Collaborative and Secure Estimation of IP Address Space Utilisation – Extended Version

Sebastian Zander*, Lachlan L. H. Andrew†, Grenville Armitage‡
s.zander@murdoch.edu.au, lachlan.andrew@monash.edu.au, garmitage@swin.edu.au

Centre for Advanced Internet Architectures Technical Report 150909A
Swinburne University of Technology
Melbourne, Australia

***Abstract*—An understanding of IPv4 address space exhaustion, and likely pressure for IPv6 adoption, requires knowing how much of the allocated IPv4 space is *actively used*. A complete view of the whole IP address space is challenging, due to privacy concerns and practical measurement challenges. To address this gap we present a collaborative and secure capture-recapture (CR) technique and tool for estimating the IPv4 address space utilisation. Datasets of IP addresses observed by multiple independent collaborators can be combined for secure CR analysis, without any individual collaborator's 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 1+ 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.**

***Index Terms*—IP space size estimation, Secure set intersection cardinality, Capture-recapture**

I. Introduction

SINCE mid 2015 all Regional Internet Registrars (RIRs), except AfricNIC, have less than one /8 of unallocated addresses remaining [1]. However, understanding the pressures for IPv6 adoption, and the scope of possible IPv4 address markets, requires plausible estimates of actual IPv4 address use – particularly the efficiency with which allocated prefixes are filled with actively-used addresses. We present estimation techniques that can track progressive exhaustion even once all IPv4 prefixes are allocated.

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

In [8], [9] 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. Now we introduce another CR technique that models heterogeneity more directly, and we show that the estimates of both approaches are similar.

CR techniques estimate population sizes from the number of IP addresses in each source and the *sizes of intersections* between all combinations of sources. Many sources of 'used' IP addresses exist, but most cannot be shared for privacy reasons. A key challenge is to efficiently combine data sources of multiple collaborators in a secure manner *without* revealing the observed IP addresses.

We developed an efficient and secure solution called 'Secure Fast Set Intersection' (SeFaSI), by applying Private Set Intersection Cardinality (PSIC) techniques to CR (with a call for collaborators in [10]). We compute the cardinalities of the intersections between data sources (for CR) while *ensuring the anonymity* of the IP addresses observed.

SeFaSI uses computationally-secure commutative encryption [11] (with 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 1+

*School of Engineering and IT, Murdoch University, Australia
†Faculty of IT, Monash University, Australia
‡Centre for Advanced Internet Architectures, Swinburne University of Technology, Australia

billion IPs. Evaluation on small networks for which ground truth is known show that estimates from SeFaSI have a much smaller error than the estimate from simply aggregating all data sources.

Our new key contributions in this paper are:

1) A novel CR technique to estimate the used IP addresses, which deals with heterogeneity of IP addresses directly;
2) A secure protocol (SeFaSI) to feed our CR estimation techniques while keeping the observed IPs private;
3) A publicly available implementation of SeFaSI [12] and our log-linear model CR technique [13],[1] which we use to validate our approach;
4) Results for the estimated number of actively used IPv4 addresses and /24 subnets at the end of 2014.

Our secure 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 data used for matching the data sources, such as names or birth dates.

After a discussion of related work in Section II, Section III explains the basics of CR, the input data needed for CR, log-linear models used in [8], [9] and our new CR model. Sections IV and V describe SeFaSI's design and implementation. Section VI presents our estimation results, and we conclude in Section VII.

II. Related Work

Pryadkin *et al.* [2] 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.* [3] 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 [3] taken in 2007 accounted for 112 million used addresses;results for later censuses are unpublished. 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 the total number of used IPv4 addresses in mid 2007 was 200–210 million.

[1]We also made IPv4 datasets not under NDA publicly available [13].

Cai *et al.* [4] 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 [5]. They detected 420 million addresses that responded to ICMP echo, which is consistent with our ping censuses [9].

In 2013–2014 we proposed using CR to estimate the population of used IPv4 addresses from multiple sources of IPv4 addresses [8], [9]. 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.* [6] 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.* [7] 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 [9]. The difference is likely due to the longer time windows we use.

Our work enhances previous work in that (1) we *estimate the total number of observable addresses* instead of just reporting the 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.

III. Capture-Recapture (CR)

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.

A. 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.

B. 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 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.

C. Assumptions for Capture-Recapture

Basic forms of CR make the following assumptions:

1) Each individual has a unique and consistent identifier across different data sources;
2) The potential population size 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 *if* an address was used and not who used it.

We assume assumption 2 holds, with all type 1 and type 2 addresses being potentially observable by each source. While address usage may be intermittent [4], 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, but that only applies when data sources correspond to measurements at different times. In our study, the data sources measure during concurrent time periods and so temporal changes in the number of addresses do not result in an "open" population.

Assumptions 3 is violated – 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, there is no significant causal relationship to introduce source dependence and so assumption 4 is technically satisfied. 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 [14]. However, assumption 3 and 4 are usually violated *in almost any real scenario*. To this end, a number of CR techniques have been developed that can deal with heterogeneous populations and source dependence.

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 [6], [9].

Finally, CR assumes 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. In [9] we argued that all of our datasets sampled IPs used by routers, servers/proxies, clients, but miss IPs used by specialised devices. Here, we make the further observation that 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 hosts due to dynamic assignment, NAT, or administrative changes.

D. 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 (like in the case of IP addresses). Given the sizes of the data sources and the sizes of all combinations of intersections of datasets one can compute a table of "capture histories".

Let N be the unknown number of distinct IP addresses. Let t denote the number of data sources indexed by $1,2,\ldots,t$. For each IP address, let s_1 to s_t be defined such that $s_i = 1$ if the address occurs in source i and $s_i = 0$ otherwise. Then the string $s_1 s_2 \ldots s_t$ is called 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 a source N_i is often not secret. Then our secure protocol described in Section IV 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 IV-G discusses approaches when some N_i are secret.)

Table I 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 individuals captured by sources 1, 2 and 3, so Z_{111} is computed as the cardinality of the intersection of sources 1, 2 and 3. To compute the counts of individuals in only one source i we need N_i. For example, Z_{001} is the number of individuals only in source 3 and is computed as $Z_{001} = N_3 - Z_{011} - Z_{101} - Z_{111}$.

Table I: 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}

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

The L-P method works only with two sources and if all four assumptions are true. 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" individuals in the second sample equals the fraction of the total population captured in the first sample, then the population N can be estimated by [15], [16]:

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

F. 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], [18], which can model (apparent) source dependence among arbitrarily many sources. For each history s, let $h(s)$ be the set of samples in which the individual 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 $u, u_1, u_2, \ldots, u_{12}, \ldots, u_{23}, \ldots$ 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. Model selection techniques are used to select the least complex model with "adequate" fit (colloquially referred to as the "best model"). During model selection some u_h are forced to 0, to reflect assumed independence between certain combinations of sources [9].

G. 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), which assumes that there are C classes of hosts, and hosts within each class c have approximately equal probability θ_{ct} of being captured in each data source t. For a single class c, the probability of observing a particular $Z_c = (Z_{c,00\ldots0}, \ldots, Z_{c,11\ldots1})$ is

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

where x_{ct} is the total number of observations of class c in source t, and D_c is the total number of distinct observations in class c. 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 [19]. With the approximation $\log(N!/(N-D)!) \approx \int_{N-D}^{N} \log(y)dy$, this becomes

$$\text{M-step} \quad \begin{cases}\theta_{ct} &= x_{ct}/N_c\\ N_c &= D_c/\left(1 - \prod_{t=1}^{T}(1-\theta_{ct})\right)\end{cases}$$

$$\text{E-step} \quad Z_{cs} = D_{\cdot,s}\frac{\Theta_{cs}}{\sum_{c'}\Theta_{c's}}$$

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 $D_c = \sum_{s\neq 00\ldots0} Z_{cs}$.

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 [20], but there appears to be no unbiased estimator for multiple classes.

IV. Secure Protocol

We now describe the goals and requirements for the protocol and the assumed adversary model, followed by an overview of the basic pairwise 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 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.

A. Goals and Requirements

The protocol must ensure that no IP addresses in individual datasets are revealed to collaborators. However, we assume that usually the sizes of datasets can be revealed to collaborators (in Section IV-G we discuss partial solutions to hide the size). It must be possible to run the protocol repeatedly without reducing its security. For example, we may want to run the protocol every few weeks or months 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 get estimates for 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 securely 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, but it is up to the collaborators to decide a priori on an acceptable level of stratification.

The protocol must not rely on trusted third parties, it must work with two or more parties (but the maximum number of parties would be relatively small, say 5–10 parties), and it must work with large datasets of 1+ billion entries. The protocol must be computationally-secure and either must be conceptually simple or a well proven technique to allow for easy verification of its security.

B. 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 are anonymous, security under the honest-but-curious adversary model is sufficient. We assume that man-in-the-middle attacks by third parties are prevented by using secure communication, such as Transport Layer Security (TLS) [21].

C. Basic Protocol

Our basic protocol is based on [11], [22]. First, we introduce 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.

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 $E_{K_i}\left(E_{K_j}(m)\right) = E_{K_j}\left(E_{K_i}(m)\right)$ 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. 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) [23] and commutative RSA (cRSA) [24]. The encryption function is

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

where the modulus p is a large safe prime number[2] shared by all parties and where $\gcd(K_i, p-1) = 1$. It is easy to see that this function is commutative since

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

From [23], [24] we know that this function is computationally-secure if the key and modulus are sufficiently large. Currently, a key size of 224 bits and a modulus size of 2048 bits is deemed to be secure [25]. Depending on the security level needed, smaller sizes may be acceptable, for example a key size of 160 bits and a modulus size of 1024 bits was seen as secure until 2010 [25].

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 (not multisets), but our protocol also enforces this. 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 values (IPs) to the sequence of ciphertexts (encrypted IPs). Let P be a procedure that maps a sequence of values to a permutation of that sequence.[3] Figure 1 shows an overview of the protocol.

In Step 1, A and B negotiate the configuration: the key and modulus sizes, and the modulus value used for

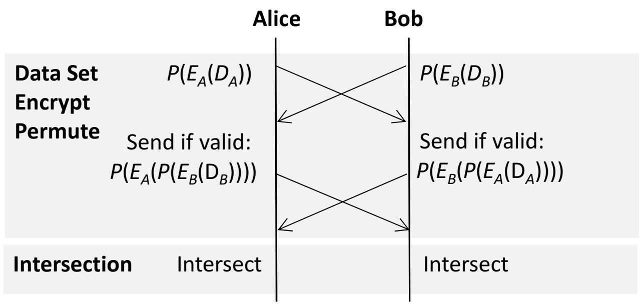


Figure 1: Basic protocol for two parties. Here P and E denote the permutation and encryption of the dataset D created by (A)lice or (B)ob.

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

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

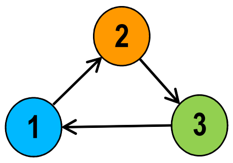


Figure 2: Ring topology for three parties – encrypted and permuted datasets are only passed in one direction

encryption. Then, A and B each generate their own secret encryption key K_i independently.

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 B compute the intersection cardinality by counting the number of equal ciphertexts in both double-encrypted sets. Since the encryption is commutative, any items 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.

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[4] 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 data set.

The parties form a unidirectional ring topology, as shown in Figure 2 for three parties. To counter possible collusion the ring topology could be generated randomly, for example using a secure multi-party computation for random ordering [26]. We assume that each party knows the whole topology and therefore all other parties.

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

[4]To simplify the description we assume one dataset per party, but in practise each party can have multiple datasets.

In Step 3, each party sends its fully-encrypted dataset to all other parties (or a subset of 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.

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 1 billion IPv4 addresses consumes 128 GB (storage and network transfer). The computational overhead is also high, since the exponentiation-based encryption function (Section IV-C1) 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 their disincentive to collaborate is vital, and so reducing the storage, network and computational costs is important.

D. Protocol with Sampling

To make the protocol scalable, we propose that initially each party generates a sampled dataset of smaller size. Simple random sampling cannot sample entries consistently across the different datasets of all parties. We resolve this by using hash-based sampling. The cardinalities of the intersections between the original datasets can be estimated based on the intersection cardinalities of the sampled datasets. We also discuss how to choose the sample rate.

1) Hash-based sampling: The goal of our hash-based sampling [27], [28] 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 [29] (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 $0 < r \leq R \leq R_M$ such that $R_M (\mod R) \ll R_M$. We can then sample the same elements from different lists at rate approximately[5] r/R by selecting only the elements with

$$H(m \oplus s) \,(\mathrm{mod}\, R) < r \,,$$

where $\oplus$ denotes string concatenation.

2) Multi-party protocol with sampling: In Step 1, all parties also need to agree on H, s, and the sample rate $p_D = r/R$. The salt s could be computed in a shared fashion, e.g. each party contributes some bits, to prevent one party from controlling which IPs are sampled.

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

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 $\hat{C} = |\tilde{D}_1 \cap \tilde{D}_2 \cap \ldots \cap \tilde{D}_k| / p_D$ 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 $\sigma^2_{\bar{n}} = Np(1-p)$. Assuming N is large, we can approximate the Binomial distribution with a Normal distribution and omit the continuity correction. Let $z_\alpha = \Phi^{-1}(1-\alpha)$ where Φ is the cumulative distribution of the standard Gaussian (with $\mu = 0$ and $\sigma = 1$). Then

$$\Pr(X \geq \bar{n} - z_\alpha \sigma_{\bar{n}}) = 1 - \alpha \,, \qquad (1)$$

which states that with a probability of $1-\alpha$ the size of the sampled dataset is above $\bar{n} - z_\alpha \sigma_{\bar{n}}$, and

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

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[6]

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

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

3) Minimum sample rate: We can determine the minimum sample rate needed, so that the relative error of the intersection cardinality estimate is not higher than a target maximum relative error $\Delta\epsilon_{\max}$ with a given confidence level $1-\alpha$, if we have a lower bound on the intersection cardinality. Let $\Delta\epsilon_{\max} = |\hat{C} - C|/C$. By (3), with probability $1-\alpha$ we have

$$\Delta\epsilon^2_{\max} \leq \left(\frac{z_{\alpha/2} \cdot \sigma}{Cp_D}\right)^2 = \left(\frac{z_{\alpha/2} \cdot \sqrt{Cp_D(1-p_D)}}{Cp_D}\right)^2 = \frac{z^2_{\alpha/2} \cdot (1-p_D)}{Cp_D} \,.$$

Solving for p_D gives that the relative error will be less than $\Delta\epsilon_{\max}$ with probability $1-\alpha$ if

$$p_D \geq \frac{z^2_{\alpha/2}}{\Delta\epsilon^2_{\max} C + z^2_{\alpha/2}} \,. \qquad (4)$$

In practice we do not know C, but it is sufficient to use a lower bound in (4). If multiple different intersection cardinalities should be computed from more than two datasets, we need to use the minimum of all expected intersection cardinalities to determine the sample rate.

If no lower bound on C is known then a pilot computation with a low sampling rate can be used. From this, a $1-\alpha/2$ one-sided confidence interval for C can be calculated. The lower bound of that interval can be used in a variant of (4) with $z_{\alpha/2}$ replaced by $z_{\alpha/4}$. The probability that the resulting sampling rate gives an error at most $\Delta\epsilon_{\max}$ is $(1-\alpha/2)^2 > 1-\alpha$. This approach is still more efficient than computing the intersection sizes(s) without sampling, if the sum of pilot sample rate and p_D is significantly smaller than 1. However, the choice of sampling rate for the pilot involves a risk tradeoff; a rate too low could result in the confidence interval containing $C = 0$, in which case (4) requires $p_D = 1$ (i.e., no sampling) for the final calculation.

E. Security

The two-party protocol is computationally-secure in the presence of semi-honest adversaries [30] and it is easy to see that the security extends to the multi-party case. The protocol is protected against third parties by using properly configured encryption for all communication, such as TLS [21].

There are no security issues in Step 1. In Step 2 all parties exchange the encrypted permuted IP datasets. Since the encryption is 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 permutations prevent any party from knowing the mapping between its IPs and the final ciphertexts, and thus prevents any party from detecting

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

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

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.

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.

F. 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 $O(kN)$. Since every party has to forward every other party's dataset, the communication complexity is also $O\left(k^2N\right)$ overall and $O(kN)$ for each party.

The sampling of the data only requires one hash computation, one modulo operation and one comparison per IP (neglecting the concatenation with the salt).[7] Fast hash functions only require few integer multiplications per data byte (plus some add, shift and logical operations). For example, the Murmur hash function [31] performs only $3(b+1)$ multiplications where $b = \lceil \text{item size}/4\text{bytes} \rceil$ (a total of six multiplications per IPv4 addresses). The encryption requires one modulo exponentiation per dataset item. Raising a number to an exponent of K bits requires between K and $2K$ multiplications where typically K is a few hundred.[8]

G. Hiding dataset sizes

Previously 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 [11] proposed to hide the size by padding a dataset with 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 III-D). We now briefly describe partial solutions if some parties want to hide their dataset size.

1) Anonymise dataset ownership: This approach requires more than two parties and its effectiveness increases with an increasing number of parties. It is based on layered encryption to anonymise the ownership of datasets similar to onion routing [33]. Every party needs a public/private key pair, where initially the public keys are announced to all other parties. We also assume each party has a unique ID, for example an IP address.

In Step 2, after the initial encryption of a dataset, each party generates a random permutation of the set of $k-1$ IDs of the other parties. This list of $k-1$ IDs plus a final special ID meaning "no forwarding" is the path. Each party encrypts the path in a layered fashion in the reverse order of IDs – from the last ID to the first ID. First, it encrypts the last ID with the public key of the last hop. Then, it encrypts the resulting ciphertext plus the second last ID with the key of the second last hop and so on. The encrypted path is attached to the dataset.

Then Step 2 is carried out. However, each dataset is not forwarded in a circle as described in section IV-C3 but according to the attached encrypted path. At each hop, a party encrypts the dataset as decribed before and also decrypts the outermost encryption layer of the remaining path with its private key to obtain the ID of the next hop. Then, it fowards the dataset together with the remaining encrypted path to the next hop.

The onion routing makes it impossible to link a particular fully-encrypted dataset to a specific party with negligible communication and processing cost. Dataset sizes are not hidden but cannot be attributed to a specific party. An issue arises if the owner of a dataset can be guessed by others purely based on the dataset size. The only solution is to downsample the original datasets of all parties to similar sizes at the cost of some information loss for CR.

2) Dataset merging: If one or more parties are willing to divulge the sizes of their datasets (or even the contents) to another party or parties, then merging multiple datasets prior to running the secure 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

[7]If we restrict $R = 2^e$ with $e = [0, 1, \ldots, N]$ we could replace the slower modulo operation with a much faster AND operation.

[8]Usually Montgomery multiplication [32] is used to implement $a^K \bmod m$, which requires between K and $2K$ Montgomery steps (plus the two transformations to/from Montgomery space that are negligible for large K).

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 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. 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 protool 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.

H. Probing attack detection

The above 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 [11]. 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. The idea behind our scheme is that all parties must provide a dataset of a minimum size and agree on a *valid set* of IPv4 addresses allowing them to check that all fully encrypted datasets consist mainly of valid IP addresses.

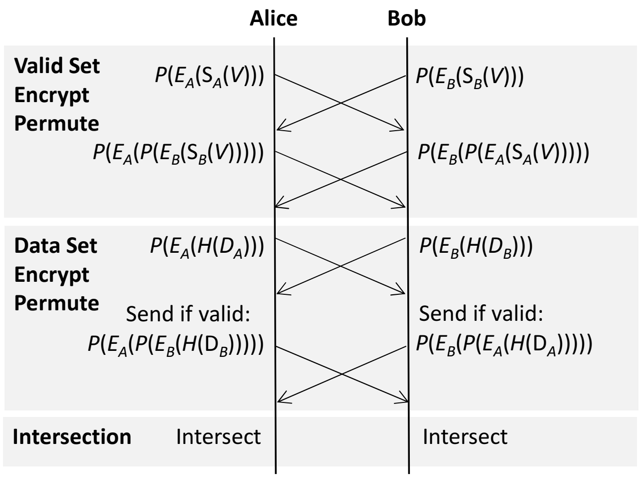


Figure 3: Modified protocol with probing detection for two parties. Here P, E, S and H denote the permutation, encryption, sampling and hash-sampling of the valid set V or the dataset D created by (A)lice or (B)ob. Alice and Bob use private encryption functions and private sampling functions for valid sets, and the same hash-sampling function for datasets.

We introduce our probing detection method for the two-party case here, but extending it to the multi-party case is straight-forward.

1) Modified protocol overview: In Step 1, now all parties also agree on the minimum dataset size $N_{\min}$ and a set of valid items V. In the context of IPv4 addresses the valid set could be the set of routed IPv4 addresses (from advertised routing data). The valid set may be very large. We assume it is usually sampled down to a manageable size with a sampling function S_i, and in Step 1 all parties also agree on the sample rate for the valid set p_V (typically $p_V < p_D$). Figure 3 shows an overview of the modified protocol with probe detection (except the configuration).

A new step "Valid set encryption and permutation" is introduced between Step 1 and Step 2. In this step each party selects an independent and secret random sample of the valid set. These samples can be taken using a good random number generator seeded with *secret* seed values randomly chosen by each party. The seed must be long enough so that the chance of two parties choosing the same value is negligible. Otherwise, an attacker has a chance of guessing another party's secret seed, which may allow the attacker to perform an undetected probing attack by crafting a probe dataset tailored to the sampled valid set of another party.

The parties first perform the same computation as in Step 2 (see Section IV-C2) but with the sampled valid sets. First, all parties encrypt and permute their own valid set. Then all parties encrypt and permute every other party's valid set, using the *same* K_i used later to encrypt the data. In the end each party holds a valid set encrypted by all parties.

Then, in a modified Step 2 a party infers a probing attack if an incoming dataset is smaller than $N_{\min}$ after duplicate removal. Also, after a party has encrypted a dataset for the final time, but before sending it to other parties, the party compares the fully-encrypted dataset against its fully-encrypted valid set. If the set intersection cardinality of an encrypted dataset and an encrypted valid set is below an expected threshold given p_V, a probing attack is inferred.[9] The inferring party deletes the probe dataset and informs all parties of the attack. This prevents the attack, since the prober never get its fully-encrypted dataset. Other parties can still compute the intersection cardinalities.

Note that every time the protocol is used, each party selects a new K_i as part of the configuration (see IV-C2). No party can ever obtain two fully-encrypted datasets of another party encrypted with the same keys. This prevents differential probing attacks, i.e., attacks where a prober participates in the protocol multiple times, once with a valid dataset, the next time with the same dataset plus a small number of IPs to be probed, and so on.

2) New valid set encryption and permutation step: In this step A and B generate the sampled valid set, encrypt and permute it, and send their encrypted and permuted valid sets to each other. A sends $P(E_A(S_A(V)))$ to B, and B sends $P(E_B(S_B(V)))$ to A. A and B then encrypt and permute each other's valid set and return the double-encrypted valid sets to each other. A sends $P(E_A(P(E_B(S_B(V)))))$ to B, and B sends $P(E_B(P(E_A(S_A(V)))))$ to A.

A and B may compute the set intersection cardinality of their valid sets to check if their valid sets are indeed consistent. If the valid set is not sampled the overlap should be 100%. If the valid set is sampled the overlap should be close to $100p_V^2\%$ of the original valid set and within the interval given by (2), assuming hashing is independent because A and B selected different random salts independently. If the valid set overlap is outside the expected range due to a mistake, A and B need to abort and renegotiate the valid set.

3) Modified dataset encryption and permutation step: In the modified Step 2 A and B sample, encrypt and permute their own datasets, then send the encrypted and permuted sets to each other as before. Next, A and B check if each other's filtered dataset is at least of size $N_{\min}$. Given $N_{\min}$ and sampling rate p_D, each party $i \in A, B$ checks if the sampled dataset size n_i exceeds the threshold $\bar{n}_{\min} - z_\alpha \sigma_{\bar{n}_{\min}}$, which by (1) occurs with probability $\geq 1 - \alpha$ if $N_i \geq N_{\min}$. If so, i encrypts and permutes the received dataset. Otherwise, it aborts to prevent a probing attack. However, if A and B agree that a false positive has occurred, the process can be repeated, after A and B have jointly chosen a different dataset sampling salt. The choice of α balances the expected number of times this process must be repeated against the probability of failing to detect a probing attack.

Then each party i computes the set intersection cardinality c_i between its double-encrypted valid set and the other party's double-encrypted dataset. Each dataset item is present in the valid set with probability p_V, the sampling rate used for the valid set. Party i then checks if c_i is at least $p_V n_i - z_\alpha \sqrt{p_V(1-p_V)n_i}$ given the other party's sampled dataset size n_i. If the valid set was not subsampled, $p_V = 1$, this reduces to checking if $c_i = n_i$, which is always true if the data items are all valid. Otherwise, it is true for valid data with probability $1-\alpha$ by (1).[10]

If a dataset is recognised as valid, party i will return it to the other party. Otherwise A and B will abort the protocol, and the probing party cannot learn anything. If A and B suspect a false positive, then the process can be repeated after A and B have chosen different private valid set sampling salts. Again, α is chosen to balance the expected number of times this will occur against the probability of failing to detect a probing attack.

4) Probe attack detection errors: Probing attacks can be misdetected in two ways. False negatives are probe attacks that are not detected, whereas false positives are legitimate datasets misclassified as probe attacks. Our technique may make errors in three cases:

1) when checking the actual size of a dataset against $N_{\min}$,
2) when checking the size of a sampled dataset against $N_{\min}$, and

[9] Since the sampled valid sets are encrypted and permuted and the chosen samples are secret, no party knows which items are in another party's valid set. Hence, an attacker cannot produce a probing dataset that produces a set intersection cardinality over the threshold.

[10] A prober cannot enforce this with many invalid items, because the actual overlap depends on the validating party's *private* validation salt.

3) when checking a dataset against a (sampled) valid set.

Let N_{probe} be the size limit for a practical probing dataset (say no more than a few hundred IPs) and let n_{probe} be the probing dataset sampled with rate p_D. IP address datasets N_i that are useful for CR are typically large (say at least a few millions of addresses), so $N_{\text{probe}} \ll N_i$ for any i. This makes it relatively easy to choose the algorithm parameters such that false negatives *and* false positives are very unlikely and no repeats are needed. We now describe how to do this.

We assume the parties have agreed on a value for N_{probe} ($N_{\text{probe}} \ll N_i$) and the sample rates p_D and p_V. $N_{\min}$ is selected so that $N_{\text{probe}} < N_{\min} \leq N_i$ for any i, so false negatives or false positives are not possible in case 1.

More specifically with $p_D < 1$ all parties agree on an $N_{\min}$ where the upper threshold for a sampled probing dataset $\bar{n}_{\text{probe}} + z_\alpha \sigma_{\bar{n}_{\text{probe}}}$ is smaller than the test threshold $\bar{n}_{min} - z_\alpha \sigma_{\bar{n}_{min}}$ based on $N_{\min}$ (see Section IV-H3) with very high probability (say for $\alpha \leq 10^{-4}$). This means false negatives in case 2 are unlikely. Since each party's dataset must be of size $N_{\min}$ or larger, false positives in case 2 are even more unlikely. For example, with $N_{\text{probe}} = 1000$, $p_D = 10^{-2}$ and $\alpha = 10^{-4}$ we only need $N_{\min} = 5,000$ (in this case the upper threshold for the sampled probing dataset is approaching 22 and the lower threshold for a dataset of size $N_{\min}$ is over 23).

In case 3 the upper threshold for the cardinality of a sampled probing dataset is $p_V N_{\text{probe}} + z_\alpha \sqrt{p_V(1-p_V)N_{\text{probe}}}$, since a prober can ensure that all addresses that should be probed are in the sampled dataset (other addresses in the dataset are invalid), but the prober does not know which addresses are in a sampled valid set of another party. The lower threshold for the cardinality of legitimate datasets is $p_V n_i - z_\alpha \sqrt{p_V(1-p_V)n_i}$ (see Section IV-H3). Given N_{probe}, p_D and p_V we can calculate the upper threshold for a probing dataset with very high probability (say for $\alpha \leq 10^{-4}$). Then we can compute the smallest N_i for which the lower threshold for a legitimate dataset is over this limit with the same low α, so both false negatives and false positives are unlikely. For example, with $N_{\text{probe}} = 1000$, $p_D = 10^{-2}$, $p_V = 10^{-3}$ and $\alpha = 10^{-4}$ the upper threshold for a probing dataset is 1 and we need $N_i \geq 2 \cdot 10^6$ (lower threshold for legitimate data of 3). As shown in Section VI, typical IP datasets are much larger than two million entries.

False positives in cases 2 and 3 can be addressed by repeating the process with different salts, and possibly a higher sampling rate (see Section IV-H3). For the failure probability to decay exponentially with the number of repetitions, failures at successive attempts must be independent. This will occur if H is a perfect (pseudo-)random hash and the salts are chosen without replacement, since the set of arguments to the hashes will then be disjoint between repetitions.

V. Prototype Evaluation

Now we describe our prototype implementation, analyse the accuracy of CR, analyse the impact of dataset sampling and the prototype's performance.

A. Implementation

We implemented a prototype (SeFaSI) based on a mix of tools written in Python and C [12]. The prototype consists of separate tools implementing basic functions (such as sampling, encryption, set intersection cardinality computation) and a "main" tool that implements the secure 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 [13].

SeFaSI implements the whole protocol in Python, since Python source code is small and readable, allowing all parties to 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 and libopenssl's modular exponentiation function. We use the Murmur hash function for sampling as it is fast and has a good distribution that passes the usual tests for hash functions [31].

The results in Section V-C 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.

B. Validation against Ground Truth

As with any real-world CR application, it is not possible to validate the accuracy of our CR approach, since we do not know the ground truth. However, we compared our observed and estimated IP addresses with the ground truth for several networks for which we know the ground truth. (This is an updated version of the analysis in [9].) In most cases[11], the ground truth is an estimate of the number of actively used IPv4 addresses at peak times (high watermarks). We compare this against the observed

[11]For networks B and G we know the number of unique addresses observed in our measurement time window.

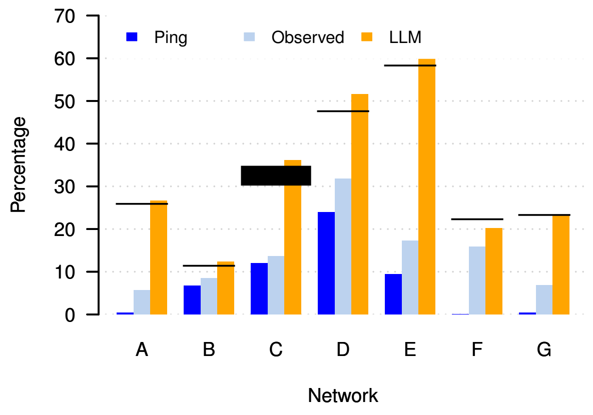


Figure 4: Comparison of LLM estimates, observed and pingable addresses (vertical bars) against the number of reported used IPv4 addresses (horizontal lines)

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 (using the data sources described in Section VI-A). For privacy reasons we cannot reveal the identity of the networks. The largest network is two /16 subnets and the smallest network is roughly one /20 in multiple allocations.

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 truncated Poisson and BIC [9] (vertical bars), and the ground truth (horizontal lines), all as percentages of the sizes of the networks. Note that for network C we only have a range for the ground truth, and network F blocked our pinger, 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 5 of our data sources provide substantial numbers of addresses, which allows CR to work.

C. Sampling Error

First we verify that the sampling error of our prototype is consistent with (3). Then we analyse the impact of the sampling on the CR population estimates.

1) Cardinality estimate sampling error: First we analyse the error caused by sampling and investigate whether the empirical error is consistent with the confidence interval from (3). We used artificially generated datasets of random IPs. Each dataset had 250,000 IPs and we varied the overlap between the datasets. We show the results for two sources (results for three or more sources are very similar).

Figure 5 shows the relative errors of the estimates depending on different true cardinality and different sample rates as boxplots for two sources (100 runs for each setting). The relative error decreases with increasing sample rate and increasing true cardinality. As indicated in Section IV-D2, the error does not depend on the number of sources.

We also analysed the percentage of experiments where the true value lies within the 95% confidence interval (CI) given by (3). If our derived CI is correct, then for a large number of runs the measured percentages should always approach 95%. For the settings in Figure 5 the average fraction of estimates within the CI (plus or minus one standard deviation) is $94.9 \pm 1.8\%$. We conclude our approximate CI is close to the true 95% CI.

2) CR estimate sampling errors: We repeatedly sampled all data sources from [8], each time with a different sample salt, and then computed the LLM CR estimates from the sampled data sources. The estimated range is determined based on the profile likelihood confidence interval [34] with $\alpha = 10^{-7}$ to obtain wide CIs. We investigated the two most commonly used information criteria to select the best model: Akaike's Information Criterion (AIC) and the Bayesian Information Criterion (BIC) [35].

We first investigate the case where the "best model" is selected in each run from the sampled data. This results in larger errors but reflects the realistic case where we never have the unsampled data and hence do not know the best model based on the unsampled data. For comparison we also provide results for the case where the best model was selected from the unsampled data and is then used for all runs with sampled data.

a) LLMs with best model from samples: Figure 6 shows the relative errors for the LLM estimates of the total number of addresses (lower and upper values of LLM estimated ranges) for AIC and BIC model selection depending on different sample rates. We treat the

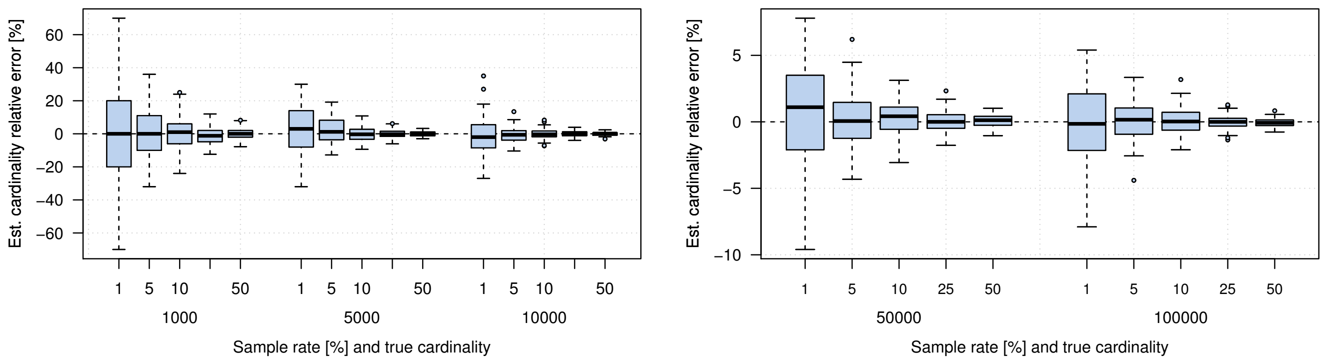


Figure 5: Relative error of cardinality estimates depending on true cardinality size and sample rate (two data sources)

estimates for the unsampled data sources as true values and performed 100 runs with different sample salts per sample rate. The relative errors for LLMs with AIC or BIC are below 5%. Also, with sample rates of 1% or higher the error is usually within 2%. With BIC the errors are generally smaller, and even for a sample rate of 0.1% the error is roughly within 2%. This demonstrates that our sampling technique is very effective.

The larger errors for LLMs are mainly caused by model inconsistencies, since the model was selected independently for each set of sampled datasets. For sample rates of 5% and 10% the selected models are quite consistent. For sample rates of 1% and lower, models for different samples start to diverge.

Compared to the BIC, the AIC selects more parameters *and* also leads to higher variation in the parameter selection. For example, with 1% sample rate the AIC selected 60–67 parameters with 28 parameters occurring in only some of the 100 models, while the BIC selected 52–56 parameters with 11 parameters present in only some of the 100 models. The AIC increases the variance, since it selects more parameters that model higher-level interactions between many sources, for which the intersection cardinalities are smaller and the relative sample error is larger. Also, for higher-level interaction parameters there is a higher chance that sample errors influence the parameter selection, since the inclusion of these parameters changes the model score much less than the inclusion of parameters representing interactions between fewer sources.

We can see some bias in the LLM estimates. The bias is very small for sample rates of 5% or higher, but increases significantly with decreasing sample rate. The bias 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] Again, the effect is dominated by parameters that represent interactions between fewer sources. Both AIC and BIC tend to include several of these parameters and show a positive bias.

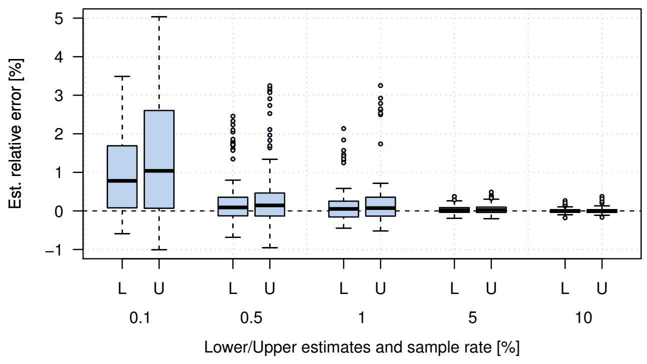
b) LLMs with best model from unsampled data: To demonstrate that the bias in the LLM estimates is caused solely by the model selection process, we conducted another experiment with the AIC where the model was fixed – in each run the used model was the model selected by AIC for the unsampled data. Figure 7 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 sample rates, treating the estimates for the unsampled data sources as true values (100 runs with different sample salts per sample rate). It shows that with a fixed model the LLM estimate is unbiased and has substantially smaller error.

D. 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 in all performance measurements we used a single CPU core only.

1) Sampling speed: Table II shows the number of IPv4 addresses processed per second when sampling depending on the sample rate measured for the C implementation (average of 10 runs). File-write overhead increases with higher sample rates. Even with 50% sample rate our

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



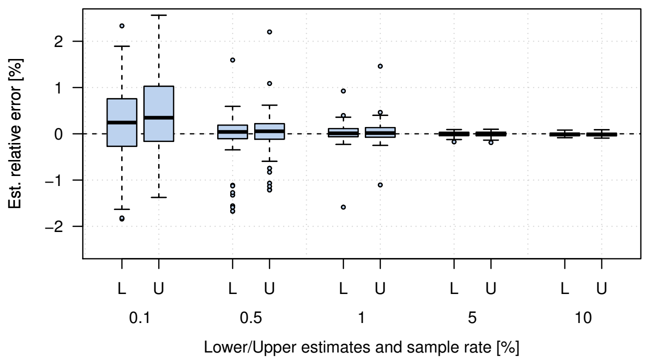


Figure 6: Relative error of (L)ower and (U)pper values of CR estimated ranges depending on the sample rate for LLMs using the AIC (left) and BIC (right)

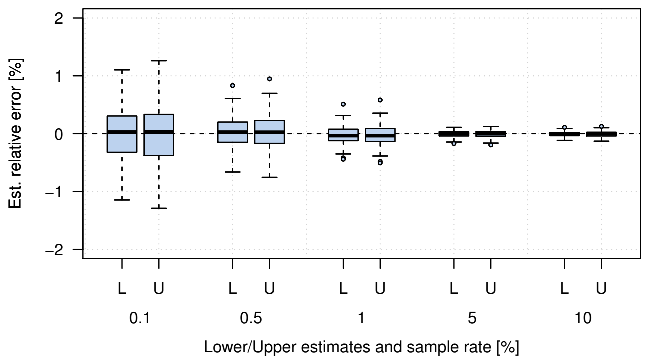


Figure 7: Relative error of (L)ower and (U)pper values of CR estimated ranges depending on the sample rate for LLMs with AIC, using the AIC model selected for the unsampled data

Table II: Speed of sampling IPv4 addresses depending on sample rate

Sample rate [%]	IPv4s/second
1	5,450,000
5	5,300,000
10	5,200,000
50	4,350,000

prototype can sample about 4,35 million IPs per second, so it would take only 4 minutes to sample a dataset with 1 billion IPv4 addresses.

2) Encryption and permutation speed: Table III shows our C implementation's encryption and permutation speed versus key and modulus sizes for both IPv4 addresses and already encrypted IPv4 addresses (average over 10 runs). As expected, the rate decreases with increasing key and modulus sizes. While it takes longer to encrypt

Table III: Speed of encrypting and permutating both raw IPv4 addresses (4 bytes) and encrypted IPv4 addresses (size=modulus) depending on key and modulus sizes.

Key/Modulus size [bits]	Raw IPv4s [/second]	Encrypted IPv4s [/second]
128/128	29,000	28,000
160/256	19,000	18,500
160/512	10,000	10,000
160/1024[a]	3,600	3,500
224/2048[b]	1,100	1,000

[a]NIST 2010 (Legacy) [25]
[b]NIST secure until 2030 [25]

already-encrypted IPs (long input data), the difference is relatively small (which we attribute to libopenssl's modular exponentiation being the bottleneck[13]). Note that the rates are dominated by the encryption, which takes 97–98% of the time. Only 2–3% of the time is used by the permutation.

With an insecure 64 bit key and 64 bit modulus it would take 9–10 hours to encrypt and permute 1 billion IPv4 addresses. With 160 bit key and 1024 bit modulus (NIST 2010 Legacy [25]) it would take a little over 3 days to encrypt and permute 1 billion IPv4 addresses. However, with a sample rate of 10% the time reduces to only 8 hours (including the sampling).

3) Set intersection cardinality speed: Table IV shows the set intersection cardinality speed depending on the number of *encrypted* IPv4 datasets and the overlap (aver-

[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.

Table IV: Speed of set intersection cardinality computation depending on key size and modulus size with encrypted IPv4 addresses

Datasets	Overlap [%]	IPv4s/second
2	100	2,180,000
3	100	1,500,000
4	100	1,150,000
5	100	950,000
2	0	1,650,000
3	0	880,000
4	0	540,000
5	0	360,000

age of 10 runs). All datasets had the same size. 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 roughly 1 billion IPv4 addresses each would take between 18 minutes and 50 minutes depending on the overlap. However, with a sample rate of 10% the time would reduce to 2–5 minutes.

VI. Estimated Used IPv4 Space

Now we present results for estimating the used IPv4 space.

A. Datasets

Table V summarises the datasets we collected from multiple sources of unique IPv4 addresses between July 2013 and December 2014. 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[14] (WIKI), potential spam email senders from [36] (SPAM), addresses of clients tested by Measurement Lab [37] tools (MLAB), web clients participating in our IPv6 readiness test [38] (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)[15]. These datasets and the pre-processing are described in more detail in [9]. We have now added a significant new dataset based on obscured IP addresses of clients connected to Netflix (NFLIX).

[14]Modification time and IPv4 address of edits by unregistered users.

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

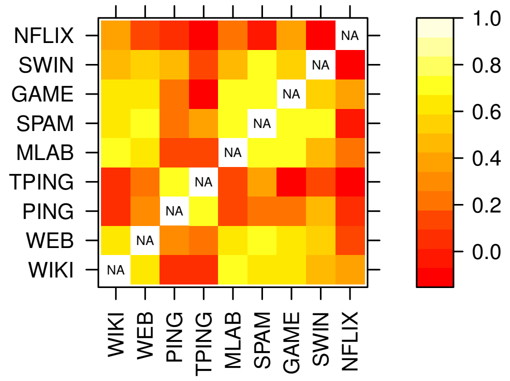


Figure 8: Estimated correlations between data sources

The only dataset that can include spoofed IPs is SWIN, and we used the heuristic from [9] 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 V 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.

B. Correlations Between Datasets

Figure 8 shows the estimated correlations between our data sources. These are the Yule coefficients computed based on 2x2 capture frequency tables for each source combination with Z_{00} estimated by CR. Most of the correlations are positive. TPING and IPING, which include servers, are strongly correlated with each other, but 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.

C. Used IPv4 space

We now present our estimates for the IPv4 space. Unless otherwise noted results are for LLMs. For LLMs we are using the adaptive divisor approach and the BIC for the model fitting [9]. 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.

Table V: 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 [37]	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 [38]	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

1) Totals: Table VI 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; it is unclear why.

A potential benefit of LCMs over LLMs is that meaningful classes of users can sometimes be identified. One 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.[16] The standard deviation is roughly 100 million addresses, but if we exclude the case of leaving out IPING it reduces to only 50 million. For /24 subnets the n estimates are very consistent, and the standard deviation is only 65,000 subnets.

2) Usage by region: Figures 9 and 10 show the absolute number and percentage of routed space of the pingable, observed, estimated used IPv4 addresses and /24 subnets for the five RIRs at the end of 2014. Note, that *the top of each bar segment* indicates the number of pingable, observed and estimated used IPv4 addresses or /24 subnets (to reduce the number of bars). The results

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

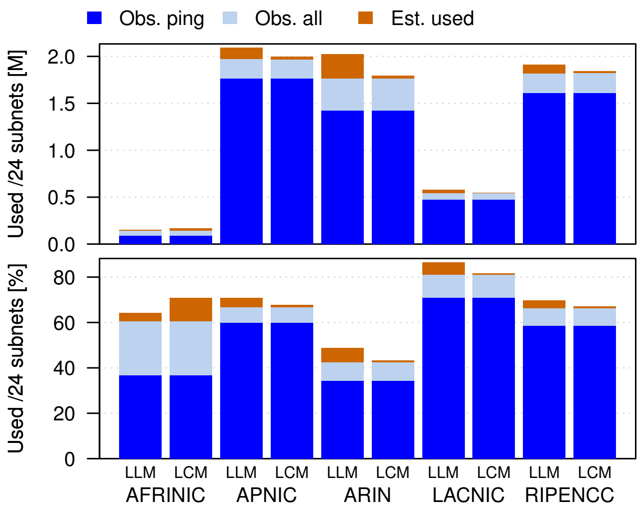


Figure 9: Pingable, observed and estimated /24 subnets at the end of 2014 for the different RIRs

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. For both IPv4 addresses and /24 subnets, APNIC, ARIN and RIPE have the highest numbers of used addresses/subnets, whereas LACNIC has the highest utilization.

3) Usage by country: Figure 11 plots the estimated used IPv4 addresses for each country against the number of allocated IP addresses (as reference), the country's GDP, the country's population size and the number of days since the first address allocation on a log-log scale for the different regions (RIRs). Results for /24 are broadly similar. The correlation between the number of estimated used addresses and GDP is clearly stronger than the correlation between the estimated used addresses

Table VI: 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]	Est. 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	0.1–0.5	10.8
			RIR	6.8	6.4		

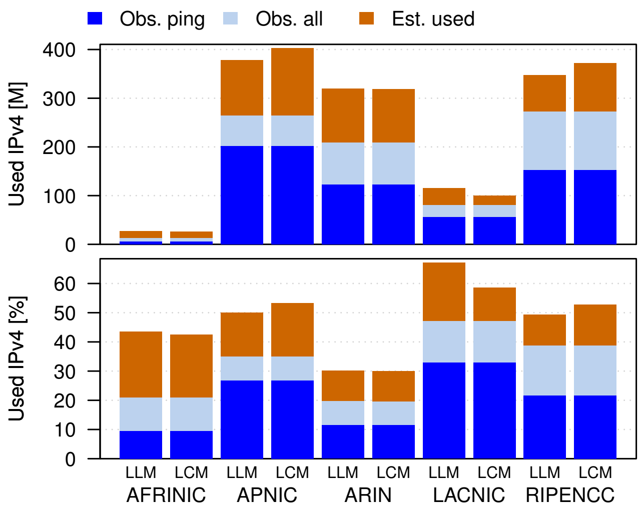


Figure 10: Pingable, observed and estimated IPv4 addresses at the end of 2014 for the different RIRs

and population or days since online. Especially for the African, Asian and South American regions we see countries with larger populations that only use a comparatively small part of the IPv4 space. The result is consistent with a strategy of allocating IPv4 addresses proportional to the need for Internet access, which is correlated more with GDP than with population.

4) Impact on IPv6 Deployment: Figure 12 plots the fraction of allocated used IPv4 addresses for each country against the percentage of hosts that are IPv6-capable or prefer to use IPv6 according to July 2015 APNIC data [39] (the x-axis is logarithmic). Results for /24 subnets are similar. Surprisingly, there is no correlation between the fraction of used IPv4 addresses (or /24 subnets) and IPv6 readiness.

VII. Conclusions and Future Work

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.

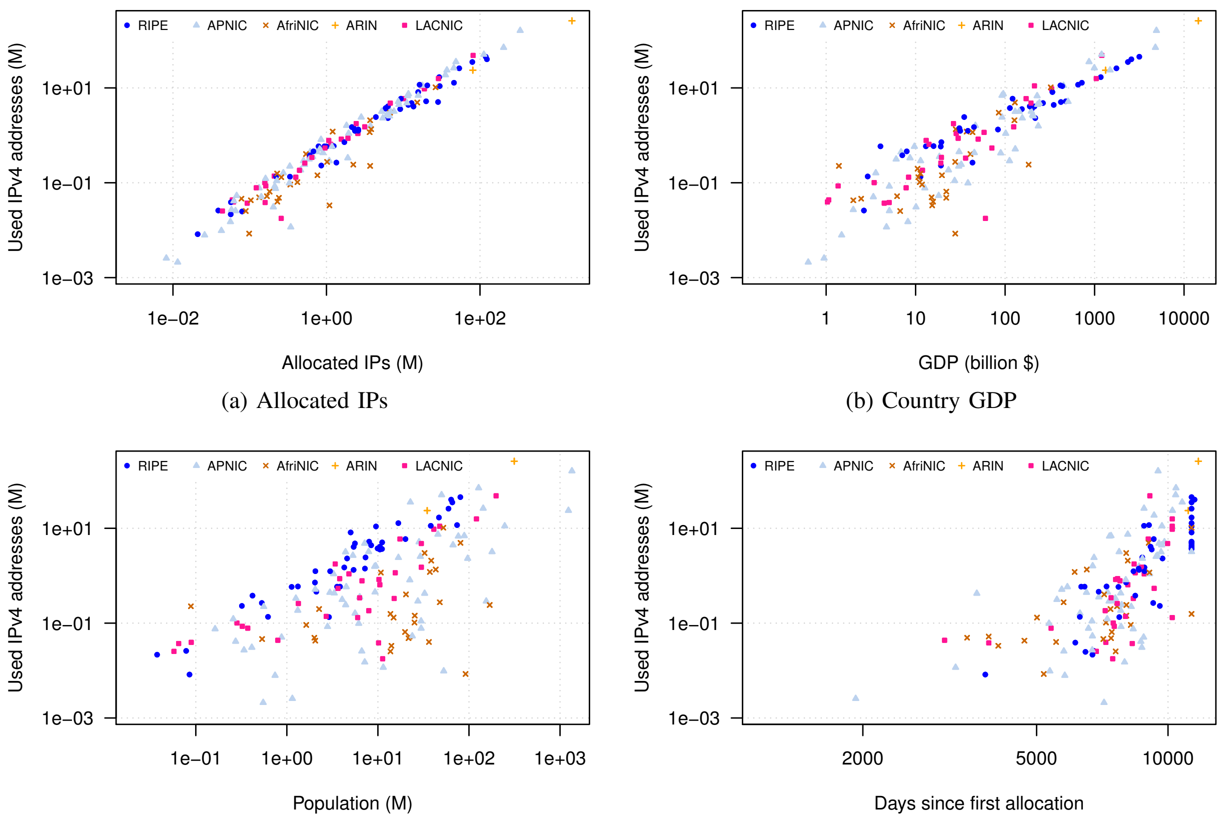
We presented a new collaborative and secure capture-recapture (CR) technique for estimating address space utilisation from multiple sources of observed IP addresses while guaranteeing the privacy of the addresses. Our technique is much more accurate than assuming all used addresses are observed and balances performance against precision of the estimated population size. Using a publicly available prototype we show our technique scales well up to 5–10 collaborators and datasets of up to 1+ billion items while the impact on CR estimation accuracy is small (with a sample rate of 1%, the relative error does not exceed 2%). Our secure technique allowed Netflix to contribute their IP addresses in a privacy-preserving manner.

Another contribution of this paper is an additional CR model, which we compare with the log-linear models we used previously. We showed 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. We also presented updated estimates of the number of used IPv4 addresses and /24 subnets for the different regions. Finally, we showed that on a per-country basis the number of used addresses is highly correlated with GDP, but seemingly uncorrelated with estimated IPv6 capability.

In the future we aim to further refine our CR model and to add more privacy-sensitive collaborators.

Acknowledgements

This research was supported by Australian Research Council grants LP110100240 (with APNIC Pty Ltd) and FT0991594. We thank G. Huston, G. Michaelson, Valve Corporation, A. Reynolds, Swinburne ITS, Caltech IMSS, Netflix, D. Buttigieg, C. Tassios, R. Bevier, B. Mattern, L. Stewart for providing data.



(a) Allocated IPs

(b) Country GDP

(c) Country population size

(d) Days since first allocation of country

Figure 11: Estimated used IPv4 addresses for each country vs. allocated /24 subnets, GDP, population and days since first allocation

References

[1] G. Huston, "IPv4 Address Report." http://www.potaroo.net/tools/ipv4/index.html.

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

[3] 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)*, pp. 169–182, 2008.

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

[5] Anonymous, "Internet Census 2012 – Port scanning /0 using insecure embedded devices." http://internetcensus2012.bitbucket.org/paper.html.

[6] 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)*, vol. 44, pp. 42–49, Jan. 2014.

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

[8] 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)*, October 2013.

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

[10] S. Zander, L. L. H. Andrew, and G. Armitage, "Estimating the used IPv4 address space with secure multi-party capture-recapture," in *INFOCOM (poster)*, (Turin, Italy), 15-18 Apr 2013.

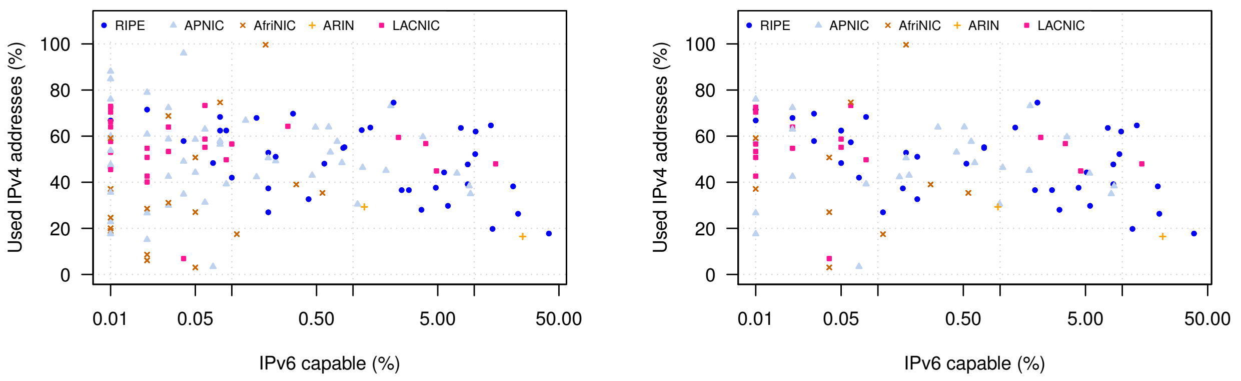
[11] J. Vaidya, C. Clifton, "Secure Set Intersection Cardinality with Application to Association Rule Mining," *J. Comput. Secur.*, vol. 13, pp. 593–622, July 2005.

[12] S. Zander, "Secure Fast Set Intersection (SeFaSI) Implementation," 2013. http://caia.swin.edu.au/mapping/sefasi.

[13] S. Zander, L. L. H. Andrew, G. Armitage, "MAPPING (Measuring And Practically Predicting INternet Growth) Tools," 2015. http://caia.swin.edu.au/mapping/tools.html.

[14] 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*, vol. 20, pp. 3123–3157, October 2001.

[15] C. G. J. Petersen, "The Yearly Immigration of Young Plaice into the Limfjord from the German Sea," *Rept. Danish Biol. Sta.*, vol. 6, pp. 1–77, 1895.



(a) IPv6 capable (b) IPv6 preferred

Figure 12: Fraction of used IPv4 addresses relative to routed space vs. percentage of IPv6-capable and IPv6-preferred hosts

[16] F. C. Lincoln, "Calculating Waterfowl Abundance on the Basis of Banding Returns," *U.S. Dept . Agric. Circ.*, vol. 118, pp. 1–4, 1930.

[17] E. B. Hook, R. R. Regal, "Capture-Recapture Methods in Epidemiology: Methods and Limitations," *Epidemiol. Rev.*, vol. 17, no. 2, pp. 243–264, 1995.

[18] A. Chao, "An Overview of Closed Capture-Recapture Models," *J. Agric. Biol. Envir. S.*, vol. 6, no. 2, pp. 158–175, 2001.

[19] S. Pledger, "Unified maximum likelihood estimates for closed capture-recapture models using mixtures," *Biometrics*, vol. 56, no. 2, pp. 434–442, 2000.

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

[21] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246 (Proposed Standard), August 2008. http://www.ietf.org/rfc/rfc5246.txt.

[22] R. Agrawal, A. Evfimievski, R. Srikant, "Information Sharing Across Private Databases," in *ACM SIGMOD International Conference on Management of Data*, pp. 86–97, 2003.

[23] S. Pohlig, M. Hellman, "An improved algorithm for computing logarithms over and its cryptographic significance," *IEEE Transactions on Information Theory*, vol. 24, pp. 106–110, January 1978.

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

[25] 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.

[26] L. Sweeney, M. Shamos, "A Multiparty Computation for Randomly Ordering Players and Making Random Selections," Tech. Rep. CMU-ISRI-04-126, Carnegie Mellon University, July 2004.

[27] A. Z. Broder, "On the Resemblance and Containment of Documents," in *Compression and Complexity of Sequences 1997*, pp. 21–29, June 1997.

[28] N. G. Duffield, M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 280–292, June 2001.

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

[30] E. De Cristofaro, P. Gasti, G. Tsudik, "Fast and Private Computation of Cardinality of Set Intersection and Union," in *11th International Conference on Cryptology and Network Security (CANS)*, 2012.

[31] A. Appleby, "MurmurHash," 2011. https://sites.google.com/site/murmurhash/.

[32] P. Montgomery, "Modular Multiplication Without Trial Division," *Math. Computation*, vol. 44, pp. 519–521, 1985.

[33] G. D. M. Reed M. G., Sylverson P. F., "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.

[34] S. Baillargeon, L.-P. Rivest, "Rcapture: Loglinear Models for Capture-Recapture in R," *J. Statistical Software*, vol. 19, pp. 1–31, April 2007.

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

[36] DNS-based Blacklist of NiX Spam. http://www.dnsbl.manitu.net/.

[37] Measurement Lab. http://www.measurementlab.net/.

[38] 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)*, Nov. 2012.

[39] APNIC, "IPv6 Capable Rate by Country." http://stats.labs.apnic.net/ipv6.