device Identification with Radiometric Signatures. In *Proc. ACM MobiCom*, pages 116–127, October 2008.

[3] S. Cabuk, C. Brodley, and C. Shields. IP Covert Channel Detection. *ACM Trans. on Info. Sys. Sec.*, 12(4), 2009.

[4] J. Chu, K. Labonte, and B. N. Levine. Availability and Locality Measurements of Peer-to-Peer File Systems. In *Proc. ITCom*, volume SPIE 4868, pages 310–321, July 2002.

[5] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. USENIX Security Symposium*, Aug. 2004.

[6] M. Edman and B. Yener. Active Attacks Against Modulation-based Radiometric Identification. Technical Report 09-02, RPI, 2009.

[7] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani. On the Power of Power Analysis in the Real World. In *Proc. CRYPTO*, pages 203–220, 2008.

[8] K. Hawley. TSA's Take on the Atlantic Article. `http://www.tsa.gov/blog/2008/10/tsas-take-on-atlantic-article.html`, Oct 21 2008.

[9] T. Howard. Don't Cache Out Your Case. *Fall, 2004 Berkeley Technology Law Journal*, 19, 2004.

[10] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice. Dissecting bittorrent: Five months in a torrent's lifetime. In *Proc. ACM PAM*, pages 1–11, 2004.

[11] N. Judish et al. Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations. US Dept. of Justice, 2009.

[12] A. Klemm, C. Lindemann, M. K. Vernon, and O. P. Waldhorst. Characterizing the query behavior in peer-to-peer file sharing systems. In

Proc. ACM IMC, pages 55–67, 2004.

[13] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis: Leaking Secrets. In *Proc. of CRYPTO*, pages 388–397, 1999.

[14] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2), April-June 2005.

[15] B. Levine, M. Liberatore, and C. Shields. Challenges to Digital Forensics Posed by Mobile Systems and Ubiquitous Computing. In *Proc. of ARO Workshop on Digital Forensics*, Sept. 2009.

[16] D. S. Menasche, A. A. Rocha, B. Li, D. Towsley, and A. Venkataramani. Content availability and bundling in swarming systems. In *ACM CoNext*, pages 121–132, 2009.

[17] S. J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of CCS 2006*, October 2006.

[18] National Research Council, Committee on Identifying the Needs of the Forensic Sciences Community. *Strengthening Forensic Science in the United States: A Path Forward*. The National Academies Press, February 2009.

[19] R. Novak. Side-channel Attack on Substitution Blocks. In *Proc. Applied Cryptography and Network Security*, pages 307—318, 2003.

[20] C. B. S. Cabuk and C. Shields. IP Covert Timing Channels: Design and Detection. In *Proc. of the 11th ACM Conference on Computer and Communications Security*, October 2004.

[21] J. Wolak, D. Finkelhor, and K. J. Mitchell. Child-Pornography Possessors Arrested in Internet-Related Crimes: Findings From the National Juvenile Online Victimization Study. Technical report, National Center for Missing & Exploited Children, 2005.