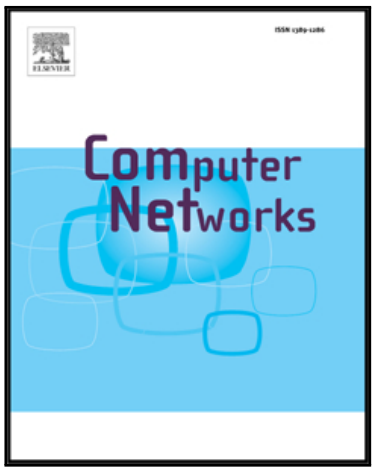

Collaborative and Privacy-Preserving Estimation of IP Address Space Utilisation

Sebastian Zander, Lachlan L.H. Andrew, Grenville Armitage

PII: S1389-1286(17)30079-8
DOI: 10.1016/j.comnet.2017.03.010
Reference: COMPNW 6125

To appear in: *Computer Networks*

Received date: 24 March 2016
Revised date: 16 February 2017
Accepted date: 10 March 2017

Please cite this article as: Sebastian Zander, Lachlan L.H. Andrew, Grenville Armitage, Collaborative and Privacy-Preserving Estimation of IP Address Space Utilisation, *Computer Networks* (2017), doi: 10.1016/j.comnet.2017.03.010

Collaborative and Privacy-Preserving Estimation of IP Address Space Utilisation

Sebastian Zander[a,*], Lachlan L. H. Andrew[b], Grenville Armitage[c]

[a] *School of Engineering and IT, Murdoch University, Perth, Australia*
[b] *Faculty of IT, Monash University, Melbourne, Australia*
[c] *School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia*

Abstract

Exhaustion of the IPv4 address space is driving mitigation technologies, such as carrier-grade NAT or IPv6. Understanding this driver requires knowing how much allocated IPv4 space is *actively used* over time – a non-trivial goal due to privacy concerns and practical measurement challenges. To address this gap we present a collaborative and privacy-preserving capture-recapture (CR) technique for estimating IP address space utilisation. Public and private datasets of IP addresses observed by multiple independent collaborators can be combined for CR analysis, without any individual collaborator's privately observed addresses leaking to the others. We show that CR estimation is much more accurate than assuming all used addresses are observed, and that our scheme scales well to datasets of over a billion addresses across several collaborators. We estimate that 1.2 billion IPv4 addresses and 6.5 million /24 subnets were actively used at the end of 2014, and also analyse address usage depending on RIR and country.

Keywords: Actively used IPv4 space, Privacy-preserving capture-recapture

1. Introduction

By mid 2015 all Regional Internet Registrars (RIRs), except AfricNIC, had less than one /8 (2^{24}) of unallocated IPv4 addresses remaining [1]. While the IPv4 space is still not allocated completely, is is practically 'exhausted' since it is hard for organisations to get new prefixes from the RIRs. Yet, at the same time, the number of devices connected to the Internet is rapidly increasing – driven by "the Internet of Things (IoT)" [2]. This has created a sense of urgency around the deployment of short- and long-term mitigation technologies, such as carrier-grade Network Address Translation (NAT) or IPv6, and markets for trading allocated IPv4 prefixes. Understanding the degree of IPv4 address prefix exhaustion requires plausible estimates of actual IPv4 address use – particularly the fraction of addresses in allocated prefixes that are actively used. We

*Corresponding author
Email addresses: s.zander@murdoch.edu.au (Sebastian Zander), lachlan.andrew@monash.edu (Lachlan L. H. Andrew), garmitage@swin.edu.au (Grenville Armitage)

present estimation techniques that can track progressive exhaustion even once all IPv4 prefixes are allocated.

Early work [3, 4, 5, 6] that studied, among other things, IPv4 space growth focused on IP addresses observed mainly by "pinging". As part of their work, Dainotti *et al.* [7, 8] estimated the used IPv4 /24 (2^8 address) subnets based on IP addresses observed in several data sources. Apart from a simple multiplier in [4], prior work did not attempt to correct their estimates for under-sampling.

To address the issue of under-sampling, in [9, 10] we proposed estimating the population of both observed and unobserved (yet still active) IPv4 addresses from multiple diverse data sources using a statistical *capture-recapture* (CR) model. One challenge of CR is heterogeneity (where the probability of observing an IP address depends on whether it is used by, for example, a server or client), which we addressed using log-linear models. We now introduce another CR technique that models heterogeneity more directly, and we compare both approaches and show that the estimates of both approaches are similar.

CR techniques estimate population sizes based on (1) the number of IP addresses in each source and (2) the *sizes of intersections* between all combinations of sources. A key challenge is to efficiently compute the input needed for CR from data sources of multiple collaborators in a privacy-preserving ("private") manner, i.e. *without* revealing the observed IP addresses. Many potential sources of 'used' IP addresses are privately held, and cannot be shared directly for privacy reasons. To solve this issue, we developed an efficient solution called Secure Fast Set Intersection (SeFaSI), by combining private (or secure) set intersection cardinality (PSIC) techniques with CR. With this approach the input data for CR can be computed while *ensuring the anonymity* of the IP addresses observed.

SeFaSI extends computationally-secure commutative encryption by adding hash-based sampling for scalability, prevents probing attacks for IPv4 addresses, and works for two or more semi-honest parties without the need for a trusted third party. We demonstrate that our protocol scales well with 5–10 collaborators and datasets of over one billion IPs. Evaluation on small networks for which ground truth is known shows that estimates from SeFaSI have a much smaller error than the estimate from simply aggregating all data sources.

Our new key contributions in this article are:

1. A novel application of CR to estimate the used IP addresses, which deals with heterogeneity of IP addresses directly;
2. A privacy-preserving protocol to feed our CR estimation techniques while keeping the observed IPs private;
3. A publicly available open source implementation of SeFaSI and our log-linear model CR technique [11], which we use to validate our approach;
4. Illustrative results for the estimated number of actively used IPv4 addresses and /24 subnets at the end of 2014.

Our new CR approach is also beneficial for other privacy-sensitive applications of CR. For example, it could be used in epidemiology to estimate populations of people with certain illnesses while ensuring the privacy of personal information used for matching the data sources, such as names or birth dates.

After a discussion of related work in Section 2, Section 3 explains the basics of CR, the input data needed for CR, log-linear models used in [10] and our new CR model. Sections 4 and 5 describe SeFaSI's design, implementation and evaluation. Section 6 presents our IPv4-space estimation results, and we conclude in Section 7.

2. Related Work

We first briefly review previous work in the area of CR before discussing related work on estimating the used IPv4 space.

2.1. Capture-Recapture

Capture-Recapture (CR), also referred to as Mark and Recapture, is an old technique that was initially developed in ecology to estimate the population of certain types of animals. The basic idea is to take multiple samples of the population (such as at different times) and then estimate the number of unsampled animals based on the overlap between samples. The first well-document CR method is the two-sample Lincoln-Petersen (L-P) method [12, 13], which we describe in Section 3.5. The L-P method was later extended to more than two samples by Schnabel [14].

The restrictions of the early CR methods, discussed in Section 3.3, led to the development of more powerful models in recent decades [15]. In ecology, CR models are classified as "closed" or "open" depending on whether they assume the population being sampled is the same for each sample or differs. The Jolly-Seber model is a commonly used model for open populations [16].

Another major area where CR methods have been applied for many years is epidemiology [17, 18]. In epidemiology, instead of sampling at different points in time, the samples are data sources concurrently collected over time, such as doctor records or hospital records. Often this data was originally collected for purposes other than CR. With concurrent data sources, the population size does not change between samples as in ecology and so in epidemiology closed models are usually used [18]. CR models commonly used in epidemiology include various types of log-linear models [17, 18] and the sample coverage approach [19].

Our application of CR is similar to the use of CR in epidemiology. Instead of medical records we use concurrently captured data sources that contain used IP addresses, such as server logs or logs from active probing.

2.2. IP Space Estimation

Pryadkin *et al.* [3] probed the whole allocated Internet with ICMP echo and TCP SYN probes. They discovered 62 million used IPv4 addresses in 2003–2004. They also showed that only a small number of allocated prefixes appeared to be heavily used, while a large part of the IPv4 space appeared unused.

Heidemann *et al.* [4] infrequently probed all allocated IPv4 addresses (*census*) and frequently probed address samples (*survey*) with ICMP echo pinging to study usage, availability and up-time of addresses. The last census from [4] accounted for 112

million used addresses in 2007[1]. Heidemann *et al.* compared ICMP probing with TCP port 80 probing and passive measurements based on small samples. They proposed a "correction factor" of 1.86, thus estimating that the total number of used IPv4 addresses in mid 2007 was 200–210 million.

Cai *et al.* [5] used ping survey data to analyse typical address block sizes and their characteristics. They did not estimate the used IPv4 address space, but observed that "most addresses in about one-fifth of /24 blocks are in use less than 10% of the time". In 2012, anonymous researchers hacked commodity routers to perform a port scan of the IPv4 Internet [6]. They detected 420 million addresses that responded to ICMP echo, which is consistent with our ping censuses [10].

In 2013–2014 we proposed using CR to estimate the population of used IPv4 addresses from multiple sources of IPv4 addresses [9, 10]. We found that our CR estimate is significantly higher than the aggregate number of observed IPv4 addresses from multiple measurement sources (1.1–1.2 billion estimated used IPv4 addresses vs. 750 million observed IPv4 addresses in mid 2014), but for /24 subnets the difference is smaller (6.2 million estimated used /24s vs. 5.9 million observed /24s in mid 2014). We also estimated the numbers of addresses and /24 subnets still available.

Dainotti *et al.* [7] developed techniques to filter out spoofed IPv4 addresses from darknet or NetFlow data and showed that the filtered datasets can be used to estimate Internet address space usage. With multiple datasets combined, they observed 4.8 million used /24 subnets in September 2012. In extended work Dainotti *et al.* [8] used further data sources, and analysed several aspects of /24 subnet usage. They identified 5.3 million used /24 subnets with data collected until October 2013. This is broadly consistent with the 5.7 million /24 subnets we observed in the year to September 2013 [10]. The difference is likely due to the longer time windows we use.

Moura *et al.* [20] repeatedly pinged 1 million IP addresses of one ISP to estimate the churn caused by dynamic IP addressing. They compared the ping data with ground truth provided by the ISP and report the accuracy of their churn estimation to be 70%. They also found that over the whole measurement duration most used IP addresses responded to ping, and while only 15% of the IP space was in use at any given time, 75% of the IP space appeared to be used over the whole measurement. Both findings are consistent with our observations that (1) for one IP address the churn leads to increased likelihood of responding to active probing and (2) that the number of *simultaneously* active addresses is significantly smaller then the number of active addresses.

Our work enhances previous work in that (1) we *estimate the total number of observable addresses* instead of just reporting the number of observed addresses and (2) our new technique allows estimation of the used addresses from *anonymised datasets*. The latter point is significant for researchers wishing to access IP address data from the wider networking community.

[1] More census data has been collected since 2007 but there is no analysis in the literature and it is not easy for organisations outside the USA to access this data. Since 2012 we have done our own ICMP *and* TCP SYN census scans (see Section 5.2).

3. Capture-Recapture Models

We now define more specifically what we are estimating, and describe the measurement setting. Then, we discuss the basic assumptions for CR in our scenario and how to extract the data required for CR. Next, we illustrate CR by describing the simplest case: the two-sample Lincoln-Petersen (L-P) estimator. This section concludes with a description of the CR models that we actually use.

3.1. Measurement Metric

Our goal is to estimate the number of IP addresses that were *actively used* during a measurement period. Since many IP addresses are (re)assigned dynamically (for example, with DHCP) and hosts may move between multiple static/dynamic addresses, the number of actively used IP addresses is likely to be higher than the number of simultaneously used addresses. For example, if dynamic addresses are drawn uniformly from a pool, all pool addresses could eventually be observed even if at most one address is in use at a time.

We argue that any addresses that could be observed during our measurement period were on "stand-by" and de facto in use. For example, addresses assigned to dynamic pools cannot be used elsewhere. In the future freed addresses from under-utilised pools may be used for other purposes, but we cannot measure such future optimisations.

3.2. Approach

We assume several collaborators have data sources of IP addresses known to be actively used during a measurement period (such as from server logs, traffic traces, or active probing). Each data source is a sample of the whole used IP address space (the total population), but may be biased, e.g., towards certain geographical areas or certain types of hosts. We can group addresses into three categories:

- Observable addresses that were sampled (type 1)
- Observable addresses that were not sampled (type 2)
- Unobservable addresses (type 3)

Type 3 addresses are those that cannot be observed for "structural" reasons, such as being in space that is not publicly routed, or being assigned to a device, such as a firewalled printer, that neither initiates external traffic nor responds to pings. In contrast, type 2 addresses are those that simply happen not to appear in our data sources, but may appear in other server logs, for example. Note that types 1 and 2 include firewalled hosts that initiate Internet traffic.

Since sampling type 3 addresses is impossible, we focus on estimating the observable portion of the IP space (the typical approach with CR). Our goal is to estimate type 2 addresses with CR and get an estimate of the actively used addresses that is more accurate than merely counting observed addresses.

3.3. Assumptions for Capture-Recapture

Four assumptions are commonly made by CR techniques.

1. Each individual has a unique and consistent identifier across different data sources;
2. The population is the same for each data source (closed population);
3. All individuals have the same probability of being observed (homogeneous population);
4. Appearance of an individual in one source does not affect its inclusion in other sources (source independence).

Assumption 1 clearly holds for IP addresses, as we only care whether an address was used and not who used it.

We make assumption 2, with all type 1 and type 2 addresses being potentially observable by each source. While address usage may be intermittent [5], addresses still remain in the population and can be observed. Likewise, previously unallocated addresses that become used can be observed. Assumption 2 is usually stated in terms of temporal variation (as "closed population"), but that only applies when data sources correspond to measurements at different times. In our study, the data sources measure concurrently during the same time periods and so temporal changes in the number of addresses do not result in each data source sampling a different population size (an "open population"). An "open" population would arise if a new dataset arrived whose first measurement occurred later than the last measurement of the existing datasets. Since most IP datasets are accumulated continually, the natural response to this new dataset would be to update all datasets before re-running CR, which again gives a "closed" population.

We cannot make assumption 3. The population of IP addresses is heterogeneous; for example, servers are more likely to respond to pinging, while client machines may be more likely to appear in certain traffic logs.

For our data sources, we believe there is no significant causal relationship to introduce source dependence. While some samples are dependent, i.e., IPs observed by our NetFlow source and one of the log-file sources, their number is insignificant ($< 1\%$ of the total). Nevertheless, heterogeneity gives rise to what is called *apparent source dependence.* For example, two sources that are biased towards client machines will *appear* to be positively correlated given another source that is less biased towards clients. Heterogeneity and source dependence are confounded and cannot be clearly separated [18]. So we cannot make assumption 4.

However, assumption 3 and 4 are usually violated *in almost any real scenario*. Because of this, a number of CR techniques have been developed that do not require assumptions 3 and 4 and can deal with heterogeneous populations and source dependence, for example, techniques based on log-linear models [17, 19] or latent class models [21]. We use these more advanced techniques.

Another implicit assumption is that a data source only samples IP addresses that were actually used. Addresses can be considered actively used when collected via active probing or from server logs of TCP-based applications (e.g., web server logs) for which a three-way handshake is completed. In other cases (e.g., passively captured

traffic traces) it is necessary to filter out observed IP addresses that were not actually used, such as spoofed IPs [7, 10].

Finally, CR assumes all individuals have a non-zero (but possibly very small) probability of appearing in any of the data sources. Individuals with zero sample probability (i.e., type 3 addresses) are not part of the CR estimate. We argue that all of our datasets sample IPs used by routers, servers/proxies, clients, but miss IPs used by specialised devices, such as printers. Sample probabilities for client IPs are non-zero even for 'narrow' data sources, because we measure over long time windows in which a single IP address is likely used by multiple devices/persons due to dynamic address assignment, NAT, or administrative changes.

3.4. Capture Histories

In ecology CR data is collected by repeatedly sampling populations. In other fields, such as epidemiology, CR is used with lists of individuals (data sources) that often were collected for purposes other than CR [18]. In our case we use data sources such as web server logs, a black list of senders of spam and active probing. The "capture history" of an individual, here an IP address, is the set of data sources in which it appears [17, 19]. This term, like "open" and "closed" populations, originates from ecology and suggests that sources are ordered chronologically, which is not the case in our application. The table of capture histories contains all possible capture histories and the total number of IP addresses for each capture history. To compute this table we need to know the sizes of all data sources and the sizes of all combinations of intersections of datasets.

Let N be the unknown number of distinct used IP addresses. Let t denote the number of data sources, and index sources by $1, 2, \ldots, t$. For each IP address, define s_1 to s_t such that $s_i = 1$ if the address occurs in source i and $s_i = 0$ otherwise. Then the string $s_1 s_2 \ldots s_t$ denotes the capture history of an IP address. The observed outcome of all measurements can then be represented by variables of the form z_s, which are the numbers of IPs with each capture history $s = s_1 s_2 \ldots s_t$. These are assumed to be instances of random variables Z_s.

Note that IPs with the capture history $00\ldots0$ are unobserved. Our goal is to use CR to estimate $Z_{00\ldots0}$. If $M = \sum_{s \setminus \{00\ldots0\}} Z_s$ is the total number of observed IPs, then the estimated population size is $\hat{N}_P = M + \hat{Z}_{00\ldots0}$. We assume the number of IP addresses in source i, N_i, is not secret for most sources. Then our private protocol described in Section 4 can be used to compute all Z_s, except $Z_{00\ldots0}$, based on the intersection cardinalities of combinations of sources and the known N_i. (Section 4.8 discusses approaches when some N_i are secret.)

Table 1 shows the case of $t = 3$, in which there are seven known capture counts $Z_{001}, Z_{010}, \ldots, Z_{111}$. For example, Z_{111} is the number of IPs captured by sources 1, 2 and 3 (say server logs, spam list and active probing), so Z_{111} is computed as the cardinality of the intersection of sources 1, 2 and 3. To compute the counts of IPs in only one source i we need to know N_i. For example, the number of IPs only in source 3 is $Z_{001} = N_3 - Z_{011} - Z_{101} - Z_{111}$.

Table 1: Example three-source capture history table

Source 1	Source 2	Source 3	Count
0	**0**	**0**	Z_{000} =?
0	**0**	**1**	Z_{001}
0	**1**	**0**	Z_{010}
0	**1**	**1**	Z_{011}
1	**0**	**0**	Z_{100}
1	**0**	**1**	Z_{101}
1	**1**	**0**	Z_{110}
1	**1**	**1**	Z_{111}

3.5. Lincoln-Petersen (L-P) Population Estimation

The L-P method works only with two sources and makes all four assumptions described in Section 3.3. Nevertheless, it provides a useful introduction to CR due to its simplicity.

Given a first source with Z_A sampled IPs, the size of the population would be known if we knew what *fraction* of the population had been sampled. To estimate this, L-P uses a second source of Z_B sampled IPs. Of these, Z_{11} IPs occur in both samples, $Z_{10} = Z_A - Z_{11}$ occur only in the first and $Z_{01} = Z_B - Z_{11}$ occur only in the second. If the fraction of "recaptured" IPs in the second sample equals the fraction of the total population captured in the first sample, then the population N can be estimated by [12, 13]:

$$\frac{Z_{11}}{Z_B} = \frac{Z_A}{N} \ , \quad N = \frac{Z_A Z_B}{Z_{11}} = \frac{Z_{01} Z_{10}}{Z_{11}} + Z_{01} + Z_{10} + Z_{11} \ .$$

3.6. Log-linear CR Models

Just as L-P uses a second sample to estimate the fraction of the population of the first sample, a third sample can be used to compensate the correlation between the first two samples (and this approach can be generalised to more than three samples). One way to use this additional information is to fit log-linear models (LLMs) [17, 19], which can model (apparent) source dependence among arbitrarily many sources. For each history s, let $h(s)$ be the set of samples in which an IP address occurs – for example, $h(101) = \{1, 3\}$. Define the indicator function $\mathbf{1}_A = 1$ if statement A is true and 0 otherwise. We can now write the following system of equations in 2^t variables $u, u_1, u_2, \ldots, u_t, u_{12}, \ldots, u_{1t}, u_{23}, \ldots$ up to $u_{12\ldots t}$:

$$\log(\mathbb{E}\,(Z_s)) = \sum_{h \subseteq h(s)} u_h = \sum_h u_h \mathbf{1}_{h \subseteq h(s)} \ .$$

For example, for $t = 3$, the system is

$$\begin{aligned}\log\left(\mathbb{E}\left(Z_{ijk}\right)\right) =& u + u_1\mathbf{1}_{i=1} + u_2\mathbf{1}_{j=1} + u_3\mathbf{1}_{k=1} \\ &+ u_{12}\mathbf{1}_{i=1\wedge j=1} + u_{13}\mathbf{1}_{i=1\wedge k=1} \\ &+ u_{23}\mathbf{1}_{j=1\wedge k=1} + u_{123}\mathbf{1}_{i=1\wedge j=1\wedge k=1}\ .\end{aligned}$$

The estimate of $Z_{00\ldots0}$ is then $\hat{Z}_{00\ldots0} = \exp(u)$. If we take $\mathbb{E}[Z_s] = z_s$ then this system has 2^t unknowns but only $2^t - 1$ equations, as $Z_{00\ldots0}$ is unknown. Hence it is customary to assume $u_{12\ldots t} = 0$ [17]. The u_h model the apparent dependencies between sources (also referred to as model parameters that describe source "interactions"). Using all u_h usually results in over-fitting. The final model ("best model") is selected by trading off between the model complexity and a goodness of fit measure [10]. Model complexity is reduced by forcing some u_h to 0, which treats sources as conditionally independent. For example, appearance of an IP address in a server log may be independent of its responding to active probing, conditioned on knowing whether or not it is in the spam black list.

3.7. Direct Models of Heterogeneity

Apparent source dependency is caused by the heterogeneity between different hosts. An alternative is to use a latent class model (LCM) [21], which assumes that there are C classes of hosts, and the N_c hosts within each class c have approximately equal probability θ_{ct} of being captured in each data source t. For example, classes may match "servers", "home PCs" and "corporate PCs". Note that this technique does not require the classes to be chosen in advance, and does not provide an interpretation for the classes it chooses. For a single class c, the probability of observing a particular $Z_c = (Z_{c,00\ldots0}, \ldots, Z_{c,11\ldots1})$ comes from the appropriate multinomial distribution, and is

$$P(Z_c; N, \theta) = \frac{N_c!}{(N_c - D_c)! \prod_{s\in\mathbb{P}} Z_{cs}!} \prod_{t=1}^{T} \theta_{ct}^{x_{ct}} (1 - \theta_{ct})^{N_c - x_{ct}} \tag{1}$$

where x_{ct} is the total number of observations of class c in source t, D_c is the total number of distinct observations in class c and $\mathbb{P}$ is the set of all capture histories except 00...0. However, the x_{ct} and D_c must also be estimated, by nominally allocating each observation to one of the classes. A locally maximum likelihood estimate can be obtained by the EM algorithm [21]. This algorithm alternately performs an M-step in which (the log of) the probability $P(Z_c; N, \theta)$ is maximised with respect to the distribution parameters θ_{ct} and N_c, and an E-step, which re-estimates the number sampled from each class, Z_{cs}, as the means given the current estimates of the distribution parameters. With the approximation $\log(N!/(N - D)!) \approx \int_{N-D}^{N} \log(y)dy$, the EM algorithm for (1) becomes

$$\begin{aligned}\text{M-step} \quad & \begin{cases}\theta_{ct} &= x_{ct}/N_c \\ N_c &= D_c / \left(1 - \prod_{t=1}^{T}(1 - \theta_{ct})\right)\end{cases} \\ \text{E-step} \quad & Z_{cs} = D_{.,s} \frac{\Theta_{cs}}{\sum_{c'} \Theta_{c's}}\end{aligned}$$

where $D_{\cdot s}$ is the total number of IPs with observation history s, and $\Theta_{cs} = N_c \prod_{t=1}^{T} \theta_{ct}^{1(t\in s)}(1-\theta_{ct})^{1(t\notin s)}$ is the expected number of IPs of class c with observation history s under the current model. The M-step calculates both θ_{ct} and N_c given the partition of observations into classes, say by repeated substitution, and the E-step re-partitions the observations by calculating Z_{cs}, from which $x_{ct} = \sum_{s:s_t=1} Z_{cs}$ and $D_c = \sum_{s\neq 00\ldots 0} Z_{cs}$.

There are many classes of Internet users, but two problems arise when using large C. First, the error surface becomes corrugated, and thousands of restarts of the EM algorithm are required to find the true maximum likelihood estimate. More subtly, a positive bias develops. Specifically, N_c is badly over-estimated when any one of the classes has too many observations that occur in only a single data source, relative to the number that occur in multiple sources. As the number of classes grows, it is increasingly common for this to occur in at least one of the classes, giving the bias. Chapman proposed an unbiased estimator for the single class case [22], but there appears to be no known unbiased estimator for multiple classes.

Note again that all of these CR models require only the sizes of the n-way intersections of the data sources, which can be found in a privacy-preserving manner using the methods of the following section.

4. Privacy-preserving Protocol

In this section, we first describe the goals and requirements for the protocol and the assumed adversary model. Then we provide a brief overview of the existing private set intersection cardinality protocols and select the one most suitable for our application as basis for our protocol (called the basic protocol). Next, we provide an overview of the basic pairwise private set intersection cardinality (PSIC) protocol which we show not to be scalable without sampling. Then, we introduce a more scalable version based on sampling. Next, we discuss the security and complexity of the protocol. Finally, we briefly describe two protocol extensions – one can be used to defend against so-called probing attacks, and the other is a partial solution to hide the sizes of datasets. We will also refer readers to our extended technical report for supplementary details [23].

4.1. Goals and Requirements

The protocol must ensure that no IP addresses are revealed to collaborators. However, we assume that usually the sizes of datasets can be revealed to collaborators (in Section 4.8 we discuss a partial solution to hide sizes). It must be possible to run the protocol repeatedly without reducing its security. For example, we may want to run the protocol every month or so with the latest data to estimate the population trend over time. While our main goal is to get a total estimate of the used space, ideally we also want to estimate different sub-populations, such as different geographic regions.

Our protocol works with sub-populations, but in this case some information leakage is inevitable. The data sources can be stratified by all collaborators before executing the protocol and then population estimates can be computed separately for each stratum. For example, IP addresses can be grouped by their allocating RIR to compute regional estimates. This stratification leaks how many addresses were observed by a collaborator in each region. We think the leakage is tolerable if the stratification is coarse – the collaborators decide a priori on an acceptable level of stratification.

The protocol must: (a) not rely on additional trusted third parties, (b) work with two or more parties (but the maximum number of parties would be relatively small, say 5–10 parties[2]), and (c) work with large datasets of over a billion IPs. Not all parties may want to compute the intersections, but at least one party must be able to do so. The protocol must be computationally-secure and it must be easy to verify its security.

4.2. Adversary Model

We assume that all collaborators participating in the protocol are potential honest-but-curious adversaries; they run the protocol correctly, but they try to learn as much information as possible. We also assume that one or more honest-but-curious adversaries may collude in order to obtain more information. With a small number of collaborators, none of which is anonymous, security under the honest-but-curious adversary model is sufficient. Our protocol can be protected against man-in-the-middle or eavesdropping attacks by running it over properly configured secure communication protocols, such as Secure Shell/Copy (SSH/SCP) or Transport Layer Security (TLS).[3]

4.3. Protocol Selection

Several techniques for the computation of private set intersection and private set intersection cardinality exist. For space reasons we refer to Pinkas *et al.* [24], which provide a recent overview of existing techniques grouped by public key / commutative encryption approaches, e.g. Vaidya *et al*. [25], techniques based on Yao's garbled circuits, e.g. Bellare *et al*. [26], and techniques based on Oblivious Transfer (OT), e.g. Pinkas *et al*. [24]. Pinkas *et al.* compared the performance of the different techniques and also developed their own efficient OT-based scheme [24].

There are three criteria for protocol selection: communication complexity, computational complexity and algorithm (i.e., code) complexity. In our scenario communication complexity is most important, as IP datasets are very large and need to be transported over the public Internet. Algorithm complexity is also important, as the simpler the algorithm is, the easier it is to convince parties to join the scheme (as they can more easily verify the security of the code). Computational complexity is not of primary concern as in our case the computations are done infrequently (e.g. once each month).

All garbled circuit schemes have very high communication complexity, two orders of magnitude above that for public key and OT schemes [24], and the amount of data exchanged is excessive for our scenario. Public key and OT schemes have similar communication overhead – OT schemes are better for 80 bit or higher security, but public key schemes are better for lower security. OT schemes perform computation much faster than public key schemes, but public key schemes are fast enough for our purposes.

[2] We assume a relatively small number of parties as the CR models do not scale well with the number of datasets in terms of computational complexity. Also, our protocol does require some initial manual setup, so a large number of parties is impractical.

[3] Note that the "no additional trusted third parties" requirement does not apply to SSH or TLS. SSH or TLS may require trusted third parties, i.e. certificate authorities, but these trusted third parties already exist and do not have to be created as in the case of our protocol.

Also, note that the higher performance of OT schemes comes at the price of high memory requirements. According to [24] a machine with 4 GB of RAM could only handle datasets of 2^{18} entries, but our datasets are 3–4 orders of magnitude larger than that (SeFaSI can process datasets of over a billion entries with less than 4 GB; see Section 5.5). Finally, public key schemes have the lowest algorithm complexity and implementations of these are the easiest to verify.

We opted to use a public key scheme as basis for our protocol as it best meets our requirements.

4.4. Basic Protocol

Our basic protocol is based on the commutative encryption approach [25]. Our protocol differs from the protocol in [25] in a number of respects, such as it computes all combinations of intersections and not just the intersection between all sources, it does not use dummy elements to mask dataset sizes (as that would make it impossible to compute the capture histories), it deals with multisets that break the security of [25], it uses sampling as [25] does not scale with the number of items, and optionally our protocol provides protection against probing attacks. While our protocol can use the same ring topology as in [25] we also propose a star message-passing topology which is more suitable for our application. (Note this is distinct from the underlying network topology, which is arbitrary.)

Below, we recall the concept of commutative encryption – the basis of the protocol. Then, we describe the basic protocol for two parties before extending it to multiple parties.

4.4.1. Commutative encryption

Let E be an encryption function and K_i be the secret encryption key of party i. Let $E_{K_i}(m)$ denote the encrypted version of plaintext m with key K_i. Then E is commutative if $E_{K_i}\left(E_{K_j}(m)\right) = E_{K_j}(E_{K_i}(m))$ for all m, i and j. We say E is computationally secure if decryption requires the secret key, it is resistant to plaintext attacks and it is collision-resistant, unless a computationally infeasible problem is solved. Specifically:

- Given $E_{K_i}(m)$ without K_i it is computationally infeasible to derive m (even if the set of possible m is small).
- Given m and $E_{K_i}(m)$ it is computationally infeasible to derive K_i, for all m and K_i.
- Given two plaintexts m_1 and m_2 it is computationally infeasible to find keys K_i and K_j such that $E_{K_i}(m_1) = E_{K_j}(m_2)$.

Two equivalent commutative encryption schemes are Pohlig-Hellman (PH) [27] and commutative RSA (cRSA) [28]. The encryption function is

$$E_{K_i}(m) = m^{K_i} \pmod{p} \ ,$$

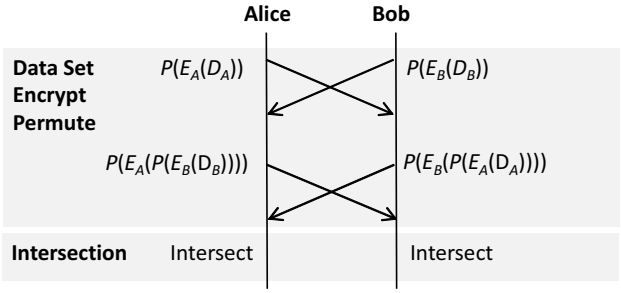

Figure 1: Basic protocol for two parties. Here P and E denote the permutation and encryption of the dataset D created by (A)lice or (B)ob.

where the modulus p is a large safe prime number[4] shared by all parties and where $\gcd(K_i, p-1) = 1$. It is easy to see that this function is commutative since

$$\begin{aligned} E_{K_i}\left(E_{K_j}(m)\right) &= m^{K_i K_j} \pmod p \\ &= m^{K_j K_i} \pmod p = E_{K_j}(E_{K_i}(m)) \ . \end{aligned}$$

From [27, 28] we know that this function is computationally-secure if the key and modulus are sufficiently large (for secure key and modulus sizes see [29]). Note that the selection of key and modulus sizes also provides a simple and convenient way to trade off security against overhead.

4.4.2. Two-party protocol

Let the two parties be (A)lice and (B)ob with datasets D_A and D_B. Both D_A and D_B should be sets and not multisets but our protocol also enforces this constraint. The algorithm has three main steps:

1. Configuration (agree on parameters)
2. Dataset encryption and permutation (main step)
3. Intersection cardinality computation

For each party i, let E_i be the encryption function, which maps a sequence of plaintexts (IPs) to the sequence of ciphertexts (encrypted IPs). Let P be a procedure that maps a sequence of values to a random permutation of that sequence.[5] Figure 1 shows an overview of the protocol.

In Step 1, A and B negotiate the configuration: the key and modulus sizes (depending on the level of security required), and the modulus value used for encryption (which must have the properties described in Section 4.4.1). Then, A and B each generate their own secret encryption key K_i independently.

[4] A safe prime is a prime of the form $p = 2q + 1$, where q is also a prime.

[5] Note that P is not a mathematical "permutation function", since the permutation can depend on its argument, or even be non-deterministic.

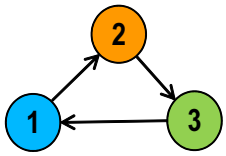

Figure 2: Ring topology for three parties – encrypted and permuted datasets are only passed in one direction

In Step 2, A and B encrypt and permute their own datasets, and then send the encrypted permuted sets to each other. A sends $P(E_A(D_A))$ to B, and B sends $P(E_B(D_B))$ to A. Then, A and B filter out any duplicate ciphertexts in the received encrypted datasets. This step is completed by the parties returning the double-encrypted datasets to each other: A sends $P(E_A(P(E_B(D_B))))$ to B, B sends $P(E_B(P(E_A(D_A))))$ to A.

In Step 3, A and/or B compute the intersection cardinality by counting the number of ciphertexts that are present in both double-encrypted sets. Since the encryption is commutative, any IPs that are in both sets will have the same ciphertexts in both double-encrypted datasets. Then, A and B know the sizes of the datasets and intersections, and can compute the capture histories.

4.4.3. Multi-party protocol

The scheme works with more than two parties as follows. Let k be the number of parties. Each party i has one[6] dataset D_i with $N_i = |D_i|$ IP addresses and a private encryption key K_i. Define a "*fully-encrypted*" data set to be the result of all parties successively applying their encryption to each of the IP addresses in one party's dataset. Parties form a unidirectional ring topology, as shown in Figure 2 for three parties.

In Step 1, all parties agree on the parameters listed in the two-party protocol description, and each party chooses its secret K_i.

In Step 2, each party encrypts the IPs of its own dataset using K_i, randomly permutes the encrypted IPs, and then passes the encrypted permuted dataset to the next party. The next party encrypts and randomly permutes the received dataset with its own K_i, passes it to the next party that has not yet processed this dataset, and so on until all datasets are fully-encrypted.

In Step 3, each party sends its fully-encrypted dataset to all other parties interested in the intersection. Then all interested parties can perform the intersection cardinality computation. Since E is commutative, for each combination of sources, the cardinality of the intersection of the ciphertexts is identical to the cardinality of the intersection of the plaintexts. This allows all interested parties to compute the capture histories.

The ring topology has the disadvantage that it requires many relationships between different parties. This can be addressed with a star topology, as shown in Figure 3 for four parties. With a star topology each party only needs a relationship with the hub and the hub controls the passing of the datasets to ensure each dataset is encrypted by every party. The hub has increased communication complexity, but we assume this is not a

[6]For simplification we assume one dataset per party, but in practise each party can have multiple datasets.

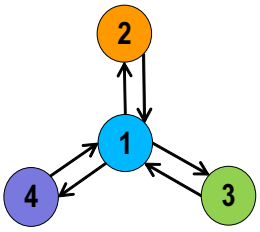

Figure 3: Star topology for four parties – encrypted and permuted datasets are passed via party 1

problem as the hub is also the party most interested in the intersection results (e.g. the researchers).

Note that in our case the ring or star topologies are *logical* topologies. Redundancy can be employed for both the PCs that handle the communication and computation and the network connections to guard against accidental failures of any of the nodes, including the hub. However, if any party chooses to drop out of the protocol during the operation of the protocol, the protocol will fail and will have to be rerun with the remaining parties. The rerun could be optimised by using some partial results obtained until the party dropped out, but this is left for future work.

4.4.4. Lack of Scalability

The basic protocol does not scale to large datasets. Encrypting each IP separately leads to significant space overhead. For example, assuming a modulus of 1024 bits, an encrypted list of a billion IPv4 addresses consumes 128 GB (storage and communication overhead). The computational overhead is also high, since the exponentiation-based encryption function (Section 4.4.1) is much slower than commonly used symmetric encryption techniques. Recall that the participating parties are often companies who have datasets, but limited incentive to collaborate. Minimising the disincentives (such as storage, network and computational costs) to collaborate is crucial to maximising the number of potential collaborators.

4.5. Protocol with Sampling

To make the protocol scale to datasets with large numbers of IP addresses, we propose that initially each party generates a sampled dataset of smaller size. However, simple random sampling cannot sample entries consistently across the different datasets of all parties. We resolve this by using hash-based sampling [30, 31]. The cardinalities of the intersections between the original datasets can be estimated based on the intersection cardinalities of the sampled datasets. We discuss in [23] how to choose the sample rate.

4.5.1. Hash-based sampling

The goal of our hash-based sampling is to sample randomly from multiple IP datasets so that if an IP address is selected, it is selected from all datasets that contain that address.

Let H be a good integer-valued hash function – one that generates different output even for very similar input and maps the inputs as uniformly as possible over its output range, which we denote $[0, R_M]$. We do not require cryptographic properties. Let the hash function be salted [32] (to obtain different independent samples for the same input) by appending a random "salt" s to the hash input m before applying the hash function. Also, choose integers r and R such that $0 < r \leq R \leq R_M$ and $R_M(\mod R) \ll R_M$. We can then sample the same elements from different lists at rate approximately[7] r/R by selecting only the elements with

$$H(m \oplus s) \pmod R < r \ ,$$

where $\oplus$ denotes string concatenation.

4.5.2. Multi-party protocol with sampling

In Step 1, all parties also need to agree on H (which must have the properties as described in Section 4.5.1), s, and the sample rate $p_D = r/R$. The sample rate p_D must be large enough to prevent intended or accidental probing attacks, but our optional probing attack detection also prevents these (see Section 4.9). We are not aware of any further attacks an attacker could mount by influencing the choice of H, s, and p_D. However, as precaution we recommend that the salt s is computed in a shared fashion, e.g. each party contributes some bits, to prevent one party from controlling which IPs are sampled (a mechanism that is simple to implement).

In Step 2, each party hash-samples its own dataset before executing Step 2 as described in Section 4.4.

In Step 3, A and B compute an estimate of the intersection cardinality. Let $C = |D_1 \cap D_2 \cap \ldots \cap D_k|$ denote the cardinality of the intersection of the datasets, $\tilde{D}_i \subseteq D_i$ the sampled datasets, and $\hat{C} = \left|\tilde{D}_1 \cap \tilde{D}_2 \cap \ldots \cap \tilde{D}_k\right| / p_D$ be an estimator for C. Since the probability that an element of the intersection is in the sample is p_D, $\hat{C}$ is an unbiased estimator for C.

A confidence interval can be constructed as follows. The number of items X in a sampled set of size N follows a Binomial distribution with a mean of $\bar{n} = Np$ and a variance $\sigma_{\bar{n}}^2 = Np(1-p)$. Assuming N is large, we can approximate the Binomial distribution with a Normal distribution and omit the continuity correction [33]. Let $z_\alpha = \Phi^{-1}(1-\alpha)$ where Φ is the cumulative distribution of the standard Gaussian (with $\mu = 0$ and $\sigma = 1$). Then

$$\Pr\left(\bar{n} - z_{\alpha/2}\sigma_{\bar{n}} \leq X \leq \bar{n} + z_{\alpha/2}\sigma_{\bar{n}}\right) = 1 - \alpha \ , \tag{2}$$

which states that with a probability of $1-\alpha$ the size of the sampled dataset is between $\bar{n} - z_{\alpha/2}\sigma_{\bar{n}}$ and $\bar{n} + z_{\alpha/2}\sigma_{\bar{n}}$.

By (2) a two-sided confidence interval for C is[8]

$$\Pr\left(\hat{C} - z_{\alpha/2}\sigma_{\hat{C}}/p_D \leq C \leq \hat{C} + z_{\alpha/2}\sigma_{\hat{C}}/p_D\right) = 1 - \alpha \ . \tag{3}$$

where $\sigma_{\hat{C}}^2 = Cp_D(1-p_D) \approx \hat{C}p_D(1-p_D)$.

[7] The rate is exact if R is a factor of R_M and the hash function maps perfectly uniformly onto $[0, R_M]$.

[8] C is unknown; we use its estimate $\hat{C}$ to get an estimated standard deviation.

4.6. Security

In Step 1 there are no direct security issues as no datasets are exchanged. If the protocol parameters are chosen as described in the previous sub sections, e.g. the key sizes are appropriately chosen, then the security of the later steps is not compromised.

In Step 2 all parties exchange the encrypted and permuted IP datasets. Since the encryption is computationally secure, no party can decrypt another party's dataset without knowing the other party's encryption key. This includes finding the plaintexts for other parties' ciphertexts by brute-force (encrypting the whole IP address space and finding matching ciphertexts). The random permutation prevents any party from knowing which IPs map to which ciphertexts in any (encrypted) version of its own dataset encrypted by another party. Since in order to do step 3 each party's dataset must be encrypted by at least one other party, the permutation thus prevents any party from detecting the presence of its IPs in other parties' datasets.

In Step 3 the only information the parties can learn are the sizes of intersections of different datasets and the sizes of the datasets, since the encryption is computationally-secure and no party can map any ciphertexts to IP addresses.

The protocol is protected against third parties by using properly configured secure communication protocols, such as SSH/SCP or TLS.

An attacker could construct a dataset where certain IPs occur multiple times and the counts of these selected IPs are unique numbers. For example, only one specific IP occurs twice, while all other IPs are present once. Since the counts are preserved in the fully-encrypted datasets, this would create a side channel allowing the attacker to probe the existence of certain IPs in other parties' sets. Our protocol prevents this attack by filtering out duplicate ciphertexts in received datasets before performing any other actions.

An attacker could also try to probe for existence of certain IPs in another party's dataset. This can be prevented by an optional extension of the protocol described in Section 4.9.

4.7. Complexity

Since every party needs to encrypt and permute the datasets of all parties, the total computational complexity of the scheme is $O\left(k^2N\right)$, where k is the number of parties and N is the average size of the datasets. However, for a single party the complexity is linear $O\left(kN\right)$. In the ring topology the communication complexity is $O\left(k^2N\right)$ overall and $O\left(kN\right)$ for each party. In the star topology the communication complexity is $O\left(2k^2N\right)$ overall, $O\left(kN\right)$ for non-hub parties and $O\left(k^2N\right)$ for the hub.

4.8. Hiding Dataset Size

Above we assumed that the sizes of the datasets can be revealed, but there may be cases where the sizes of some datasets should be kept secret. Vaidya and Clifton [25] proposed to hide the size by padding a dataset with dummy values (random strings disjoint from the item space). However, this does not work with CR where we need to know the number of IP addresses observed only by a certain source, which we can only compute if the dataset size *is known* (see Section 3.4). We propose a partial solution if some parties want to hide their dataset size.

If one or more parties are willing to divulge the sizes of their datasets (or even the contents) to another party, then merging multiple datasets prior to running the privacy-preserving protocol algorithm can hide the dataset sizes from non-trusted parties while allowing CR computations. Merging datasets also obfuscates trends in the intersections of unmerged datasets. Let us consider two cases.

In the first case, one of the parties, say A, is willing to divulge its actual dataset to party B. Party B then simply combines its own dataset(s) with A's dataset, and A does not take part in the protocol. Since the encryption and permutation is irreversible and assuming the merged datasets have roughly similar sizes (enforceable by dataset subsampling) unknown to other parties, the original sizes of the merged datasets cannot be recovered.

In the second case, neither A nor B is willing to divulge actual data, but both are willing to divulge the sizes of their datasets to each other (but not to third parties). In this case A and B initially perform the two-party protocol with each other to obtain double-encrypted versions of their datasets, which they then combine. Then when running the protocol with other parties, A and B each send this double-encrypted combined dataset as their own dataset in the first round of Step 2. When party A receives the dataset initially sent by party B, it does not re-encrypt it, but simply forwards a permuted copy. Party B does the same when receiving A's dataset. At the end of the process, each dataset will still have been encrypted exactly once by each party.

For CR, mergers should only occur between datasets with similar expected biases, such as client IP addresses collected by different web sites. Still, the merging may result in some information loss for CR.

If datasets cannot be linked to parties just based on their sizes, another mitigation technique is to anonymise dataset ownership (see description in [23]).

4.9. Probing Attack Prevention

The preceding protocol cannot resist probing attacks, where one party generates datasets with mostly invalid IPs to test whether a few valid IPs are in another party's dataset. Since no party can decrypt the fully-encrypted datasets, it is impossible to check whether the original data were valid IPs. Reasons for mounting a probing attack are to learn the number of active addresses and whether some specific IP addresses are likely to be in another party's dataset. At worst, an attacker could discover the precise address of active hosts and something about the structure of an organisation's network, thus breaching our protocol's security.

A technique to prevent probing attacks for $k > 2$ was presented in [25]. However, this approach cannot distinguish between probing and legitimate datasets with small overlap, and is unusable for CR where we have small intersections. We propose a novel defence, which can be applied to all situations where the set of permitted (valid) items is known and is not prohibitively large. In this scheme, all parties provide a dataset of a minimum size and agree on a *valid set* of IPv4 addresses. All parties create a fully-encrypted version of the valid set (using the same keys as for dataset encryption) which is then used by each party to check that a fully-encrypted dataset consist mainly of valid IP addresses (before forwarding the dataset to any other party).

The minimum dataset size prevents obvious probing attacks and accidental probing with small datasets. The valid set prevents probing attacks with crafted datasets that

mainly contain invalid IP addresses and only a small number of IP addresses to be probed. A detailed description is in [23].

5. Prototype Evaluation

Now we describe our prototype implementation, describe the datasets used for the evaluation (and analysis in Section 6), analyse the accuracy of CR, analyse the impact of dataset sampling and the prototype's performance.

5.1. Implementation

We implemented a publicly available open source prototype (SeFaSI) [11] based on a mix of tools written in Python and C. The prototype consists of separate tools implementing basic functions, such as sampling or encryption, and a "main" tool that implements the privacy-preserving protocol using the basic tools. SeFaSI creates the capture histories that are used as input by our CR population estimation tools written in R and Matlab.

SeFaSI implements the whole protocol in Python, since Python source code is small and readable, allowing all parties to easily verify the security of the implementation. We also implemented faster versions of the sampling, encryption, and set intersection cardinality computation tools in C as optional substitutes for the Python equivalents. Encryption is based on the PyCrypto library's RSA functions [34] and libopenssl's modular exponentiation function [35]. We use the Murmur hash function [36] for sampling as it is fast and has a good distribution that passes the usual tests for hash functions.

The results in Section 5.5 show that SeFaSI's performance is sufficient for practical use. Future work will include improvements, such as multi-threading support for parallel sampling or encryption.

5.2. Datasets

Table 2 summarises the datasets we collected from multiple sources of unique IPv4 addresses between July 2013 and December 2014.[9] We actively probed the whole allocated IPv4 Internet using ICMP echo requests (IPING) and TCP SYN packets to port 80 (TPING). Passively observed IPv4 data includes addresses from Wikipedia's page edit histories[10] (WIKI), potential spam email senders from [37] (SPAM), addresses of clients tested by Measurement Lab [38] tools (MLAB), web clients participating in an IPv6 readiness test [39] (WEB), anonymised server logs of game clients connecting to Valve's Steam online gaming platform (GAME), and NetFlow records from *incoming* traffic of Swinburne University of Technology's access router (SWIN)[11]. The pre-processing of the datasets is described in more detail in [10].

[9] The datasets were collected as part of the Australian Research Council-funded MAPPING project (http://caia.swin.edu.au/mapping/), which finished at the end of 2014.

[10] Modification times and IPv4 addresses of edits by unregistered users.

[11] Excluding all traffic flows of our active prober.

Table 2: Datasets, collection time window, and number of unique IPv4 addresses and /24 subnets

Dataset	Description	Time Window	Unique IPv4 [M]	Unique /24 [M]
WIKI	Clients editing Wikipedia	Jul 2013 - Dec 2014	8.8	2.5
SPAM	Potential spam email senders	Jul 2013 - Dec 2014	19.2	1.8
MLAB	Clients tested [38]	Jul 2013 - Dec 2014	27.0	3.0
SWIN	Swinburne access router NetFlow	Jul 2013 - Jul 2014	73.9	3.3
TPING	TCP port 80 census of IPv4 Internet	Jul 2013 - Dec 2014	124.2	4.2
WEB	Web clients observed [39]	Jul 2013 - Dec 2014	150.4	4.5
NFLIX	Netflix clients	Jul 2013 - March 2014	202.6	2.1
GAME	Online game clients	Jul 2013 - Dec 2014	conf	4.8
IPING	ICMP ping census of IPv4 Internet	Jul 2013 - Dec 2014	541.5	5.4

Since [10] we have added a significant new dataset based on encrypted IP addresses of clients connected to Netflix (NFLIX). Our privacy-preserving technique allowed Netflix to contribute their IP addresses without violating their policies. We used SeFaSI with a key size of 160 bit, a modulus size of 1024 bit and a sample rate of 100% (a Netflix employee visited our research centre, minimising communication overhead).

The only dataset that can include spoofed IPs is SWIN, and we used the heuristic from [10] to eliminate potential spoofed addresses. We generate datasets of unique /24 subnets by processing the IPv4 datasets and setting the last octet of each address to zero and then filtering out the duplicates. Table 2 shows the sizes *after* pre-processing. For GAME the number of IP addresses is confidential, but it is a big dataset as the large number of /24 suggests.

5.3. Validation against Ground Truth

As with any real-world CR application, it is not possible to validate the overall accuracy of our CR approach, since we do not know the ground truth. However, we can compare the observed and estimated IPv4 addresses, from our IP data sources described in Section 5.2, with the ground truth known for several networks. For privacy reasons we cannot reveal the identity of the networks (and use letters A–G as identifiers). The largest network is two /16 subnets and the smallest network is roughly one /20 in multiple allocations. For most networks, our ground truth is the number of actively used IPv4 addresses at peak times (high watermarks). We compare this number against the observed and estimated numbers of IPv4 addresses for a 12-month time window, where the high watermarks occurred between the middle and the end of the window. For networks B and G we know the actual numbers of unique addresses used in the time window.

For each network, Figure 4 shows the number of addresses that responded to ICMP ping, the number of addresses observed, the number of addresses estimated with LLM using a truncated Poisson distribution and the Bayesian Information Criterion [10] (vertical bars), and the ground truth (horizontal lines), all as percentages of the sizes of the

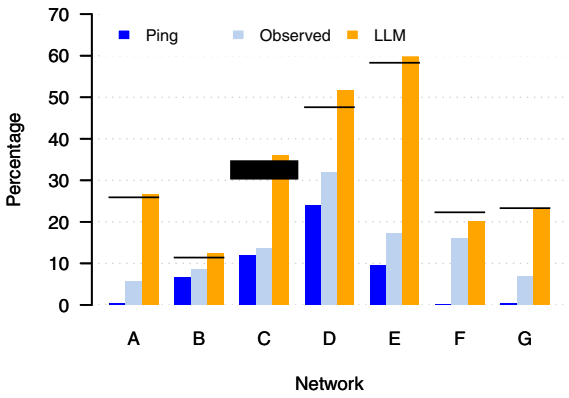

Figure 4: Comparison of LLM estimates, observed and pingable addresses (vertical bars) against the number of reported used IPv4 addresses (horizontal lines)

networks. Note that for network C we only have a range for the ground truth, and network F blocked our active probing, so we do not have ping data for network F.

The results show that the LLM estimates are close to the ground truth – *much* closer (up to 10 times closer) than the number of pingable or observed addresses. Notably the difference is smallest in cases where the ground truth corresponds closely to what we estimate (B, G). The number of observed addresses is relatively close to the ground truth for some networks (B, F), but far below the ground truth for most networks. However, the number of observed addresses is still a much better metric than the number of pingable addresses, especially for heavily firewalled networks (A, G) or networks that block pinging (F). Note that for each network, at least five of our data sources provide substantial numbers of addresses, which allows CR to work.

5.4. Sampling Error

Here we analyse the impact of the sampling on the LLM CR population estimates. In [23] we present a more detailed analysis and also verify that the sampling error of our prototype is consistent with (3).

For each sampling rate, we repeatedly sampled all data sources, each time with different sample salts (100 runs for each sampling rate). Then, we computed the LLM CR estimates from the sampled data sources and compared them against the CR estimates for the unsampled data sources we treat as true values. The estimated range is determined based on the profile likelihood confidence interval [40] and we used Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC) [41] for fitting.

We first investigate the case where the "best model" is selected in each run from the sampled data. This reflects the case where we never have the unsampled data and hence do not know the best model based on the unsampled data. Figure 5 shows the relative errors for the LLM estimates of the total number of addresses (lower and upper values of LLM estimated ranges) for different sampling rates with AIC. The relative errors are below 5%, and with sampling rates of 1% or higher the error is usually within

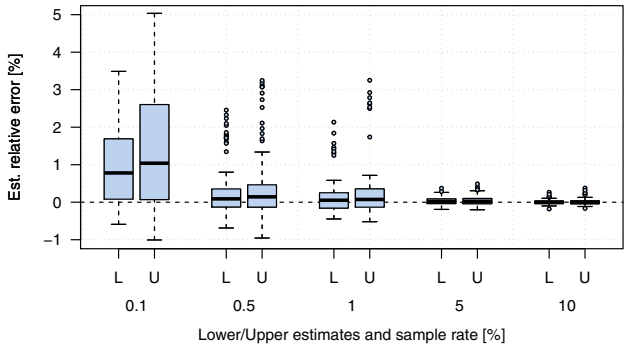

Figure 5: Relative error of (L)ower and (U)pper values of CR estimated ranges depending on the sample rate for LLMs with AIC, deriving the best model from the sampled datasets

2%. With the BIC, errors are usually below 2% even for sample rates as low as 0.1% [23].

The larger errors are mainly caused by model inconsistencies, since the model was selected independently for each set of sampled datasets. For sampling rates of 5% and 10% the selected models are quite consistent. For sampling rates of 1% and lower, models for different samples start to diverge. We can also see some bias in the LLM estimates. The bias is very small for sampling rates of 5% or higher, but increases significantly with decreasing sampling rate. It is caused by the systematic inclusion or exclusion of some parameters not used in the benchmark model built from the unsampled data. Since our data sources have mainly apparently positively correlated data sources, increasing (decreasing) the number of model parameters typically results in higher (lower) population estimates.[12] The effect is dominated by parameters that represent interactions between fewer sources.

To demonstrate that the bias in the LLM estimates is caused solely by the model selection process, we conducted another experiment where the model was fixed – in each run the model used was the model selected by AIC for the unsampled data. Figure 6 shows the relative errors for the LLM estimates of the total number of addresses (lower and upper values of LLM estimated ranges) depending on different sampling rates, treating the estimates for the unsampled data sources as true values. It shows that with a fixed model the LLM estimate is unbiased and has substantially smaller error.

5.5. Performance

We measured the performance of our prototype on a PC with an Intel i7 2.8 GHz CPU, 24 GB RAM and a file cache on a solid state disk (SSD) running FreeBSD 9.0 and Python 2.7.3. Note that for all performance measurements we used a single CPU core only and RAM used by SeFaSI never exceeded 1 GB. In this section we summarise

[12] Positively correlated data sources lead to underestimates with L-P. LLMs correct for this with positive model parameters. The more positive parameters the selected best model has, the higher is the estimate.

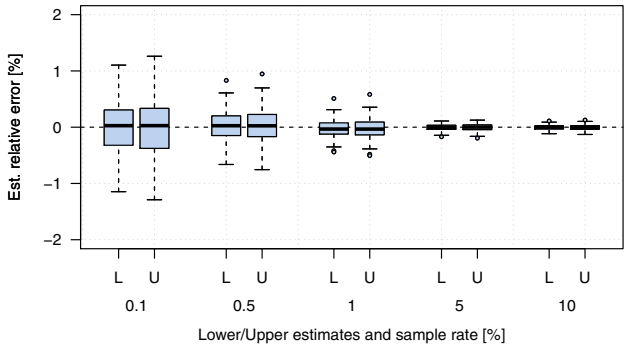

Figure 6: Relative error of (L)ower and (U)pper values of CR estimated ranges depending on the sample rate for LLMs with AIC, using the AIC model selected for the unsampled data

the results for the C implementation (averages over 10 runs); more details can be found in [23].

The sampling speed of our prototype depends on the sample rate (file-write overhead increases with higher sample rates). However, for realistic sample rates of 10% or lower, the speed is nearly constant in the range of 5.2 to 5.4 million IPs per second. Even with an unrealistically high sample rate of 50% our prototype can sample about 4.35 million IPs per second. Hence, it would take no more than 4 minutes to downsample a dataset with 1 billion IPv4 addresses.

The encryption and permutation speed depends on the key and modulus sizes. While it takes longer to encrypt already-encrypted IPs (due to longer input data), the difference is relatively small, which we attribute to libopenssl's modular exponentiation being the bottleneck[13]. With 160 bit keys and 1024 bit modulus (NIST 2010 Legacy [29]) our prototype can encrypt and permute 3,600 IPs per second. This reduces to 1,100 IPs per second with 224 bit keys and 2048 bit modulus. So with NIST 2010 it takes a little over 3 days to encrypt and permute 1 billion IPv4 addresses. However, with a dataset sample rate of 10% the time reduces to only 8 hours (including the sampling). The encryption and permutation speed is dominated by the encryption, which takes 97–98% of the time.

The set intersection cardinality speed depends on the number of encrypted datasets and the overlap. An overlap of 100% represents the best case (highest performance) and an overlap of 0% represents the worst case (lowest performance). Computing the set intersection cardinality of five datasets with 1 billion IPv4 addresses each takes 18–50 minutes depending on the overlap. However, with a dataset sample rate of 10% this reduces to 2–5 minutes.

[13] libopenssl implements modular exponentiation based on Montgomery multiplication, where the original multipliers are transformed into Montgomery space. Transformed multipliers are always of the size of the modulus in bits.

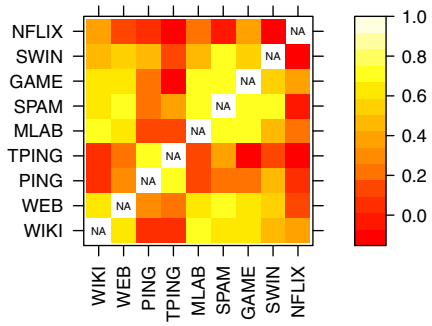

Figure 7: Estimated correlations between data sources

6. Estimated Used IPv4 Space

Now we illustrate our technique by presenting an estimation of the used IPv4 space based on the datasets described in Section 5.2. We are interested in not only the number of used IPv4 addresses, but also the number of used /24 subnets (the minimal allocation size).[14] Note that in cases where we stratified the data, a separate SeFaSI run was performed for each stratum (as mentioned in Section 4.1).

6.1. Correlations Between Datasets

We first demonstrate that no pair of our datasets are sufficient by themselves. Recall that CR using two datasets requires them to be uncorrelated. Figure 7 shows the estimated correlations between our data sources. These are Yule's Y computed based on 2x2 capture frequency tables for each source combination with Z_{00} estimated by CR. Most of the correlations are positive, although there are a few slightly negative correlations (TPING-GAME, TPING-NFLIX, SWIN-NFLIX, SPAM-NFLIX). TPING and IPING, which include servers, are strongly correlated with each other, but are less correlated with the other sources. TPING is less correlated with the client-centric passive sources than IPING is, as TPING requires an active service running. Most client-centric datasets (WIKI, WEB, MLAB, GAME) are highly correlated. SPAM also shows a high correlation with the client datasets (since most spam is sent by clients coerced into botnets). NFLIX shows less correlation with the other sources, possibly since it is more geographically limited.

6.2. Used IPv4 space

We now present our estimates for the used IPv4 space at the end of 2014. Unless otherwise noted results are for LLMs. For LLMs we are using the adaptive divisor approach and the BIC for the model fitting [10]. For LCMs we use a model with $C = 7$, chosen as a balance between unreliability due to many local optima for large C, and negative bias (indicated by leave-one-out cross validation) for small C. We exclude NFLIX for AfriNIC and African countries due to a lack of coverage.

[14] Until 2014 RIRs never allocated less than a /24 (256 addresses), e.g. see APNIC's policy [42].

Table 3: Observed and estimated used IPv4 addresses and /24 subnets at the end of December 2014

	Ping [M]	Observed [M]	Stratified	Est. Used LLM [M]	Est. Used LCM [M]	Unseen [M]	Routed [M]
IP addresses	542	842	No	1167	1239	300–400	2753
			RIR	1187	1219		
/24 subnets	5.4	6.3	No	6.6	6.4	0.1–0.4	10.8
			RIR	6.7	6.4		

6.2.1. Totals

Table 3 shows the total number of pingable, observed and estimated IPv4 addresses and /24 subnets for both LLM and LCM, as well as the number of publicly routed IPv4 addresses and /24 subnets for comparison. We show the estimates without stratification and when stratifying by RIR. The estimates for LLMs and LCMs are broadly consistent (both within 6% of the average of 1.2 billion addresses). With RIR stratification, the IPv4 address estimates of both models are closer, but the /24 subnet estimates are further apart. This indicates that further research into confidence intervals for these estimators will be worthwhile.

A potential benefit of LCMs over LLMs is that meaningful classes of users can sometimes be identified. One identified class appears to correspond to servers, since it frequently appears in TPING and seldom appears in client-based sources such as NFLIX and GAME, but no other classes stood out as clearly identifiable.

To get an idea of the error of the estimate, we computed the population estimate n times, each time leaving out one of the n sources and then computed the standard deviation for the n population estimates.[15] The standard deviation is roughly 110 million addresses. However, if we consider only cases that include the largest dataset, IPING, then it reduces to 45 million, less than 4% of the mean. Note that without NFLIX the standard deviation is about 60 million, so the inclusion of the new dataset reduced the variance of the estimate. For /24 subnets the n estimates are very consistent, and the standard deviation is only 65,000 subnets.

6.2.2. Usage by region

Figures 8 and 9 show the pingable, observed and estimated used IPv4 addresses and /24 subnets for the five RIRs at the end of 2014 as absolute numbers and percentages of the routed space. Note that *the top of each bar segment* indicates the number of pingable, observed and estimated used IPv4 addresses or /24 subnets. The results show that most /24 subnets are observed and CR estimates that the fraction of used but unseen /24 subnets is small (especially for LCMs). However, for IPv4 addresses the fraction of unseen IP addresses estimated by both CR methods is significant. It is expected that the

[15] This is more realistic than using the standard confidence interval methods for CR that likely underestimate the error due to our large sample size.

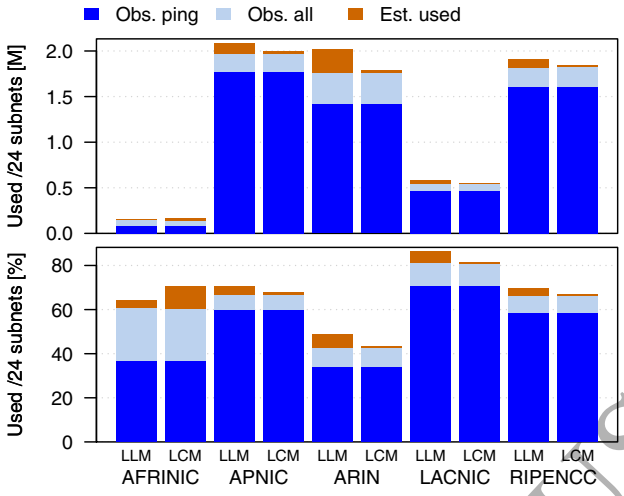

Figure 8: Pingable, observed and estimated /24 subnets at the end of 2014 for the different RIRs

fraction of unobserved addresses is larger than that for /24s, since the latter is a logical OR of all of the addresses it contains. For both IPv4 addresses and /24 subnets, APNIC (Asia-Pacific), ARIN (North America) and RIPE (Europe) have the highest numbers of used addresses/subnets, whereas LACNIC (Latin America) has the highest utilisation.

6.2.3. *Usage by allocation year*

Figures 10 and 11 show the pingable, observed and estimated used IPv4 addresses and /24 subnets depending on the allocation year (estimates for the LLM model only, since the overall trends for both models are similar). The allocation year of an IP address or a /24 is the year the IP or /24 was allocated based on the RIR allocation data.

The figures show that IP space allocated in the earlier period until and including 1998 has much lower utilisation than IP space allocated from 1999 onwards. This suggests a significant part of the IPv4 space allocated in the earlier years is either not used or underutilised – an observation consistent with the findings by Cai *et al.* [5]. The difference in utilisation between these two periods is more pronounced for IP addresses than for /24 subnets, which indicates that parts of the earlier allocated space are underutilised.

Furthermore, only 55% of the addresses allocated between 1982 and 1998 are publicly routed, whereas 94% of the IPv4 space allocated between 1999 and 2014 is publicly routed (based on Routeviews [43] data).

6.2.4. *Usage by country*

Figures 12 and 13 show the LLM estimated used IPv4 addresses for each country (with sufficient sample size) against the country's gross domestic product (GDP) and population size on a log-log scale for the different regions (RIRs). Results for /24s are

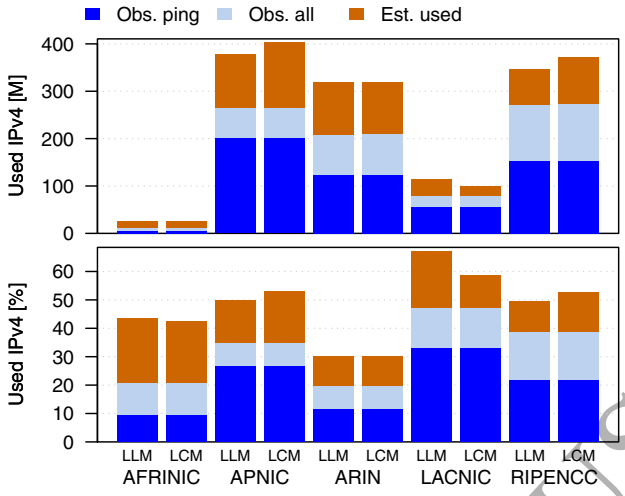

Figure 9: Pingable, observed and estimated IPv4 addresses at the end of 2014 for the different RIRs

broadly similar. The correlation can be estimated by the width of the minor component of the scatter plot. The correlation between the number of estimated used addresses and GDP is greater than the correlation with population. Especially for the African, Asian and South American regions, we see countries with large populations that use a comparatively small part of the IPv4 space. The is consistent with a strategy of allocating IPv4 addresses proportional to the demand for Internet access, which is correlated more with GDP than with population.

6.2.5. *Impact on IPv6 deployment*

Figure 14 plots the fraction of allocated used IPv4 addresses for each country against the percentage of hosts that are IPv6-capable according to July 2015 APNIC data [44] (the *x*-axis is logarithmic). Results for /24s are similar. Surprisingly, there is no correlation between the fraction of used IPv4 addresses (or /24 subnets) and IPv6 readiness. A plot against the percentage of hosts that prefer to use IPv6 over IPv4 also shows no correlation [23].

6.3. *Discussion*

Using only ping underestimates the number of used IPv4 addresses. With addresses from other data sources the number of observed used IPs increases from 550 to 850 million. However, with CR we estimate that the number of actually used IPs is much higher – around 1.2 billion addresses at the end of 2014. Without complete ground truth, which is impossible to obtain, we do not know the accuracy of this estimate, but based on samples of ground truth we have shown that CR estimates are usually much closer to the ground truth than the number of observed addresses. We also showed that the estimates of two very different CR models are broadly consistent, which indicates that there is not a high uncertainty in our estimates due to using one specific CR model.

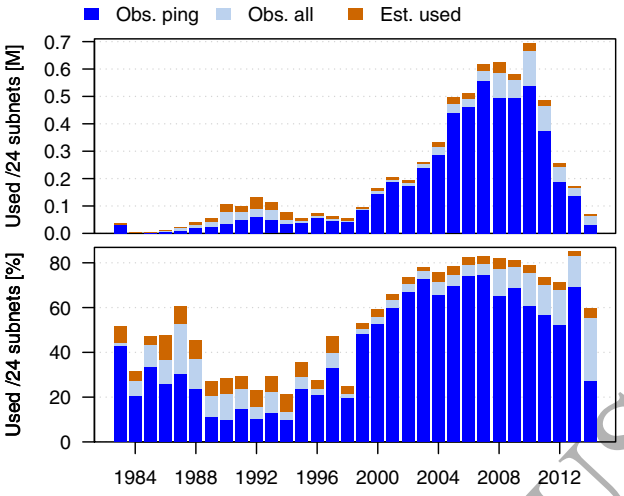

Figure 10: Pingable, observed and estimated /24 subnets depending on the allocation year

For the number of /24 subnets the CR estimates (6.4–6.7 million) are much closer to the observed number (6.3 million), and so the number of observed /24 subnets already appears to be a good indicator of /24 subnet usage. Again, the estimates of our two different CR models are broadly consistent.

When investigating the different regions based on the RIRs, for both IPv4 addresses and /24 subnets, APNIC, ARIN and RIPE have the highest numbers of used addresses/subnets, whereas LACNIC has the highest utilisation. When investigating the used space depending on when it was allocated, it is clear that IPv4 space allocated until 1998 is much less used than space that has been allocated after 1998.

Our CR estimates are likely below the actual used space due to the data sources we used. In future work we aim to include additional sources that could not be accessed previously without the privacy-preserving protocol. On the other hand we know that out of the 3.58 billion allocated addresses only 2.75 billion IP addresses were actually publicly routed at the end of 2014 [43]. Furthermore, there are clear signs of low utilisation in networks where we have ground truth (see Section 5.3) and older IP allocations (see Section 6.2.3). It appears that large parts of the IPv4 space were either not used or underutilised at the end of 2014.

The apparent shortage of IPv4 addresses is heavily influenced by fragmentation of the address space. IPv4 space is hierarchical due to the need for efficient management and routing. The RIRs (and prior to the RIRs the central Internet Registry) allocated large contiguous blocks of addresses (defined by address prefixes) to organisations, who then subdivided the blocks to allocate to smaller bodies, and so on.

Until 1993, prefixes had to be /8, /16 or /24, and allocations were often larger than needed, because initially the IPv4 space was perceived as plentiful and the strategy was to err towards shorter rather than longer prefixes so that organisations could grow within their allocated space. The problem with this approach was the huge effect of

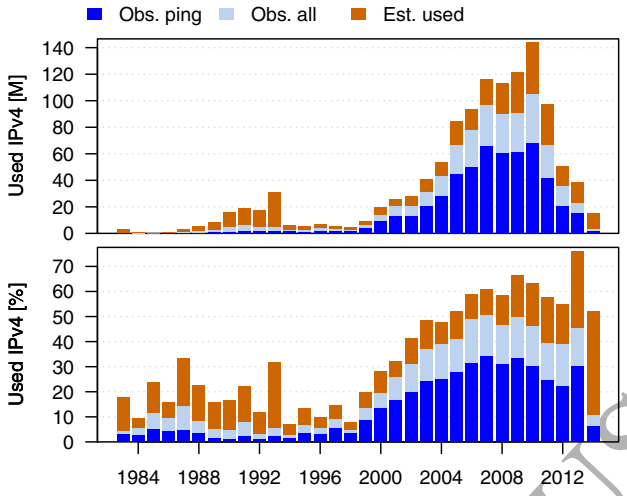

Figure 11: Pingable, observed and estimated IPv4 addresses depending on the allocation year

upsizing. For example, an organisation for which a /16 (65,536 addresses) was a little bit too small was allocated a /8 (16.7 million addresses). Demand for the Internet arose earlier in wealthier countries (consistent with Section 6.2.4), so these benefited more from the 'generous' allocation sizes handed out in early years. From 1993 onwards Classless Inter-Domain Routing (CIDR) removed the "three prefix sizes fit all" restriction, but the minimum allocation size of RIRs remained /24 or larger in order to limit the size of the global routing table. For example, until the year 2000 APNIC's minimum allocation size was /19 and it was only gradually reduced to /24 in the year 2011 [42].

The hierarchical structure, 'generous' allocation sizes given out in early years and the minimum allocation sizes cause many addresses to be unused at the end of 2014. Ongoing periodic analysis is required to observe longitudinal trends. Our proposed technique allows doing this while keeping observed IP addresses private.

7. Conclusions

A better understanding of IPv4 address space exhaustion, and likely pressures for IPv6 adoption, requires estimating how much allocated IPv4 space is *actively used*. As no single online service has complete visibility into IP address space utilisation, such estimates require diverse parties to share private datasets of active IP addresses. Sharing raw data can reveal a party's business scope and compromise a party's user base.

This paper has presented a new collaborative and privacy-preserving capture-recapture (CR) technique for estimating address space utilisation from multiple sources of observed IP addresses while guaranteeing the privacy of the addresses. Our technique is much more accurate than assuming all used addresses are observed and bal-

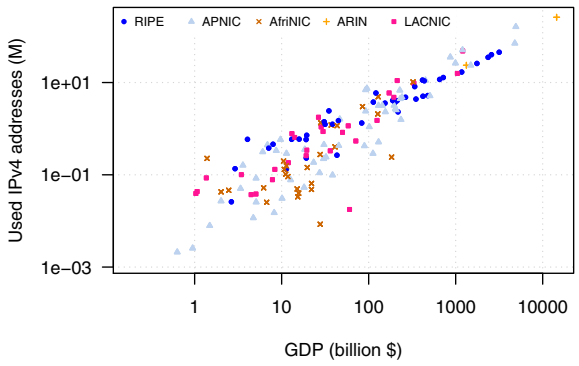

Figure 12: Estimated used IPv4 addresses vs. GDP for each country

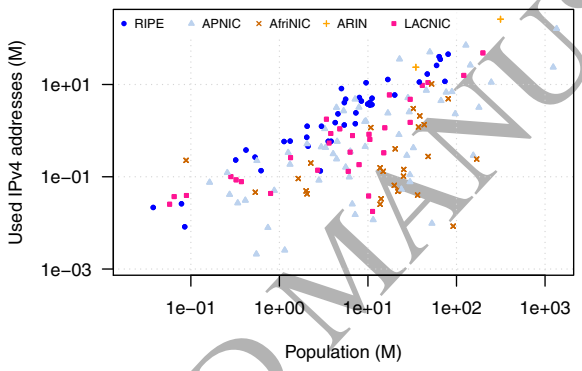

Figure 13: Estimated used IPv4 addresses vs. population for each country

ances performance against precision of the estimated population size. Using a publicly released prototype we show our technique scales well up to 5–10 collaborators and datasets of up to over one billion IPs while the impact on CR estimation accuracy is small (with a sample rate of 1%, the relative error does not exceed 2%). Our technique allowed Netflix to contribute their IP addresses in a privacy-preserving manner.

Another contribution of this paper is an additional CR model. We compare the new model with the log-linear model we introduced previously and show that the estimates of both models are broadly consistent: approximately 1.2 billion IPv4 addresses and 6.5 million /24 subnets were actively used at the end of 2014. The broadly consistent estimates indicate that there is not a high uncertainty in our estimates due to using one specific model. We also present estimates of the number of used IPv4 addresses and /24 subnets for the different regions and different allocation years. Finally, we show that on a per-country basis the number of used addresses is highly correlated with GDP, but seemingly uncorrelated with estimated IPv6 capability.

While our CR estimates may still be lower than the actually used space, there are clear signs of underutilisation at the end of 2014. The IPv4 space is “exhausted”,

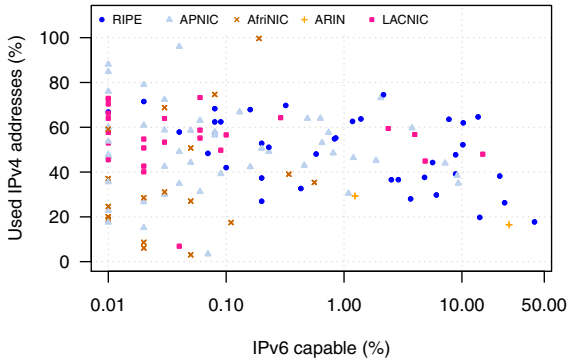

Figure 14: Fraction of allocated used IPv4 addresses vs. percentage of IPv6-capable hosts

because hierarchical allocation, generous allocation sizes given out in early years and minimum allocation sizes to limit the global routing table size cause many addresses to be unused. However, while parts of the unused IPv4 space may be reclaimed, this could only provide short-term relief, given the number of networked devices will rapidly increase due to the Internet of Things (IoT) [2]. A long-term solution exists in the form of IPv6.

Acknowledgements

This research was supported by Australian Research Council grants LP110100240 (co-funded by, and in collaboration with, APNIC Pty Ltd) and FT0991594. We thank G. Huston, G. Michaelson, D. Buttigieg, C. Tassios, A. Reynolds, L. Stewart, Valve Corporation, Swinburne ITS, and Netflix Inc. for providing IP data. We thank the area editor, L. Iannone, and the anonymous reviewers for their insightful comments.

References

[1] G. Huston, IPv4 Address Report, `http://www.potaroo.net/tools/ipv4/index.html`, retrieved 18 Mar 2016.

[2] J. Bradley, J. Barbier, D. Handler, Embracing the Internet of Everything To Capture Your Share of $14.4 Trillion, Cisco White Paper, `http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy.pdf`, retrieved 23 Aug 2016 (2013).

[3] Y. Pryadkin, R. Lindell, J. Bannister, R. Govindan, An Empirical Evaluation of IP Address Space Occupancy, Technical Report ISI-TR 598, USC/ISI (2004).

[4] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, J. Bannister, Census and Survey of the Visible Internet, in: ACM Conference on Internet measurement (IMC), 2008, pp. 169–182.

[5] X. Cai, J. Heidemann, Understanding Block-level Address Usage in the Visible Internet, in: ACM SIGCOMM Conference, 2010, pp. 99–110.

[6] Anonymous, Internet Census 2012 – Port scanning /0 using insecure embedded devices, `http://internetcensus2012.bitbucket.org/paper.html`, retrieved 18 Mar 2016.

[7] A. Dainotti, K. Benson, A. King, kc claffy, M. Kallitsis, E. Glatz, X. Dimitropoulos, Estimating Internet Address Space Usage Through Passive Measurements, ACM Computer Communication Review (CCR) 44 (1) (2014) 42–49.

[8] A. Dainotti, K. Benson, A. King, k. claffy, E. Glatz, X. Dimitropoulos, P. Richter, A. Finamore, A. Snoeren, Lost in Space: Improving Inference of IPv4 Address Space Utilization, Tech. rep., Center for Applied Internet Data Analysis (CAIDA) (Oct 2014).

[9] S. Zander, L. L. H. Andrew, G. Armitage, G. Huston, Estimating IPv4 Address Space Usage with Capture-Recapture, in: 7th IEEE Workshop on Network Measurements (WNM), 2013.

[10] S. Zander, L. L. H. Andrew, G. Armitage, Capturing Ghosts: Predicting the Used IPv4 Space by Inferring Unobserved Addresses, in: Internet Measurement Conference (IMC), 2014.

[11] S. Zander, L. L. H. Andrew, G. Armitage, MAPPING (Measuring And Practically Predicting INternet Growth) Tools, `http://caia.swin.edu.au/mapping/tools.html`, retrieved 18 Mar 2016.

[12] C. G. J. Petersen, The Yearly Immigration of Young Plaice into the Limfjord from the German Sea, Rept. Danish Biol. Sta. 6 (1895) 1–77.

[13] F. C. Lincoln, Calculating Waterfowl Abundance on the Basis of Banding Returns, U.S. Dept . Agric. Circ. 118 (1930) 1–4.

[14] Z. E. Schnabel, The estimation of the total fish population of a lake, Amer. Math. Mon. 45 (1938) 348–352.

[15] R. McCrea, B. Morgan, Analysis of Capture-recapture Data, CRC Press, 2014.

[16] K. H. Pollock, J. D. Nichols, C. Brownie, J. E. Hines, Statistical Inference for Capture-Recapture Experiments, Wildlife Monographs (107) (1990) 3–97.

[17] E. B. Hook, R. R. Regal, Capture-Recapture Methods in Epidemiology: Methods and Limitations, Epidemiol. Rev. 17 (2) (1995) 243–264.

[18] A. Chao, P. K. Tsay, S. H. Lin, W. Y. Shau, D. Y. Chao, The Applications of Capture-Recapture Models to Epidemiological Data, Statistics in Medicine 20 (2001) 3123–3157.

[19] A. Chao, An Overview of Closed Capture-Recapture Models, J. Agric. Biol. Envir. S. 6 (2) (2001) 158–175.

[20] G. C. M. Moura, C. Gãnán, Q. Lone, P. Poursaied, H. Asghari, and M. van Eeten, How Dynamic is the ISPs Address Space? Towards Internet-Wide DHCP Churn Estimation, in: IFIP Networking Conference, 2015.

[21] S. Pledger, Unified maximum likelihood estimates for closed capture-recapture models using mixtures, Biometrics 56 (2) (2000) 434–442.

[22] A. Chao, P. K. Tsay, A Sample Coverage Approach to Multiple-System Estimation with Applications to Census Undercount, Journal of the American Statistical Association 93 (1998) 282–293.

[23] S. Zander, L. L. H. Andrew, G. Armitage, Collaborative and Secure Estimation of IP Address Space Utilisation – Extended Version, Tech. Rep. 150909A, Centre for Advanced Internet Architectures, Swinburne University of Technology, `http://caia.swin.edu.au/reports/150909A/CAIA-TR-150909A.pdf` (September 2015).

[24] B. Pinkas, T. Schneider, M. Zohner, Faster Private Set Intersection Based on OT Extension, in: Proceedings of the 23rd USENIX Conference on Security Symposium, 2014, pp. 797–812.

[25] J. Vaidya, C. Clifton, Secure Set Intersection Cardinality with Application to Association Rule Mining, J. Comput. Secur. 13 (4) (2005) 593–622.

[26] M. Bellare, V. T. Hoang, S. Keelveedhi, P. Rogaway, Efficient Garbling from a Fixed-Key Blockcipher, in: IEEE Symposium on Security and Privacy (SP), 2013, pp. 478–492.

[27] S. Pohlig, M. Hellman, An improved algorithm for computing logarithms over and its cryptographic significance, IEEE Transactions on Information Theory 24 (1) (1978) 106–110.

[28] A. Shamir, R. Rivest, L. Adleman, Mental Poker, MIT/LCS/TM-125, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA (1979).

[29] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, Recommendation for Key Management, Special Publication 800-57 Part 1 Rev. 3, NIST (July 2012).

[30] A. Z. Broder, On the Resemblance and Containment of Documents, in: Compression and Complexity of Sequences 1997, 1997, pp. 21–29.

[31] N. G. Duffield, M. Grossglauser, Trajectory sampling for direct traffic observation, IEEE/ACM Trans. Netw. 9 (3) (2001) 280–292.

[32] R. Morris, K. Thompson, Password Security: A Case History, Bell Laboratories, Murray Hill, NJ, USA, `http://cm.bell-labs.com/cm/cs/who/dmr/passwd.ps` (April 1978).

[33] W. Feller, On the normal approximation to the binomial distribution, The Annals of Mathematical Statistics 16 (4) (1945) 319–329.

[34] D. C. Litzenberger, PyCrypto – The Python Cryptography Toolkit, https://www.dlitz.net/software/pycrypto/, retrieved 17 Jun 2016.

[35] OpenSSL Software Foundation, OpenSSL - Cryptography and SSL/TLS Toolkit, https://www.openssl.org/, retrieved 17 Jun 2016.

[36] A. Appleby, MurmurHash, https://sites.google.com/site/murmurhash/, retrieved 17 Jun 2016.

[37] DNS-based Blacklist of NiX Spam, http://www.dnsbl.manitu.net/, retrieved 18 Mar 2016.

[38] Measurement Lab, http://www.measurementlab.net/, retrieved 18 Mar 2016.

[39] S. Zander, L. L. H. Andrew, G. Armitage, G. Huston, G. Michaelson, Mitigating Sampling Error when Measuring Internet Client IPv6 Capabilities, in: ACM Internet Measurement Conference (IMC), 2012.

[40] S. Baillargeon, L.-P. Rivest, Rcapture: Loglinear Models for Capture-Recapture in R, J. Statistical Software 19 (5) (2007) 1–31.

[41] E. Cooch, G. C. White, Program MARK: A Gentle Introduction, Cornell University, 2009.

[42] APNIC, Minimum prefix and delegations sizes, https://www.apnic.net/manage-ip/manage-resources/address-status/min-prefix, retrieved 23 Aug 2016.

[43] University of Oregon Route Views Project, http://www.routeviews.org/, retrieved 18 Aug 2016.

[44] APNIC, IPv6 Capable Rate by Country, http://stats.labs.apnic.net/ipv6, retrieved 18 Mar 2016.

Sebastian Zander received the Dipl.-Ing. degree from Technical University Berlin, Germany in 1999 and the PhD degree from Swinburne University of Technology, Australia in 2010. He is currently a Lecturer at Murdoch University, Australia. From 1999 to 2004 he worked as researcher and project manager at Fraunhofer FOKUS, Germany and from 2010 to 2015 he was a research fellow at Swinburne University. He co-authored "Information Hiding in Communication Networks" (Wiley, 2016). His research interests include the IPv4 to IPv6 transition, network measurement, traffic classification, covert channels and network security. He is a member of the IEEE and the Australian Computer Society (ACS).

Lachlan Andrew received the B.Sc., B.E. and Ph.D. degrees in 1992, 1993, and 1997, from the University of Melbourne, Australia. He is currently with Monash University, Australia. From 2010 to 2014 he was an ARC Future Fellow. From 2008 to 2013 he was with Swinburne University of Technology, Australia. From 2005 to 2008, he was a senior research engineer at Caltech. Prior to that, he was at the University of Melbourne and RMIT, Australia. He was co-recipient of the best paper award at IGCC2012, IEEE INFOCOM 2011 and IEEE MASS 2007, and the 2014 William Bennett paper award.

Grenville Armitage received the B.Eng. degree (Hons.) in electrical engineering and the Ph.D. degree in electronic engineering from the University of Melbourne, Melbourne, Australia, in 1988 and 1994, respectively. He is currently a Professor of telecommunications engineering, Swinburne University of Technology, Melbourne, Australia. He authored Quality of Service In IP Networks: Foundations for a Multi-Service Internet (Macmillan, 2000) and co-authored Networking and Online Games—Understanding and Engineering Multiplayer Internet Games (Wiley, 2006). Prof. Armitage is a member of the IEEE, the Association for Computing Machinery (ACM) and the ACM Special Interest Group on Data Communication (SIGCOMM).

Sebastian Zander

Lachlan Andrew

Grenville Armitage