Do You Know Where Your Headers Are? Comparing the Privacy of Network Architectures with Share Count Analysis

David Naylor
Carnegie Mellon University
dnaylor@cs.cmu.edu

Peter Steenkiste
Carnegie Mellon University
prs@cs.cmu.edu

ABSTRACT

Online privacy is more important now than ever. Using encryption goes a long way by hiding application data from third parties, but some amount of private information is still exposed in packet headers. Tools like Tor are designed to address this concern, and, more recently, research efforts to replace IP have begun to consider how a network architecture itself can improve privacy.

Unfortunately, we do not have a good way to quantitatively compare network architectures (in terms of privacy and in general). We take the first steps in this direction by presenting *share count analysis*, a methodology for measuring "how private" an architecture is. We describe our design and implementation and present initial results indicating that this approach is promising.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

privacy; anonymity; anonymity metrics

1. INTRODUCTION

There is an increasing desire for privacy in the Internet. It is a tough problem: each packet gives away a lot of personal information. Most important is the payload; fortunately, encryption is an effective way to hide this content, illustrated by the growing use of TLS [13]. Headers, however, also leak information, such as the packet's sender and receiver. Protecting this information is much harder because the network needs to know the destination to deliver the packet and the destination needs to know the sender to respond.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
HotNets '15 November 16–17 2015, Philadelphia, PA USA
Copyright 2015 ACM 978-1-4503-4047-2 ...$15.00.
http://dx.doi.org/10.1145/2834050.2834104.

Many techniques have been developed to prevent headers from leaking identity information (e.g., mixes, crowds, and onion routing). More recently, a number of network architectures have also focused on improving privacy [9, 14, 12]. However, none of these solutions can defend against an arbitrary adversary with unlimited capabilities. Instead, the tools typically focus on a particular aspect of privacy or a particular type of attacker. This raises the question: how can we quantify and compare the effectiveness of various solutions?

This turns out to be a difficult problem. First, different network architectures use different network headers, an adversary's primary information source, and different network deployments may use different types of devices (e.g., NATs or onion routers) which manipulate those headers in different ways. A comparison tool must understand each architecture's headers and what information they leak. Second, the space of adversary capabilities is large, ranging from local sniffers on public Wi-Fi to global surveillance. Finally, it is not clear what metrics to use since both solutions and attackers differ widely. While there has been work on tool-specific analyses (e.g., for mixes, crowds, and onion routing [15, 5, 7]), it is not clear how to generalize this research to evaluate any tool or architecture.

This paper takes a first step towards a tool that can compare how well different tools and architectures preserve user privacy. We introduce a technique called *share count analysis*, which is based on two ideas. First, instead of working with specific network headers (e.g., IPv4) and network devices (e.g., a NAT), we define generic models for network headers that contain only four canonical *meta-fields* and then define devices based on how they interact with those meta-fields. By mapping the specifics of different header formats and devices onto these models, we can more easily compare their privacy properties. Second, we develop a flexible adversary model that allows our tool to evaluate how well different solutions deal with several specific adversaries. We implement share count analysis, use it to compare a variety of architectures, and present results indicating that this technique is promising.

2. GOALS AND CHALLENGES

Our goal is to measure "how private" a network architecture or tool is. We would like to do this in a general way, that is, analyze how an arbitrary architecture or tool stands up against an arbitrary adversary over a variety of privacy metrics. The primary difficulty we face, then, is making our technique *general*. In particular, we see four concrete challenges:

(1) Supporting arbitrary packet formats. We want to support any architecture, and each architecture has a different header format. For example, the Accountable and Private Internet Protocol (APIP) [14] replaces IP's source addresses with two new addresses, a *return address* and an *accountability address*. Similarly, the Internet Indirection Infrastructure (i3) [16] uses a flow identifier in place of the ultimate destination. Other proposed architectures differ even more radically; for example, NDN [9] uses object names as destinations and no source addresses at all. Given this variety, building a tool that can manipulate header fields and understands what personal information they give away requires care.

(2) Supporting arbitrary network elements. A lot of network devices, like NATs, change headers as packets move through the network, and these changes affect how much personal information the packet leaks. For example, though not traditionally deployed for privacy, NATs replace a source address that identifies the sending machine with one that only identifies the source network, making it harder to link two flows from the same sender. Some architectures or tools introduce new types of devices (for example, Tor relays and i3 rendezvous nodes both change packet headers in a way that is critical to the privacy those solutions provide), so we also need to be able to handle new in-network behavior. And even well-known devices, like NATs, might operate on a field that different architectures interpret differently, so the impact such a change has on privacy is not consistent.

(3) Supporting arbitrary adversaries. A sufficiently powerful adversary can often learn anything. Therefore, it is important to be able to say that a particular tool can protect a particular piece of information *from a particular adversary*. Rather than design our model with a particular adversary in mind (e.g., an attacker who can compromise k of n Tor nodes), we would like to model arbitrary adversaries (e.g., one who can compromise some Tor nodes but perhaps also a link inside the sender's local network, before the NAT).

(4) Considering a broad definition of privacy. Many privacy analyses focus on a single metric, like the size of the anonymity set or the entropy in the distribution of probabilities that host H sent message M. Our ideal tool would answer multiple questions about what the adversary is able to learn as well as characterize the costs of this privacy (e.g., performance overhead).

3. SHARE COUNT ANALYSIS

Our technique, *share count analysis*, draws inspiration from header space analysis (HSA) [10], though the details are different since our goal is not to verify properties of *actual operational networks* but rather to *quantitatively compare network architectures*. Like HSA, we send test packets through models of network devices and track how the headers change. Unlike HSA, we are not concerned with the actual bits in the header, but rather how much information they give away. We represent this leaked information using *share counts*, which we explain in §3.2. First we describe our models for the adversary, packets, and the network, and then we describe how we use share counts to analyze an architecture.

3.1 Threat Model

We assume the adversary has the following goals and abilities. We think this threat model is sufficiently general to cover a wide range of realistic adversaries, but in the future we hope to expand it to include even more capable attackers.

Goals: The adversary has two specific goals:

(G1) **Given a packet, link the human sender to the destination *("WHO").*** We assume that the adversary's best shot at doing this is to learn the packet's source network, which, combined with some amount of external information (see Table 1), could yield a small set of human individuals.

(G2) **Given a user, construct a history of their online activity *("WHAT").*** For this, the adversary must link flows to a common "sender ID" (e.g., an IP address). The sender ID may be opaque in the sense that it does not identify a person; here it is only needed to link flows from the same (potentially unknown) person.

Abilities: The adversary can have vantage points at any number of links or network boxes (e.g., routers or NATs). At each vantage point, it can inspect a packet's network and transport headers; we assume application payloads are encrypted. Equality checks can be used on these encrypted payloads to determine if packets observed at two different vantage points are actually the same packet. If a box re-encrypts a packet's body, then as far as the adversary can tell, the "before" and "after" versions are two different packets (unless the adversary controls the box that does the re-encryption).

Information leaked to the destination in application payloads is beyond the scope of this paper. We also do not consider timing attacks or active adversaries that can drop, modify, or inject packets in an attempt to learn something. Finally, we do not currently consider information the adversary could infer based on the topological location of a vantage point.

This network info...	...narrows the anonymity set to...	...if the adversary knows
Source domain	customers of source ISP	list of ISP's customers
Source domain	affiliates of business/university	employee/student roster
Source domain	residents of neighborhood	network address to location mapping
Destination domain	customers of company/service X	list of X's customers
Destination domain	people interested in topic Y	advertising profiles
Destination domain	particular user of public site Z	activity log (posts & timings) from Z

Table 1: Examples of external information an adversary could use to connect packets to human users.

3.2 Network Model

Packets We address the first challenge, diversity in header format, by making no assumptions about the contents or formats of headers. Instead, we represent each packet as four *meta-fields*: all of the header bits that contribute to identifying (1) the sender, (2) the source network, (3) the destination, and (4) the flow. These meta-fields may overlap—for example, in IP, the source address both identifies the sender and contributes to identifying the flow. Before using our tool, a human expert must decide how to map the bits in each architecture's header to these four meta-fields. (Figure 1 shows this mapping for TCP/IP.)

In addition to the meta-fields, each packet has a *body*, which, as noted above, we assume is encrypted and so we treat as an opaque value used only for equality testing to link multiple snapshots of the same packet together even if the headers have changed.

Finally, each packet carries a *share count* for each meta-field. A share count indicates how many entities in the network could share the current value for that meta-field. For example, how many senders share the same value for the Sender-ID? In IP, when a packet leaves the sender, its Sender-ID share count is 1; when it leaves a NAT, the share count is the number of hosts in the source network. We also include a fifth share count, which tracks how many of a sender's *flows* share the same Sender-ID (as opposed to how many *senders*). For instance, if a host is multihomed and sends half of its traffic on each link, the sender flow share count is $\frac{1}{2}$.

Network Boxes We model a path through the network as a series of *network boxes* connected by links. Boxes may change the values of some header fields (and therefore change the values of some meta-fields and share counts) or of the body (e.g., a Tor relay re-encrypts it). To meet the second challenge, diversity of network elements, each box is defined by two properties (which must be manually specified for each architecture):

(1) Does the box change any *meta-fields* and/or the *body*? We do not model actual values; each field is represented as an integer, which is incremented if a box changes it.

(2) What are the *share counts* for each meta-field after a packet leaves the box?

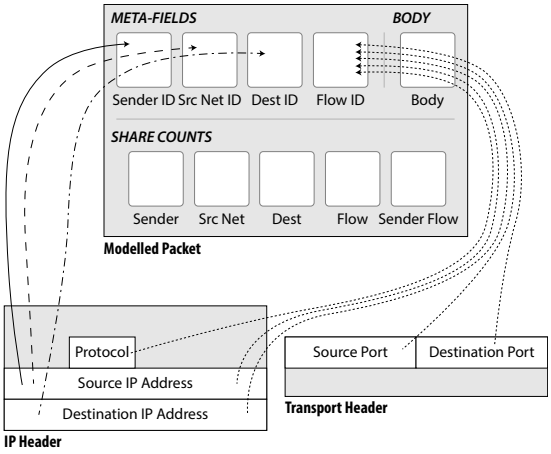


Figure 1: Example showing how TCP/IP fields map to meta-fields.

3.3 Analysis Procedure

Share count analysis uses the model presented above by creating a path of network boxes, labelling some links and boxes as vantage points, sending a symbolic packet along the path, and recording the share counts seen by the adversary. The adversary learns something if a share count ever reaches one. (Being able to model an adversary simply by placing vantage points on any set of links and boxes gives us the flexibility to address the third challenge, adversary diversity.) Here we describe the details, presented in five steps.

(1) Generate paths. Since we do not currently take the physical location of a vantage point into account, we test using a single path. The base path is:

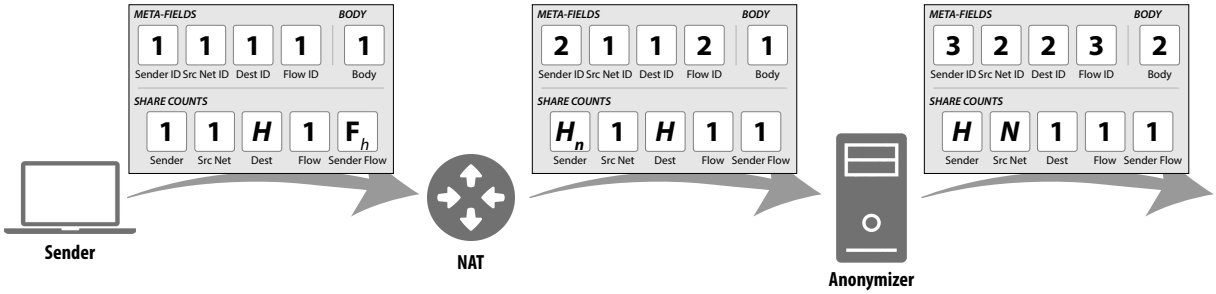
```
[sender]-[router]-[destination]
```

When we test an architecture or tool that requires specialized boxes, we add those boxes as needed. For example, the test path for IP w/ NAT is:

```
[sender]-[nat]-[router]-[destination]
```

Depending on the adversary we want to model, we set certain links and boxes to be vantage points (see §3.4).

(2) Forward test packet. Next we forward a test packet through the path, updating the meta-fields, body, and share counts at each box (Figure 2). At each vantage point, we save a snapshot of the packet state.



(N = num networks; H = num hosts; H_n = hosts per network; F_h = flows per host)

Figure 2: Test packet travelling through a NAT and an anonymizer.

(3) Group "linkable snapshots." The adversary uses the packet body to link packet snapshots from different vantage points. If a box changes the body, then, as far as the adversary knows, snapshots from vantage points on either side of that box were generated by different packets, meaning the adversary cannot combine the information it learns from each set of snapshots individually. Of course, if the adversary has a vantage point *on* a box that changes the body, then it *can* link the snapshots. The result of this step is one or more sets of *linkable snapshots*.

(4) Consolidate information from snapshots. For each set of linkable snapshots and for each meta-field, select the snapshot with the minimum share count. Record this share count and the corresponding meta-field value. For the Flow-ID share count and the sender flow share count, we instead record the maximum.

(5) Test what adversary learned. We can use the minimum share counts to test whether the adversary achieved its goals. It was able to link the source network to the destination (G1) if:

(Source-Network-ID share count == 1) &&
(Destination-ID share count == 1)

Similarly, the adversary linked the sender to the destination (G2) if:

(Sender-ID share count == 1) &&
(Destination-ID share count == 1)

3.4 Metrics

Share count analysis allows us to evaluate architectures using multiple metrics, each involving a run of the procedure described in §3.3. We start by asking whether four real-world adversaries can achieve G1 and G2:

(1) **Local Eavesdropper** *(e.g., someone sniffing open Wi-Fi)*: We place a vantage point on the link leaving the sender.

(2) **Global Surveillance** *(e.g., monitoring by a nation-state)*: We place a vantage point on every router.

(3) **Source ISP** *(e.g., employer or school)*: We place a vantage point on the link leaving the sender. We relax G1 and G2 to only check the Destination-ID share count, since the source ISP already knows the sender and source network.

(4) **Destination** *(e.g., web server)*: We place a vantage point on the destination. We relax G1 and G2 to only check the Source-Network-ID and Sender-ID share counts, since the destination already knows its own identity.

Next, we consider arbitrary network adversaries by testing all combinations of vantage points and asking:

(5) What is the minimum number of vantage points needed to achieve G1? And G2?

(6) How many different combinations can achieve these minimums? (More possibilities means more opportunities for the adversary to succeed.)

Finally, we check two adversary-independent metrics:

(7) What is the maximum Flow-ID share count at seen at any box?

(8) What is the maximum sender flow share count seen at any box?

For privacy reasons, some architectures may give multiple TCP flows the same Flow-ID. However, this means that network boxes have coarser-grained handles for traffic engineering and, worse, that if an administrator wants to block a misbehaving flow, other benign flows will be blocked with it. Therefore, the maximum Flow-ID share count (7) is a measure of collateral damage. If a single host has multiple Flow-IDs, the maximum sender flow share count (8) indicates how successfully the adversary can reconstruct a user's activity history.

Finally, note that the eight questions we list here are merely examples of the kinds of metrics share count analysis supports; as we extend the model, we can ask more (and more sophisticated) questions. As it stands, though, this list demonstrates the variety of information we can learn from share counts. This is how we address our fourth challenge, metric diversity.

	1-G1	1-G2	2-G1	2-G2	3	4-G1	4-G2	5-G1 (6-G1)	5-G2 (6-G2)	7	8
IP	•	•	•	•	•	•	•	1 (3)	1 (3)	1	F_h
IP-NAT	•	•	•	○	•	•	○	1 (5)	1 (2)	1	F_h
IP-Tor	○	○	○	○	○	○	○	3 (1)	3 (1)	1	F_h
APIP-ISP-NAT-Unique	•	•	•	•	•	•	•	1 (5)	1 (5)	1	$\frac{F_h \cdot H_d}{K}$
APIP-External-Encrypted-Shared	○	○	○	○	•	•	•	∞ (0)	∞ (0)	H_f	1
i3	○	○	○	○	○	○	○	1 (1)	1 (1)	1	F_h

(F_h = flows per host; H_d = hosts per delegate; K = num flow IDs; H_f = hosts per flow ID)

Table 2: Overall comparison of representative combinations of architectures and tools.

4. RESULTS

We implemented share count analysis and ran it on IP, APIP, and i3. We tested five variants of IP (IP, IP w/ NAT, IP w/ Anonymizer, IP w/ Tor, and IP w/ NAT and Tor). We assume i3 is deployed as an overlay where packets from endpoints to the i3 infrastructure expose the endpoint's IP address and that TLS is used between endpoints and the i3 rendezvous node, meaning the payload is re-encrypted at the rendezvous server.

APIP packets carry two addresses: a *return address* (for replying to the sender) and an *accountability address* (which points to a third party delegate who fields complaints about the packet). We tested all eight combinations of the following three options: First, senders can "hide" their return address using (1) **NAT** or (2) **encryption** (the return address is encrypted with the payload, so it is only accessible to the final destination). Second, the delegate could be run by (1) the **source ISP**, meaning the accountability address gives away the sender's source network, or (2) an **external third party**. Third, each APIP packet contains a flow ID, which routers use to block malicious flows. Delegates assign each client either (1) a set of flow IDs that are **unique** to it or (2) a set of flow IDs that are **shared** among multiple of the delegate's clients.

In this section we summarize the results, which are labelled by question number (1–8, §3.4) and goal (G1 or G2, §3.1). For yes/no questions, • = yes and ○ = no.

Comparing Architectures To start, Table 2 presents the complete results for IP, IP w/ NAT, IP w/ Tor, a weak and a strong variant of APIP, and i3.

First, we see that IP was not designed with privacy in mind; you need Tor to get any kind of privacy guarantees. Using a NAT helps a little by preventing adversaries without vantage points inside the NAT from linking two separate flows initiated by the same sender. Tor does quite well, although this analysis does not reflect its performance overhead.

Second, the weakest version of APIP (ISP delegate with NATed return addresses and unique flow IDs) appears to be no better than plain IP. There is one subtle difference in column 8, however: since each sender has a set of flow IDs to choose from for each flow, the adversary can only link a fraction of a sender's activity. (Furthermore, though not represented in the results, any version of APIP has stronger accountability than IP.)

Next, the strongest version of APIP we tested falls short of Tor in terms of privacy, but it comes with none of Tor's performance costs. Also, note that the infinities for 5-G1 and 5-G2 are misleading; currently our model only considers on-path boxes, so infinity is correct in the sense that there is no combination of on-path boxes that could be compromised to achieve G1 or G2. However, if the accountability delegate were compromised, these connections could be made.

Finally, i3's stats match Tor's, with the exception that fewer vantage points are needed to achieve G1 and G2 (the adversary only needs to compromise the i3 rendezvous node instead of three Tor relays).

When Can the Adversary Learn the Source Network? The columns in the table below indicate whether a local sniffer, global surveillance, and the destination can learn the source network, respectively.

	1-G1	2-G1	4-G1
IP	•	•	•
IP-NAT	•	•	•
IP-Tor	○	○	○
APIP-ISP--**	•	•	•
APIP-External--**	○	○	•

Since IP was not designed to protect this information, only Tor is able to hide it; even a NAT does not help. For APIP, if source domains act as accountability delegates, then the accountability address gives away the source network. With an external delegate, the source network is hidden from local and global adversaries. However, the destination server still learns the source network. If the return address is included (encrypted) inside the packet body, then the destination learns the sender's unmodified address and therefore the source network. If the return address is hidden using NAT, then the destination cannot connect the packet to a particular sender, but still learns the source domain. This could be mitigated if multiple ISPs were willing to perform address translation as packets leave their networks.

Finding Effective Vantage Points Share count analysis can give us a sense of *how hard* it is for an adversary to achieve a particular goal by determining the minimum number of vantage points needed (fewer vantage

points means easier attack) and how many ways those vantage points could be placed (more options means easier attack). For example, in IP a single vantage point is enough to link both the sender and the source network to the destination. With Tor, on the other hand, three vantage points are needed (the three Tor relays). And, with Tor and a NAT, linking the sender to the destination requires a fourth vantage point: one inside the NAT to link the sender's internal and external addresses.

	5-G1	*6-G1*	*5-G2*	*6-G2*
IP	1	3	1	3
IP-Tor	3	1	3	1
IP-NAT-Tor	3	1	4	2

Of course, share count analysis can also produce the successful paths themselves. As a sanity check, we see that for IP with NAT and Tor, the adversary must compromise the Tor relays and also either the link inside the NAT or the NAT itself (* indicates a vantage point):

```
[sender]*[nat]-[router]-[tor-entry*]-[router]-[tor-
relay*]-[router]-[tor-exit*]-[router]-[destination]

[sender]-[nat*]-[router]-[tor-entry*]-[router]-[tor-
relay*]-[router]-[tor-exit*]-[router]-[destination]
```

The Cost of Privacy Next we consider the adversary's ability to link multiple flows to the same user for the different variants of APIP. The first column in the table below shows whether a global adversary can link the packet to both the destination and the sender (2-G2). The second column shows how many total TCP flows are shut off when a misbehaving flow is reported (in APIP, routers block bad flows that have been reported to accountability delegates). Finally, the third column shows how many of the sender's flows can be linked (in APIP, each sender is given a group of flow IDs to assign to its flows as it chooses).

	2-G2	*7*	*8*
APIP--*-Unique*	•	1	$\frac{F_h \cdot H_d}{K}$
APIP--*-Shared*	∘	H_f	1

(F_h = flows per host; H_d = hosts per delegate; K = num flow IDs; H_f = hosts per flow ID)

First, notice the obvious tradeoff between sender-flow linkability and collateral damage. With shared flow IDs, the adversary cannot link flows to users, so it cannot build a history of any user's online behavior. On the other hand, since multiple hosts share the same set of flow IDs, when a router blocks a malicious flow, it could also block up to H_f benign hosts' flows as well.

Second, even when the adversary can link flows to senders (i.e., when each sender is assigned a set of unique flow IDs), since it has multiple flow IDs to choose among, traffic sent with different flow IDs appears to come from different senders. In the third column, we see that the maximum fraction of the sender's flows that can be linked is $\frac{F_h \cdot H_d}{K}$. The denominator, K, is the number of possible flow IDs, which is determined by the number of bits in the header given to the flow ID. This is useful feedback to protocol designers, who can now see a direct numerical link between privacy and header format.

5. EXTENSIONS AND FUTURE WORK

We see a number of possible extensions for share count analysis. First, an adversary can learn something about a packet based on the physical location of the vantage point. Accounting for topology could give us a richer adversary model. Second, though our analysis in this paper touched on the costs associated with some privacy tools, we would like to push this further. For example, for Tor we might report the latency increase or the computational cost of the extra crypto. Next, increased privacy sometimes raises concerns about decreased accountability; it would be nice to measure "how accountable" an architecture or tool is. This might be possible by modelling administrators as adversaries with special capabilities and verifying that they *can* connect a packet to a user. Finally, though this may be substantially more difficult, it would be nice to support a more sophisticated threat model, for instance, one in which the adversary could perform timing analysis attacks.

6. RELATED WORK

We are not the first to try to quantify privacy. Many privacy metrics have been proposed, based on anonymity set size [3], information theory [5, 15, 2, 4], and combinatorics [6, 8]. These efforts focus primarily on defining the anonymity *metric* itself and only describe how to use it to *measure* a protocol using toy examples. Our work complements them by describing a way to measure any architecture or tool. Furthermore, these analyses focus only on sender anonymity, while we consider a range of metrics, like the ability to identify the source network. Finally, our tool pursues a broader picture by also measuring certain costs of privacy.

Many other efforts measure information leakage at the application layer, e.g., in social networks [1] or web browsing [11]. These tools are complimentary to ours, which focuses on the network and transport layers.

7. CONCLUSION

In this paper we present share count analysis, a methodology for measuring "how private" a network architecture or tool is according to various privacy metrics. We also present initial results indicating its potential and suggest several directions for future work.

Acknowledgements This work was funded in part by NSF under award number CNS-1345305 and by the DoD through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program.

8. REFERENCES

[1] J. Becker and H. Chen. Measuring privacy risk in online social networks. In *Web 2.0 Security and Privacy (W2SP)*, Oakland, CA, May 21, 2009.

[2] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. In *Trustworthy Global Computing*, pages 281–300. Springer, 2007.

[3] D. Chaum. Untraceable electronic mail, return address, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[4] Y. Deng, J. Pang, and P. Wu. Measuring anonymity with relative entropy. In *Formal Aspects in Security and Trust*, pages 65–79. Springer, 2007.

[5] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68. Springer Berlin Heidelberg, 2003.

[6] M. Edman, F. Sivrikaya, and B. Yener. A combinatorial approach to measuring anonymity. In *Intelligence and Security Informatics, 2007 IEEE*, pages 356–363. IEEE, 2007.

[7] J. Feigenbaum, A. Johnson, and P. Syverson. A model of onion routing with provable anonymity. In *Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security*, FC'07/USEC'07, pages 57–71, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] B. Gierlichs, C. Troncoso, C. Diaz, B. Preneel, and I. Verbauwhede. Revisiting a combinatorial approach toward measuring anonymity. In *Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society*, WPES '08, pages 111–116, New York, NY, USA, 2008. ACM.

[9] V. Jacobson, D. K. Smetters, J. D. Thornton, et al. Networking named content. CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[10] P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: Static checking for networks. NSDI '12, pages 113–126, San Jose, CA, 2012. USENIX.

[11] B. Krishnamurthy, D. Malandrino, and C. E. Wills. Measuring privacy loss and the impact of privacy protection in web browsing. SOUPS '07, pages 52–63, New York, NY, USA, 2007. ACM.

[12] V. Liu, S. Han, A. Krishnamurthy, and T. Anderson. Tor instead of ip. In *HotNets '11*, page 14. ACM, 2011.

[13] D. Naylor, A. Finamore, I. Leontiadis, et al. The Cost of the "S" in HTTPS. CoNEXT '14, pages 133–140, New York, NY, USA, 2014. ACM.

[14] D. Naylor, M. K. Mukerjee, and P. Steenkiste. Balancing accountability and privacy in the network. SIGCOMM '14, pages 75–86, New York, NY, USA, 2014. ACM.

[15] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In R. Dingledine and P. Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53. Springer Berlin Heidelberg, 2003.

[16] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. SIGCOMM '02, pages 73–86, New York, NY, USA, 2002. ACM.