Entropy/IP: Uncovering Structure in IPv6 Addresses

Paweł Foremski
Akamai Technologies
IITiS PAN
pjf@iitis.pl

David Plonka
Akamai Technologies
plonka@akamai.com

Arthur Berger
Akamai Technologies
MIT CSAIL
arthur@akamai.com

ABSTRACT

In this paper, we introduce *Entropy/IP*: a system that discovers Internet address structure based on analyses of a subset of IPv6 addresses known to be active, *i.e.,* training data, gleaned by readily available passive and active means. The system is completely automated and employs a combination of information-theoretic and machine learning techniques to probabilistically model IPv6 addresses. We present results showing that our system is effective in exposing structural characteristics of portions of the active IPv6 Internet address space, populated by clients, services, and routers.

In addition to visualizing the address structure for exploration, the system uses its models to generate candidate addresses for scanning. For each of 15 evaluated datasets, we train on 1K addresses and generate 1M candidates for scanning. We achieve some success in 14 datasets, finding up to 40% of the generated addresses to be active. In 11 of these datasets, we find active *network identifiers* (*e.g.,* /64 prefixes or "subnets") not seen in training. Thus, we provide the first evidence that it is practical to discover subnets and hosts by scanning probabilistically selected areas of the IPv6 address space not known to contain active hosts *a priori.*

1. INTRODUCTION

Understanding the structure of Internet addresses has become increasingly complicated with the introduction, evolution, and operation of Internet Protocol version 6 (IPv6). Complications arise both from *(a)* IPv6's address assignment features, *e.g.,* stateless address autoconfiguration (SLAAC), in which clients choose their own addresses, and from *(b)* the freedom allowed by IPv6's vast address space and enormous prefix allocations from address registries, *e.g.,* 2^{96} addresses (by default) to each Internet Service Provider (ISP).

By empirical observation today, we see many address complications in the large-scale operation of IPv6. As of August 2016, estimates suggest that 11% of World-Wide Web (WWW) users have IPv6 capability and use it to access popular sites [16]. Yet, at this modest level, measurements show *billions* of active IPv6 WWW client addresses being used monthly, and tens to hundreds of millions of IPv6 router addresses. Addresses often differ in the spatial and temporal characteristics from one operator or network to the next [27]. Complications that we observed include (but are in no way limited to): addresses with Modified EUI-64 interface identifiers that are curiously *not* tagged as globally unique, *stable* addresses containing pseudo-random numbers in their interface identifiers,[1] and even addresses containing pseudo-random numbers in their *network identifiers.* These are the sort of challenges we face with IPv6.

In this work, we study sets of IPv6 addresses and the structural characteristics embedded within them. We are primarily motivated by the challenges of widespread native IPv6 operation (in parallel with IPv4), as it exists today. First, WWW services often wish to deliver content differentially, *e.g.,* based on a client's geographic location or its host reputation; understanding IPv6 structure would help determine how these IPv4 characteristics might, likewise, apply to clients' IPv6 addresses. Second, security analysts wish to be able to assess IPv6 networks' vulnerability to host scanning; since full IPv6 address-space scans are infeasible, new methods must take its structure into account. Third, network operators, as well as engineers and researchers, wish to track IPv6 deployment and growth to prioritize their work; a systematic approach to deep understanding of IPv6 address structure aids measurement interpretations that clearly differ from IPv4 to IPv6, *e.g.,* so that we properly recognize the significance of estimated subnet sizes and measured active host and prefix counts.

The goal of our system, "Entropy/IP," is to provide a means by which one might remotely glean and un-

IMC 2016, November 14 - 16, 2016, Santa Monica, CA, USA

ISBN 978-1-4503-4526-2/16/11…$15.00

DOI: http://dx.doi.org/10.1145/2987443.2987445

[1] The authors are aware of proposed addressing schemes involving both randomized MAC addresses and stable privacy addresses. These privacy mechanisms increase complication.

derstand a network's addressing plan. Ultimately, we would like to discover Classless Inter-Domain Routing (CIDR) prefixes, Interior Gateway Protocol (IGP) subnets, network identifiers, and interface identifiers (IID) of each address. We employ three core techniques:

- **Entropy Analysis:** A key aspect of our system is that it leverages information-theoretic entropy [30]. We employ it to measure the variability of values at each of 32 positions of *nybbles* (*i.e.,* hexadecimal characters) in IPv6 addresses. We then compare the entropy of adjacent nybbles to detect significant differences, with the expectation that these represent boundaries between semantically distinct parts of each address. When there are no significant differences, we group adjacent nybbles together to form larger *segments.* For instance, subnet identifiers or stateless address auto-configuration (SLAAC) attributes might be formed of adjacent nybbles that exhibit similar entropy.

We chose to compute the entropy of nybbles at a certain position across a set of addresses as an aid to understand the structure of these addresses. Here, we are guided by our experience that "stateless" analysis of individual address content is clearly error-prone, and "stateful" analysis (spatial or temporal) has benefits as Plonka and Berger [27] show. For instance, the reasonable, but stateless, rules to detect pseudo-random IIDs implemented in the `addr6` tool [31], misclassify `2001:db8:221:ffff:ffff:ffff:ffc0:122a` as having a randomized IID even when it is accompanied by one thousand other similarly constructed addresses in the `2001:db8:221:ffff:ffff:ffff:ff::/104` prefix. Context is important for accurate understanding of address content and entropy is an ideal metric to recognize portions of addresses that contain information that might be pertinent to address structure. For instance, oft repeated bits—*e.g.,* those `f` characters—may contain hints about IID assignment practice, but contain little *information* (in an information-theoretic sense) and are, thus, unlikely to help in reverse-engineering netmasks and subnet structure.

- **Clustering:** The second key aspect of our system is that we employ unsupervised machine learning for clustering of the segments' values based on their distribution and the frequencies of occurrence of those values. We use the popular DBSCAN clustering algorithm [10].
- **Statistical Modeling:** The third key aspect of our system is the use of Bayesian Networks (BNs) to statistically model IPv6 addresses. In short, we automatically determine conditional probabilities amongst clusters of segments' values in a hierarchical fashion, *i.e.,* directed left to right, across the address segments. This aspect was motivated by painstaking experiences in visual examination of large sets of IPv6 addresses. It seems promising that a machine learning technique could unveil address structure similar to how a researcher sometimes can. Unsupervised learning seems especially appropriate since prior work found numerous instances of structure that does not follow RFC-defined address assignment policies [27]. Thus, we do not train our system to recognize well-known features—such as `ff:fe` in Modified EUI-64, or ostensibly pseudo-random numbers in privacy addresses—but rather rely on our system's entropic underpinnings to *discover* these unique characteristics for itself.

In stepwise fashion, Entropy/IP ingests a sample set of IP addresses, computes entropies, discovers and mines segments, builds a BN model, and prepares a graphical web page with the following elements for a network analyst to navigate and explore the exposed structure:

- a plot of entropy and aggregate count ratio,
- a BN, showing address segments inter-dependencies,
- a segment value browser with frequency heat map,
- a target address generator.

Entropy/IP's user interface is shown in Figures 1 and 2. A live demo, which allows for public operation, is available at http://entropy-ip.com/.

There are numerous applications of structural analysis of active IP addresses. These include *(a)* identifying homogeneous groups of client addresses, *e.g.,* to assist in IP geolocation or in the mapping of clients to content hosted on Content Distribution Networks (CDNs), *(b)* supporting network situational awareness efforts, *e.g.,* in cyber defense or in competitive analysis, and *(c)* selecting candidate targets for active measurements, *e.g.,* traceroutes campaigns, vulnerability assessments, or reachability surveys. With respect to survey, temporary addresses complicate the estimation of users or service subscribers by counting IPv6 addresses or prefixes at any one length (*e.g.,* 64 bits), so understanding addressing in network structure is critical in interpreting IPv6 address or prefix counts. If we could interpret these counts, they could be used to inform, *e.g.,* policy, standards, and capacity-planning decisions. Yet another application of structure analysis is *(d)* remotely assessing networks' addressing plan and address assignment policy. This is valuable for host reputation and access control, *i.e.,* when mitigating abuse originating from sources within that network. Such external assessments are also valuable to the subject networks *themselves, e.g.,* to assess potential security or privacy risks [3]. For instance, one network operator, whom we contacted to comment on our results, asked *us* whether or not their customers' addresses could be predicted, *i.e.,* whether or not their address assignments appear to support user privacy as intended.

This work makes the following contributions:

(1) We present an automated system that discovers aspects of networks' IPv6 address layout based on observations of a subset of that network's active addresses. Our system employs an entropy-based technique in combination with standard machine learning and statistical modeling techniques to discover structural characteristics in arbitrary IPv6 address sets.

(2) We improve upon prior works on address classification by employing a measure of *entropy* to identify address sets that have very high entropy values across

multiple adjacent nybbles—which likely reveal pseudo-random segments—and middle-to-high range values, as well as abrupt changes in entropy between segments—which likely reveal addressing structure. This improves identification of privacy addresses.
(3) We present results and an evaluation of our system that demonstrate how it enables analysts to interactively explore the structural characteristics of arbitrary sets of addresses and, if desired, generate candidate target addresses for active scanning more broadly than existing methods described in the literature.

The remainder of this paper is organized as follows. In Section 2, we discuss related works. In Section 3, we describe the data used in our empirical study. In Section 4, we present our system, the methods by which it is implemented, and their component techniques. In Section 5, we present results of our evaluation. In Section 6, we discuss limitations of our method and future work. Subsequently, we conclude this paper in Section 7.

2. RELATED WORK

To the best of our knowledge, the components in our method have not previously been applied to the problem of uncovering IP address structure. However, statistics, entropy, and machine learning have been applied to network traffic analysis in numerous works involving IP addresses and other traffic features. For instance, Lee and Xiang [24] develop models for network anomaly detection based on entropy across features of traffic records, including host identifiers, though they do not specifically mention IP addresses. Feinstein *et al.* [12] develop a detector that relies on their observation that attacks from distributed sources result in increased entropy of packet header features; they focus on source IP address in their evaluation. Subsequently, both Lakhina *et al.* [23] and Wagner *et al.* [34] likewise compute entropy across IP header features. These works differ from ours in that they treat individual addresses as atomic, *i.e.,* semantically opaque, and largely use entropy as a measure of feature distribution, *e.g.,* address set inflation, and typically detect when it varies in time-series.

One work that deals specifically with entropy and IPv6 addresses is that of Strayer *et al.* [32] Circa 2004, they note that the IPv6 packet header, including the source and destination addresses, exhibits less entropy (per byte) than that of IPv4. We believe this is precisely why our method is effective. In IPv6 addresses, information relevant to structure or forwarding need not be "compressed" into only a 4-byte identifier, as it is in IPv4 addresses. Instead, information can (and often is) spread across an IPv6 address, *i.e.,* all of 32 nybbles.

Since then, the introduction and popularization of privacy extensions [26] for IPv6 addresses changed the situation by adding pseudo-random values, and thus high entropy, to IPv6 addresses. While random values have high information content in an information-theoretic sense, the information therein is not *pertinent* to network structure. Thus, it is useful to identify these so-called "privacy addresses" and either disregard their random segments (IIDs) or treat them specially. Works by Malone [25], Gont and Chown [14] (as implemented in the `addr6` tool [31]), and Plonka and Berger [27] each attempt to identify pseudo-random numbers in IPv6 address segments. We do so as well in this work, differing in that we leverage entropy.

Kohler *et al.* [19] employ a hierarchical approach within IPv4 addresses and count aggregates (prefixes) at each possible length, 0 through 32. Plonka and Berger [27] build on that approach and introduce Multi-Resolution Aggregate (MRA) Count Ratios for IPv6 addresses and use them to discover structure in addresses. They were also inspired by prior works [2, 9] that summarize the IP address space into an abbreviated structure, albeit entirely synthetic. In contrast, in this work we take a looser hierarchical approach: we separately consider each nybble position and we group them into segments of varying lengths. Thus, Entropy/IP provides a complimentary viewpoint into IPv6 addresses, independent from MRA analysis. However, for the interest of readers who are familiar with MRA plots, we show 4-bit Aggregate Count Ratios (ACR) in some of our figures (normalized to a range of $0 - 1$). At a high level, ACR reveals how much a segment of the address is relevant to grouping addresses into areas of the address space. The higher the ACR value, the more pertinent to prefix discrimination a given segment is. To understand the contribution of this paper, one can ignore the ACR metric, though.

Krishnamurthy and Wang's work [22], circa 2000, is similar to ours in its premise, *i.e.,* automated grouping of homogeneous active addresses, where only the BGP prefixes are known in advance. It differs significantly in its methods and focuses only on IPv4.

Nascent work, which at first glance is most similar to ours, is that of Ullrich *et al.* [33], who develop a pattern-discovery-based scanning approach to IPv6 reconnaissance. They algorithmically detect recurring bit patterns (*i.e.,* structure) in the IID portion of training subsets of known router, server, and client addresses, and then generate candidate targets according to those patterns. They report improved performance versus the surveillance approach outlined by Gont and Chown [14] that relies on a mixture of patterns known or observed *a priori*, as implemented in the `scan6` [31] tool. Our work differs most significantly from these prior works in that we discover patterns across *whole* IPv6 addresses, including the network identifier portion, whereas theirs focus only on the bottom 64 bits (*i.e.,* ostensibly the IID). Thus, they assume a surveyor or adversary knows which /64 prefixes to target. Since our method can also be used to generate target /64 prefixes (see Section 5.6), it could be used in concert with theirs.

There are a number of informative related works regarding applications of our system, *e.g.,* active scanning and probing of the IP address space. They are pertinent in that *(a)* they offer the performance necessary for scanning at large scale [8] or *(b)* they find that address discovery by passive monitoring significantly improves target selection, and therefore efficiency and/or coverage, in subsequent active scanning campaigns [3,4]. Most recently, Gasser *et al.* [13] focused specifically on the challenge of generating hit lists for scanning the IPv6 address space based on IPs gleaned from a number of common sources, some of which we use as well. They also contribute IPv6 support in `zmap` [8], suggesting IPv6 scanning is feasible, but do not contribute a strategy to algorithmically generate candidate targets. The Shodan search engine resorted to infiltrating the NTP server pool to discover active IPv6 host addresses that it subsequently probed [15], presumably since sequential or random full scanning of the IPv6 space is infeasible. Our work differs from these in that we probabilistically generate hit lists of targets not yet seen.

3. DATASETS

For our study, we used 3.5 billion IPv6 addresses total, all of which were collected in Q1 2016 from several data sources. We assembled both small sets for various real-world networks and large sets, aggregated by type. Table 1 summarizes the smaller datasets of 3 types: Servers, Routers, and Clients (end-users). In each category, we arranged IPv6 addresses (IPs) into individual sets for each of 5 major operators.

Type	**ID**	**Data Sources**				
		DNSDB	FDNS	rDNS	TR	CDN
Servers	S1	110 K	180 K	-		
	S2	290 K	4.7 K	-		
	S3	7.5 K	65 K	-		
	S4	12 K	5.7 K	-		
	S5	33 K	1.7 K	30 K		
	AS	790 K				
Routers	R1	-	28 K	1.8 K	6.7 M	
	R2	-	55 K	-	180 K	
	R3	460	10 K	11 K	7.5 K	
	R4	50	-	2.5 K	900	
	R5	10	-	1.3 K	380	
	AR				12 M	
Clients	C1					83 M
	C2					8.2 M
	C3					530 M
	C4					39 M
	C5					43 M
	AC					3.5 G

Table 1: Number of unique IPv6 addresses in "Small" datasets of IPv6 addresses (`S*`, `R*`, and `C*`), and in "Aggregate" datasets (`AS`, `AR`, and `AC`).

For Servers (`S1-S5`), `S1` represents a web hosting company, `S2` and `S3` represent two different CDNs, and `S4` represents a certain cloud provider. The operator of `S5` is commonly known for offering all of these services, and for providing many public web services, *e.g.*, a social network. Among Routers (`R1-R5`), all datasets represent router interfaces of major global Internet carriers. For Clients (`C1-C5`), all datasets represent leading ISP networks that deliver wired and mobile Internet access for domestic and enterprise customers.

To collect Server addresses, we employed the Domain Name System (DNS), as it is most common to rendezvous with services by domain name. For Routers, we used DNS and a large-scale traceroute dataset (column "TR" in Table 1), comprised of router interface addresses on paths between servers of a major CDN, and from these servers to some clients. For Clients, we used addresses of the clients involved in web requests to the CDN, during 17-23 March 2016. In order to evaluate with generally available data, we also collected client addresses via the BitTorrent network.

The first data source in Table 1 (column "DNSDB") presents the number of addresses found via DNSDB: a large DNS database offered by Farsight Security [11] that passively collects DNS data worldwide. It offers a broad view of queries and responses, allows for fetching forward records by host or network address (an "inverse" query), and resolves wild-card queries (*e.g.*, `*.ip6.isp.net/PTR`). For Servers, we queried DNSDB for prefixes used by operators for their IPv6 services, inferred from WHOIS and BGP data. For Routers, besides prefixes, we used wild-carded forward and reverse domain queries. We restricted the router IPs gleaned from DNS to those that appeared in our traceroutes.

For the second data source (column "FDNS"), we applied analogous techniques on the Forward DNS dataset by Rapid7 Labs [29]. This dataset is periodically recreated by actively querying DNS for domains found in various sources (including TLD zone files, Internet-wide scans, web pages, etc). For the last DNS data source (column "rDNS") we applied the technique described in RFC 7707 [14, pp. 23] by Gont and Chown that leverages DNS reverse mappings for obtaining IPv6 addresses of a particular network.

We also collected *aggregate* datasets for each category: `AS` for Servers, `AR` for Routers, and `AC` for Clients. As data sources, we used DNS for `AS` (790K IPs in 4.3K /32 prefixes), large-scale traceroute measurements for `AR` (12M IPs in 5.5K /32 prefixes), and 7-day CDN traffic for `AC` (3.5 billion IPs in 6.0K /32 prefixes). The aggregates cover the individual sets presented in Table 1. In order to avoid some networks from being over-represented, in Section 5.1, we used stratified sampling by randomly selecting 1K IPs from the /32 prefixes.

Inspired by work of Defeche and Vyncke [5], we also collected an aggregate of client addresses from the public BitTorrent Peer-to-Peer (P2P) network: dataset `AT`. To collect these, we built a custom BitTorrent client that crawled various trackers and Distributed Hash Table (DHT) peers during 5-8 March 2016. We collected 220K peer addresses in 1.8K /32 prefixes by running our software from Singapore, US East Coast, and Europe.

We employ address anonymization when presenting results. We changed the first 32 bits in IPv6 addresses to the documentation prefix (`2001:db8::/32`), incrementing the first nybble when necessary. To anonymize IPv4 addresses embedded within IPv6 addresses, we changed the first byte to the `127.0.0.0/8` prefix.

4. METHODOLOGY

In this section, we introduce our system by its visual interface, and then we detail our underlying methodology. In Fig. 1, we present the analysis results for a set of 24K WWW client addresses in a Japanese telco's prefix, collected from a CDN during a week's time.

The main components of Entropy/IP's visual interface are as follows. First, Fig. 1(a) plots entropy per address nybble, across the dataset. (We detail this in Section 4.1.) Here, the trained eye can see that the addresses are covered by one /40 prefix. In short, the address segments—delineated by dashed vertical lines and labeled with capital letters *A* through *K* at the top—are comprised of nybbles having similar entropy. Apart of that, we always make bits 1-32 the segment *A*. (We detail this in Section 4.2.)

Second, Fig. 1(b,c) are examples of Entropy/IP's conditional probability browser. Here, we show the distributions of values inside segments by a colored heat map. (We detail this in Section 4.3.) For example, segment *A* always has the value `20010db8`, which is reflected in 100% probability. In this example, the length of segment *C* is two nybbles, in which four distinct values were observed: the most popular being `10` at 60% in Fig. 1(b). Ranges are shown as two values (low to high) within one colored box, *e.g.,* segment *J* having an interval of `0000ed18068` to `fffb2bc655b` at 40%.

In the transition from Fig. 1(b) to Fig. 1(c), the analyst is curious how the probabilities would change if one conditioned on the segment *J* having the value `00000`.... Clicking on this value yields Fig. 1(c), showing for instance that now *C* has the value `10` at 100%, and likewise for value `0` in segments *H* and *I*.

Fig. 2 shows the structure of an associated Bayesian Network (BN), with nodes representing the segments and edges indicating a statistical dependency. (We detail this in Section 4.4.) Here, the red edges show that the segment *J* is directly dependent on segments *C* and *H*, which is analyzed in Table 2. The segments can influence each other in the opposite direction and through other segments. Thus, selecting a particular value for *J* influences *F* through *C*, which is the reason for different distribution for the segment *F* in Fig. 1(c) vs. Fig. 1(b).

A live, functional demo of Entropy/IP interface is publicly available at http://entropy-ip.com/.

4.1 Entropy Analysis

Entropy is a measure of unpredictability in information content [30]. Usually it is defined as $H(X)$

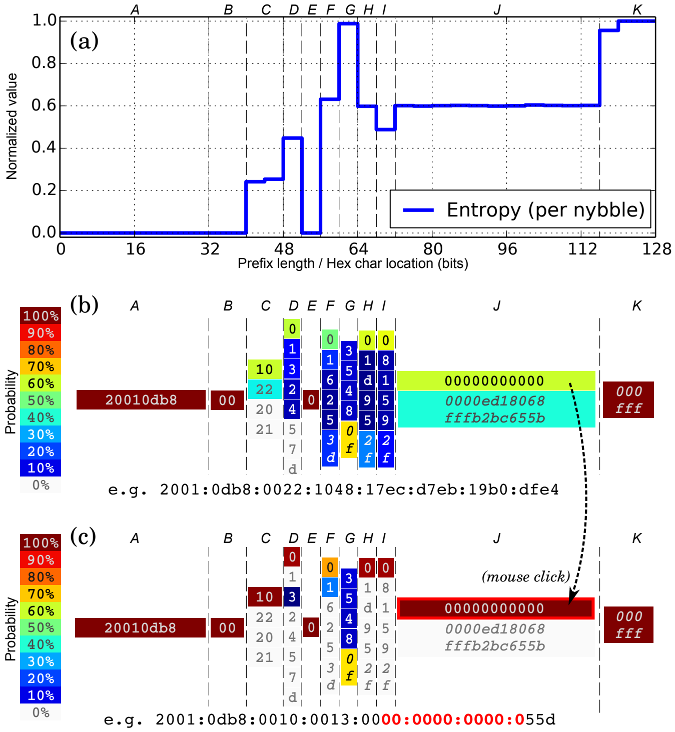


Figure 1: Entropy/IP's user interface displaying an analysis of a Japanese telco prefix with 24K active client IPs. Entropy by nybble plotted in (a). In (b), we select the `00000`... value (60%) for segment *J* by mouse click, resulting in updated probabilities in (c) (*e.g.,* 100%).

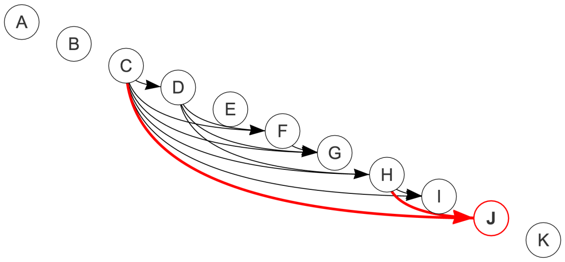


Figure 2: Dependencies between segments. Red color indicates direct probabilistic influence on segment J.

H	C 10	 22	 20	 21
0	100%	0.48%	7.7%	14%
1	17%	0.17%	6.3%	13%
d	25%	0.17%	7.7%	5.9%
9	17%	0.17%	9.1%	9.1%
5	20%	0.17%	7.1%	7.1%
2-f	2.3%	0.017%	0.79%	0.77%

Table 2: Probability for segment *J* in Fig. 2 equal `00000`..., conditional on values in segments *H* and *C*.

for a discrete random variable X with possible values $\{x_1, \cdots, x_k\}$ and a probability mass function $P(X)$:

$$H(X) = -\sum_{i=1}^{k} P(x_i) \log P(x_i). \tag{1}$$

In general, the higher the entropy, the more equally probable values of X are: if $H(X) = 0$, then X takes only one value; if $H(X)$ is maximum, then $P(X)$ is uniform. For the remainder of the paper, we normalize entropy by dividing it by $\log k$ (maximum value).

In order to use entropy for analyzing the structure of IPv6 addresses, let D be a set of addresses expressed as 32 hexadecimal digits without colons, *e.g.*, as in Fig. 3.

```
0        1         2         3
12345678901234567890123456789012
20010db840011111000000000000111c
20010db840011111000000000000111f
20010db84003 1c13000000000000200c
20010db8400a2f2a000000000000200f
20010db840011111000000000000111f
```

Figure 3: Sample IPv6 addresses in fixed-width format, sans colon characters.

Let us consider the values in the i-th hex character position of the addresses in D as instances of a random variable X_i, $i = 1, \ldots, 32$. That is, let us focus on values of a specific nybble across all addresses in D. In Fig. 3, the last character takes "c" twice and "f" thrice. Thus, X_{32} has empirical probability mass function $\hat{P}(X_{32}) = \{p_c = \frac{2}{5}, p_f = \frac{3}{5}\}$. Since there are 16 possible hex characters, the maximum entropy is $\log 16$, and thus the normalized empirical entropy is:

$$\hat{H}(X_{32}) = \frac{-(p_c \log p_c + p_f \log p_f)}{\log 16} \approx 0.24. \tag{2}$$

By repeating the above for all i, we get 32 values that reveal statistical properties of individual hex characters across the set: the smaller the value, the greater chances the character stays constant. Let us also introduce a notion of *total entropy* $\hat{H}_S$,

$$\hat{H}_S(D) = \sum_{i=1}^{32} \hat{H}(X_i), \tag{3}$$

which quantifies variability of addresses in D, *i.e.,* how hard it is to guess actual addresses by chance[2].

4.2 Address Segmentation

Entropy exposes the parts of IPv6 addresses that are variable versus those that remain relatively constant. Let us use it to group adjacent hex characters into contiguous blocks of bits with similar entropy. We will call such blocks *segments* and label them with capital letters. For instance, in Fig. 3, the hex characters 1-11 and 17-28 are constant (entropy = 0), whereas the values in hex characters 12-16 and 29-32 are changing (entropy $\neq$ 0). Hence, they form 4 segments: A (1-11), B (12-16), C (17-28), and D (29-32). By segmenting IPv6 addresses, we distinguish contiguous groups of bits that differ in joint variability.

We propose a simple threshold-based segmentation algorithm. Consider the entropy of successive nybbles, left to right. Start a new segment at nybble i whenever $\hat{H}(X_i)$ compared with $\hat{H}(X_{i-1})$ passes through any of the thresholds $T = \{0.025, 0.1, 0.3, 0.5, 0.9\}$. We also employ a hysteresis of $T_h = 0.05$, *i.e.*, we require

$$|\hat{H}(X_i) - \hat{H}(X_{i-1})| > T_h \tag{4}$$

to start the new segment. For example, if $\hat{H}(X_{i-1}) = 0.49$, then in order to start the next segment $\hat{H}(X_i)$ has to be either <0.3 or >0.54, with 0.3 being the lower threshold for $\hat{H}(X_{i-1})$ in T (without hysteresis) and 0.54 being $\hat{H}(X_{i-1}) + T_h$ (with hysteresis.) We found this set of parameters T and T_h during development by evaluation on real-world networks. The parameters can be tuned to match specific networks, yet we identified the proposed values to be universal and produce least number of segments with similarly distributed nybbles.

In addition to the thresholds, we always make the bits 1-32 a separate segment. This is motivated by the common practice of Regional Internet Registries (RIRs), who use a /32 prefix as the smallest block assigned to local Internet operators [1]. Similarly, we always put a boundary after the 64th bit, as it commonly separates the network identifier from the interface identifier [17].

4.3 Segment Mining

Further in our analysis, we want to understand why some segments appear non-random (*i.e.*, have entropy <1). We hope to uncover common elements of IPv6 addresses, which possibly have a semantic meaning [18].

Let us delve into a specific segment k. First, reduce the input dataset D down to D_k: for each address, drop all nybbles outside of segment k. Next, search D_k for the set V_k of popular values and ranges that cover considerable parts of D_k. For example, in Fig. 3 the segment of nybbles 12-16 has $D_k = \{$11111, 11111, 31c13, a2f2a, 11111$\}$ and $V_k = \{$11111$\}$.

We propose a heuristic approach for building V_k that focuses on three aspects of data: *(a)* frequencies of values, *(b)* the values themselves, and *(c)* both characteristics considered together. We address them separately in the steps described below. In each step, we nominate at most the top 10 elements to V_k and remove them from D_k. If there is ≤0.1% of values left, we finish.

For *(a),* we use a well-known outlier detection method to find unusually prevalent values in D_k. Assuming normal distribution of *frequencies* of values, we select the values more common than $Q_3 + 1.5{\cdot}IQR$, where Q_3 is the third quartile and IQR is the inter-quartile range. For example, see the values labeled C1 through C5 in

[2] Note: "total entropy" is not the entropy if one considered the whole address as a single element, and computed the probability mass function of those elements—that entropy, normalized, would be very low.

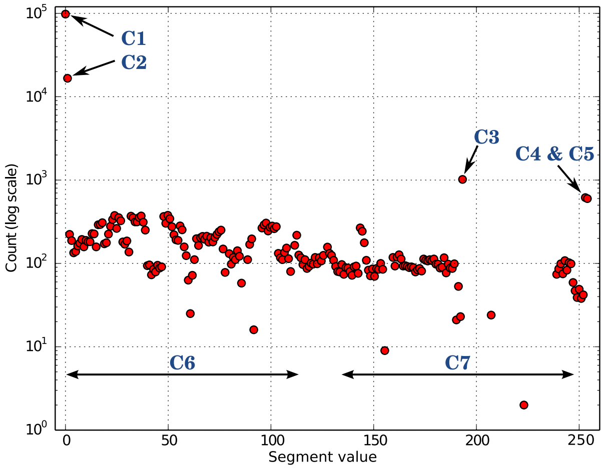


Figure 4: Histogram for values in segment C of dataset S1, as a scatter plot. Annotations show codes of common values and ranges.

Fig. 4. In this example, the segment has length of two nybbles (thus 256 possible values), which is the X-axis in Fig. 4, and the Y-axis is the number of times a given value appears in set D_k. Next, for *(b),* we run on D_k the popular DBSCAN data clustering algorithm [10], parametrized to find highly dense ranges of values. In this step, we use the minimum and maximum values of the discovered clusters as ranges added to V_k. Then, for *(c),* we run the DBSCAN algorithm again, but on a histogram of D_k, that is, on a vector of values vs. their counts. We tune the algorithm to find ranges of values that are both uniformly distributed and relatively continuous (*e.g.*, C6 in Fig. 4). Finally, if anything is left, we either close V_k with a range of $(\min D_k, \max D_k)$, or if $|D_k| \leq 10$ we take the whole D_k.

We preserve the order of elements added to V_k and we keep their empirical frequencies. In Table 3, we present the result of our algorithm for the S1 dataset. The column "Code" gives labels for segment values, which can be used to encode (or "compress") addresses, *e.g.*,

`2001:0db8:08c2:2500:0000:d9a0:5345:0012`

can be rewritten using Table 3 as a vector:

$$(\text{A1, B2, C3, D4, E5, F1, G12, H1, I2, J3}).$$

Note that we lose details of the original address when we encode a segment with a code that represents a range (*e.g.*, G12); this is acceptable for our purposes. In further discussion we represent IPs as instances of random vectors, where each dimension corresponds to segment k and takes categorical values that reference V_k.

4.4 Modeling IPv6 Addresses

Having the addresses represented as random vectors enables us to easily apply well-known statistical models. Consequently, we can reveal inter-dependencies between the segments, and uncover structures hidden within IPv6 addresses. For this, we use Bayesian Networks (BNs).

Seg.	Code	Value	Freq.
A (1-32)	A1	20010db8	63.50%
	A2	30010db8	36.50%
B (32-40)	B1	10	77.80%
	B2	08	15.42%
	B3	09	5.05%
	B4	07	0.70%
	B5	00	0.55%
	B6	05	0.47%
C (40-48)	C1	00	67.02%
	C2	01	11.13%
	C3	c2	0.67%
	C4	fe	0.41%
	C5	ff	0.41%
	C6	*02-5b*	11.94%
	C7	*5c-fd*	8.42%
D (48-52)	D1	0	10.10%
	D2	5	9.24%
	D3	4	9.11%
	D4	2	9.05%
	D5	1	8.90%
	D6	*3-f*	53.61%
E (52-56)	E1	0	69.69%
	E2	1	5.41%
	E3	2	4.72%
	E4	3	3.75%
	E5	5	2.23%
	E6	*4-f*	14.20%
F (56-64)	F1	00	14.18%
	F2	53	0.65%
	F3	*01-ff*	85.17%

Seg.	Code	Value	Freq.
G (64-116)	G1	000000000000	0.29%
	G2	0127016000630	0.11%
	G3	0127020800160	0.11%
	G4	0127020801800	0.08%
	G5	0127007100620	0.07%
	G6	0127022700290	0.06%
	G7	0127016001550	0.06%
	G8	0127016001130	0.06%
	G9	0127016000620	0.06%
	G10	0127022702170	0.06%
	G11	*0000000000001-* *0000000000af0*	13.02%
	G12	*0000d9a050050-* *0000d9a053f90*	0.39%
	G13	*0127022701090-* *0127022701270*	0.20%
	G14	*010332b0b1e17-* *fffd8c3ab1643*	84.90%
	G15	*0000000001a10-* *00fd12c41fce6*	0.55%
H (116-120)	H1	0	49.51%
	H2	8	37.35%
	H3	*1-f*	13.14%
I (120-124)	I1	0	51.62%
	I2	1	19.90%
	I3	2	9.63%
	I4	3	4.46%
	I5	4	2.38%
	I6	*5-f*	12.02%
J (124-128)	J1	0	16.44%
	J2	1	8.20%
	J3	2	7.69%
	J4	3	6.93%
	J5	4	6.54%
	J6	*5-f*	54.21%

Table 3: Segment mining results for dataset S1.

BN is a statistical model that represents jointly distributed random variables in the form of a directed acyclic graph [20]. Each vertex represents a single variable X and holds its probability distribution, conditioned on the values of the other variables. An edge from vertex Y to X indicates that X is statistically dependent on Y. BN can be used to model complex phenomena involving many variables. It splits complex distributions into smaller, interconnected pieces, which are easier to comprehend and manage.

Let us find a BN that represents a dataset of IPv6 addresses rewritten as random vectors. We need to learn the BN structure from data (*i.e.,* discover statistical dependencies), and we need to fit its parameters (*i.e.,* estimate conditional probability distributions). For this purpose, we use the "BNFinder" software, which implements the relevant state-of-the-art methods [6, 35]. Since learning BNs from data is generally NP-hard, we constrain the network so that given segment k can only depend on previous segments $<k$, *e.g.,* B can directly depend on A, but not on C. However, note that C can still influence B through evidential reasoning; that is, probabilistic influence can flow "backwards." We refer the reader to [20] for more details on BNs.

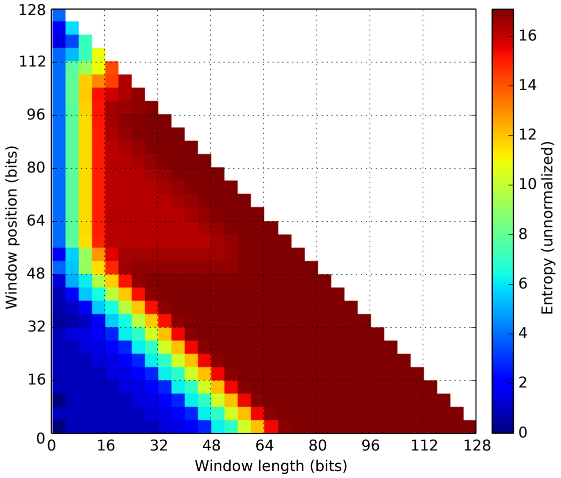


Figure 5: An illustration for the preliminary idea of windowing analysis of entropy (dataset S1).

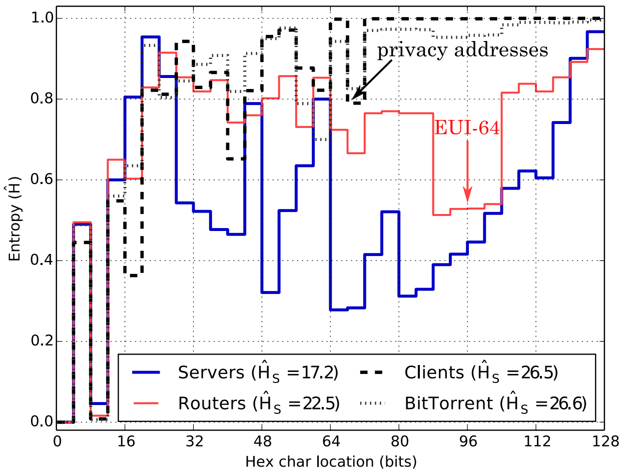


Figure 6: Entropy of aggregate datasets (A*).

Once the BN model is found, we may use it for multiple purposes. For example, we may query the BN with various segment values set *a priori* and discover how this affects the probability distributions in the other segments. We may also use the BN to generate candidate addresses that match the model (optionally constrained to certain segment values), which can be used for targeted scanning of IPv6 networks.

4.5 Discussion

We presented a heuristic system that, given a set of IPv6 addresses, discovers address segments with their popular values and a probabilistic structure. Obviously, we did not fully explore the design space possible for such a system; thus, our viewpoint is neither definitive nor the only possible. In order to encourage further development of similar tools, below we comment on our design choices and on possible adaptations.

We chose to focus on 4-bit chunks of IPv6 addresses, *i.e.*, nybbles. It is possible to adapt Entropy/IP to other bit widths by modifying the entropy analysis step. For instance, one could use the bit widths of 1 or 16, as used by Plonka and Berger in [27] for MRA plots. However, we found the 4-bit approach simple and sufficient. If more granularity is needed, then the segment mining step can discover individual values. Conversely, if less granularity is needed, then the address segmentation step can coalesce adjacent nybbles. Besides, the 4-bit granularity matches the textual representations of IPv6 addresses, in which a single hex character is the smallest possible address chunk, thus matches the human analyst's canonical perspective.

Let us briefly mention a preliminary idea of *windowing analysis*, devised during early development of Entropy/IP. For a set of IPv6 addresses, we evaluated the entropy (unnormalized) for every possible address segment, determined by windows of varying length and position. For example, in Fig. 5 we visualize such an analysis for dataset S1: every (X, Y) point on the plane shows the entropy calculated across the dataset for bits Y through $Y+X$. Here, note that one could use a different variability measure than the entropy, *e.g.*, number of distinct values, inter-quartile range, frequency of the most popular value, or a weighted mean thereof. We believe this may be especially useful in conjunction with the windowing analysis for visual discovery of patterns.

We also tried segmenting using the difference between entropy of adjacent nybbles. However, taking into account the number of segments and distributions of their nybbles, the simple thresholds algorithm performed better. Note that segmentation without *a priori* knowledge is problematic. Judging just by the apparent features of an addressing space may lead to uncovering the intents of network administrators, but may impose an artificial model on the data as well. However, we believe the segment mining and BN modeling steps together can compensate for minor glitches in segmentation.

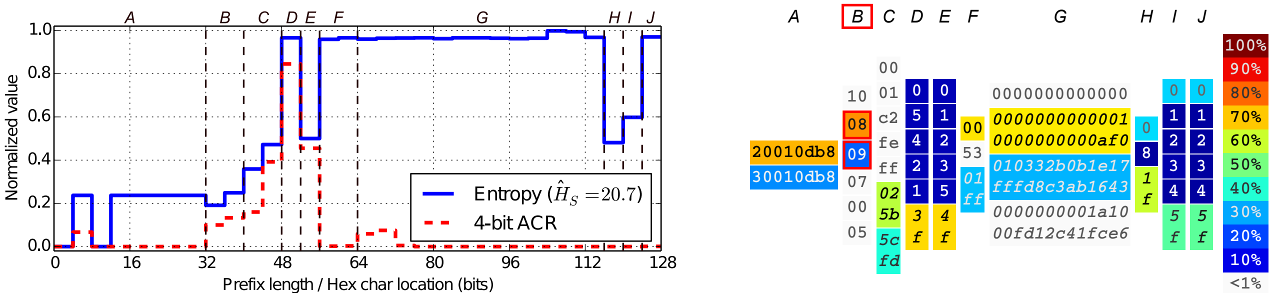
Finally, we considered other tools (than BNs) for modeling the IPv6 addresses, *e.g.*, Probability Trees (PTs) [7] and Markov Models (MMs) [28]. PTs, although conceptually simple, require information on virtually every possible combination of the segment values, and hence need abundant training data. The other alternative, MMs, assume that a given segment depends only on the previous segment. Thus, MMs cannot directly handle dependency between non-adjacent segments. Overall, we found BNs to produce models that are powerful, easy to explore, and concise.

5. EVALUATION

In this section, we demonstrate the efficacy of our methods on the datasets introduced in Section 3.

5.1 Big Picture: Aggregates

In Fig. 6, we present entropy characteristics for IPv6 addresses in our aggregate datasets, for three types of hosts. We see that the characteristics differ considerably past the first 32 bits; this is reflected in the $\hat{H}_S$



(a) entropy vs. 4-bit ACR

(b) BN probabilities conditional on B equal 08 or 09 (skipped probabilities <0.1% in G for brevity)

Figure 7: Results for server dataset S1.

metric. Among the datasets, the client addresses were the most random, especially in the bottom 64 bits, ostensibly the IPv6 interface identifier. We see $\hat{H} \approx 1$, with the exception of $\hat{H} \approx 0.8$ for bits 68-72. This largely matches the RFC specifications for the "u" bit in SLAAC addresses [26]. However, if all of the clients used temporary addresses, we would observe an entropy of 0.75 for bits 68-72. Hence, some of these addresses were not standard SLAAC privacy addresses.

We see a similar yet smaller dip for router addresses in bits 68-72, and a deeper drop to $\hat{H} \approx 0.5$ in bits 88-104. This suggests impact of IPv6 addresses based on link-layer identifiers (e.g., Ethernet MAC address), which have the word `0xfffe` inserted in bits 88-104 [17]. Again, if all of the router addresses followed this standard, we would observe $\hat{H} = 0$. Hence, a major portion of router addresses did not have MAC-based Modified EUI-64 IIDs.

On the other hand, in Fig. 6, we see no evidence of Modified EUI-64 or privacy SLAAC addresses for servers. Instead, we highlight an interesting phenomenon: the entropy is oscillating across the address, revealing popular locations for discriminators of various entities in the addressing hierarchy (*e.g.,* subnetworks). In general, the addresses in dataset AS are the least random, compared with the other sets. This demonstrates that servers are more likely to have predictable addresses. Note the steady increase in entropy from bit 80 to 128; this reflects the tendency to use the lower order bits when assigning static addresses to servers.

Finally, in Fig. 6, we do not find significant differences between client addresses collected from CDN (dataset AC) vs. collected from the BitTorrent network (dataset AT), except for bits 88-104, which suggests Modified EUI-64 SLAAC addressing is more common for BitTorrent clients than general web clients. Thus, we believe it is possible to glean structure in client IPv6 networks using such freely-available data sources.

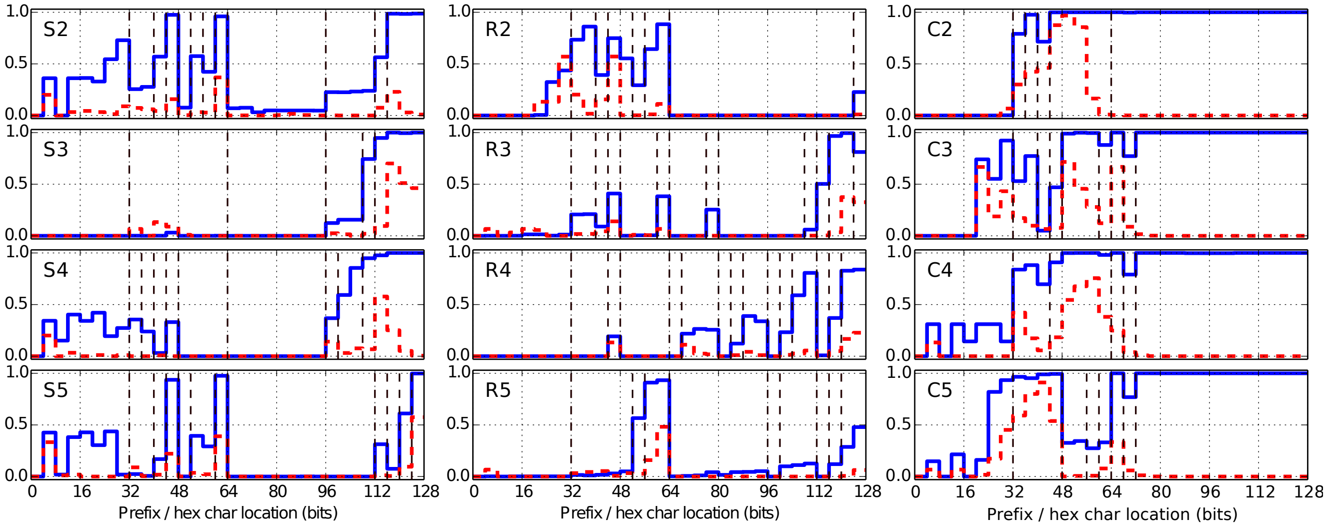
5.2 Servers

In Fig. 7, we present analysis results for dataset S1—the addresses of a major web hosting provider—in which we discovered 10 segments. The BN model, visualized in Fig. 7(b), shows probability distributions across these segments, conditioned on *B* equal to either *B2* or *B3* (real values 08 or 09, respectively), which represents approx. 20% of addresses in S1.

The network has two /32 prefixes, which differ in popularity: 64% vs. 36%. Their actual values are anonymized in Fig. 7(b), but in Fig. 7(a) we see in segment *A* that the two actual prefix values differ in six hex characters; their entropy is non-zero. ACR is non-zero only for bits 4-8, which means each /8 prefix holds just one /32 prefix. By exploring the BN model, we found the addressing scheme largely the same for both prefixes, but *B1* is 10% more likely for *A2* than for *A1*.

We further found that segment *B* selects a variant of addressing used on the lower bits: probability distributions differ for *B1* vs. *B2/B3*, vs. *B4/B6*, vs. *B5*. In other words, the network has 4 variants of addressing deployed across its /40 prefixes. For example, in Fig. 7(a) we see high entropy in segments *F* and *G* (*i.e.,* high variability in bits 56-116). However, when we constrained the segment *B* to *B2/B3*, in Fig. 7(b), we find a major drop in the variability of bits 56-116: the majority of addresses in this variant are essentially non-random. For the *B4/B6* variant, we found 67% of IPv6 addresses encode literal IPv4 addresses in segments *G-J*. We verified these IPv4 addresses belong to the same operator and respond to ping requests with similar round-trip times as the IPv6 addresses in which they were embedded. (This strongly suggests these IPv6/IPv4 address pairs are aliases on dual-stacked hosts.)

Next, in Fig. 7(a), we see that segments *C* through *E* have high values for both entropy and ACR. This ostensibly means that bits 40-56—apart from being variable—are utilized for discriminating prefixes. In contrast, for segment *F* (bits 56-64), we see high entropy with ACR near zero. In this area, the address is variable, but appears to carry little useful information to discriminate addresses from one another; typically, each /56 prefix, here, covers just a single active *random* /64 prefix. We observe a similar phenomenon for segments *G-J* (i.e., bottom 64 bits): each /64 prefix contains just a few IPv6 addresses. A subset of interface identifiers appears pseudo-random, *e.g.*, see G14 in Table 3. However, due



(a) servers (b) routers (c) clients

Figure 8: Brief plots for server datasets S2-S5 (a), router datasets R2-R5 (b), and client datasets C2-C5 (c). Solid blue lines show per-nybble entropy and dashed red lines show 4-bit ACR. Segment labels skipped for brevity.

to non-random addressing variants discussed above, the entropy in Fig. 7(a) is below 1. Note no drop in entropy for bits 68-72 (characteristic for SLAAC), which suggests the operator has its own algorithm for generating interface identifiers. Moreover, entropy in segments *H*-*J* reveals a structure in the addresses, but without any effect on ACR, thus, unlikely to be subnetting.

Due to space constraints, we briefly present entropy analysis for the rest of server datasets in Fig. 8(a). The addresses exhibit less variability across hex characters and ostensibly do not use SLAAC, which is consistent with our observations for AS. Plots for S2 and S3 demonstrate addressing effects for different types of CDNs; the first network distributes traffic using DNS and IP unicast, while the second employs IP anycast. In consequence, S2 has many globally distributed prefixes, while S3 basically uses just one /96 prefix worldwide, which is reflected in entropy plots. For S4, a major cloud provider, we found that—apart of a simple structure in bits 32-48—only the last 32 bits are utilized for discriminating hosts and networks. For S5, we found that the last 2-4 nybbles often identify the service type (or type of content), deployed across many /64 prefixes (inferred by manual analysis of DNS records). In sum, our analysis identified structure in all evaluated sets.

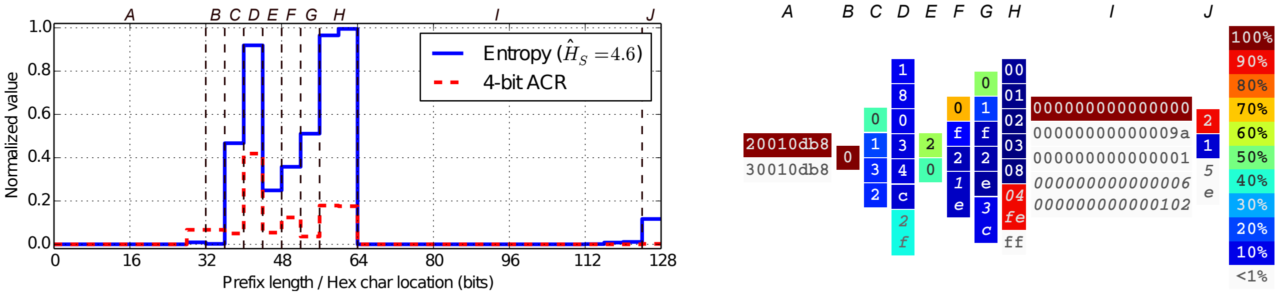
5.3 Routers

Fig. 9(a) illustrates analysis of the R1 dataset. We see a clear division between bits used ostensibly for discriminating prefixes (bits 28-64) and for discriminating interfaces within the prefixes (mostly bits 124-128). The network, clearly, does not implement pseudo-random interface identifiers, which is visible in entropy close to zero for bits 64-124. By evaluating the BN model, visualized in Fig. 9(b), we find that segment *I* is largely a string of zeros, and the last hex character is either 1 or 2, which we believe is common for point-to-point links between this network's routers.

Consider the analysis of the other router datasets, briefly presented in Fig. 8(b). For R2, we find a similar pattern as in R1: bottom 64 bits equal either 1 or 2. For R3, we find bits 48-116 to follow a quite predictable pattern, with a majority of the hex characters equal to zero. We see that bits 32-48 discriminate prefixes, and the last 12 bits largely appear pseudo-random. However, note that values that look random do not necessarily come from a random number generator: the administrator could be systematically assigning values to these nybbles in near equal proportions. Interestingly, in R4, we find the IPv6 interface identifiers encode literal IPv4 addresses (ostensibly assigned to the same router interface), written as octets in base 10 across 16-bit aligned words (*i.e.,* colon-separated in IPv6 presentation format), hence the IID pattern visible in Fig. 8(b) for R4. Finally, for R5, we find the addresses to discriminate largely in bits 52-64, while the bottom bits follow a predictable structure. Overall, we find the router addresses to implement simple patterns, yet unique and variable across operators.

5.4 Clients

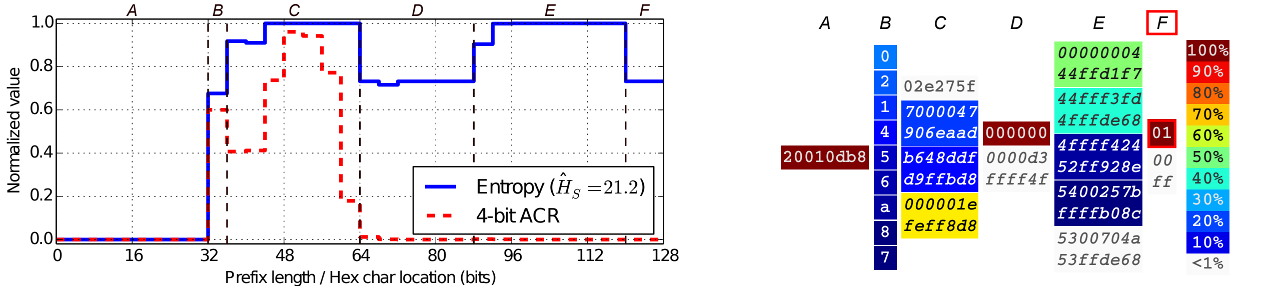
In Fig. 10(a), we present the analysis for dataset C1. The addresses correspond to a large mobile operator. Our analysis finds only six segments in the addresses, three of which uncover statistical structure in interface identifiers. In Fig. 10(b), we visualize the BN model conditioned on the last two hex characters equal to 01, which corresponds to 47% of all IPs in dataset C1.



(a) entropy vs. 4-bit ACR

(b) BN probabilities

Figure 9: Results for router dataset R1.



(a) entropy vs. 4-bit ACR

(b) BN probabilities conditional on F equal 01

Figure 10: Results for client dataset C1.

In segments *B*-*C*, we see that bits 32-64 discriminate prefixes, as the ACR value is relatively high. Segment *B* takes only lower values (0-8), hence its entropy is lower than in segment *C*. For segments *D*-*F* we expected to see the impact of privacy addressing, already observed for the client aggregate (Fig. 6). Instead, we find an extraordinary pattern of entropy close to 0.7 for segments *D* and *F*, and entropy close to 1 for the segment *E* between them. The lower entropy in *D* and *F* is due to two popular values: 00000 (*D1*) and 01 (*F1*). Further, we find (in the BN model) that segments *D* and *F* are statistically dependent, which is visible in Fig. 10(b): conditioning the BN model on *F1* makes *D* a string of zeros (*i.e.*, *D1*). We also found dependence between *E* and *F*, which makes the first hex character in segment *E* more predictable when the addresses end with *F1*, *i.e.*, 00 (manifested in lower entropy for bits 88-92). On the other hand, when addresses end with other values, all bottom 64 bits are pseudo-random.

In order to investigate the underlying source of the pattern, we examined log files of the CDN corresponding to dataset C1. By "User-Agent" headers in web requests made to the CDN from that particular network, we find the addresses that follow the pattern originate only from Android devices of one popular vendor. In contrast, the addresses without the pattern originate from mobile devices of various vendors. Thus, we suspect the reason for what we see in Fig. 10(b), lies in client-side interface identifier selection in a specific version of Android built by a particular vendor for that network. We did not observe similar patterns in any other evaluated dataset.

Fig. 8(c) displays entropy vs. ACR plots for the other datasets of active client addresses. For these datasets, we found the interface identifiers to be pseudo-random, *e.g.,* as in privacy addresses; this is visible where entropy is near 1 and ACR is near 0 in the low 64 bits. This matches our findings for dataset AC. However, we did not observe a drop in entropy on bits 68-72, characteristic for SLAAC privacy addresses, for dataset C2, which represents a mobile operator. In all client addresses, we found various structures in bits 32-64 (or even 20-64), which demonstrates that operators implement their own addressing schemes for the /64 prefixes. Below we show some of these structures are predictable.

5.5 Scanning IPv6 Networks

One of the supplementary applications of our method is candidate address generation for active measurements of the IPv6 Internet. In this subsection, we evaluate Entropy/IP for scanning addresses of servers and routers.

For scanning IPv6 addresses, we apply the following methodology. A BN model for a particular network is trained on a random sample of 1K real IPv6 addresses known *a priori*. Then, the model is used to generate 1M candidate targets for scanning, and the result is evaluated. For our experiments we used the datasets S* and R*: for each of them, we randomly selected 1K IPs as the training set, and used the remaining part as the testing set.

Dataset	Found IPv6 addresses				Success rate	New /64s
	Test set	Ping	rDNS	Overall		
S1	0	0	0	0	0.0%	0
S2	6.4 K	160 K	29	160 K	16%	12 K
S3	62 K	430 K	0	430 K	43%	0
S4	480	23 K	0	23 K	2.3%	0
S5	44 K	53 K	18 K	66 K	6.6%	1.2 K
R1	20 K	33 K	37 K	44 K	4.4%	24 K
R2	14 K	9.7 K	22	19 K	1.9%	8.2 K
R3	11 K	10 K	16 K	19 K	1.9%	70
R4	1.6 K	400	1.7 K	1.7 K	1.7%	0
R5	4.3 K	3.3 K	2.3 K	5.5 K	0.55%	380
	160 K	720 K	75 K	770 K		46 K

Table 4: IPv6 scanning results for Servers (S*) and Routers (R*). Models trained on 1K real addresses and tested on 1M generated addresses.

Dataset	Training sample size				
	100	1 K	10 K	100 K	1 M
S5	5.2%	6.7%	7.0%	2.9%	-
R1	4.1%	4.4%	4.4%	-	-
C5	16%	20%	21%	20%	14%

Table 5: Success rate vs. training set size.

We counted the number of generated candidates that were present in the testing set, which is presented in column "Test set" in Table 4. Independently, for each dataset we scanned the 1M generated IPs using ICMPv6 echo requests: we show the number of IPs that replied with an echo response in column "Ping". Similarly, column "rDNS" gives the number of IPs that had a reverse DNS record. (We manually removed records that appeared dynamically generated.) In column "Overall," we present the number of candidate IPs that passed at least one of the three tests, with the corresponding success rate in the next column. For these IPs, in column "New /64s" we give the number of /64 prefixes that were not present in the training data. Note that some networks use just a few prefixes globally (*e.g.*, S3).

As visible in Table 4, in total we found 770K addresses and 46K prefixes in 10 networks. We used only 1K training IPs per network, which corresponds to a considerable gain in discovered IPs vs. what we used for training the BN models. We believe this resembles a relatively common situation in which one has a limited set of existing IPs from the target network and wishes to use them to bootstrap active address discovery. We found it possible to find new addresses for each dataset except for S1, which uses largely pseudo-random IIDs for a major subset of its addresses. On the other hand, we achieved 43% success rate for S3 (a large CDN). Note the probability of guessing by chance an IPv6 address in a known /32 prefix is approx. 1 in 2^{96}.

We acknowledge some limitations of our evaluation method: first, part of the positive responses to our "Ping" and "rDNS" tests might have been generated automatically (*e.g.* replying to any ping request destined to a certain prefix, causing false positives); second, the network may have far fewer hosts than the generated 1M candidates; finally, we might get a number of false negatives due to restricted datasets and due to networks blocking our ping requests and reverse DNS queries.

We believe our results indicate that Entropy/IP implements a fruitful approach to IPv6 scanning. Comparing to existing works, we find it complementary. The scan6 tool by Gont [31] and the method by Ullrich *et al.* [33] attempt to predict interface identifiers; they do not tackle guessing the network identifiers. In contrast, our statistical model successfully predicted active /64 prefixes not seen in the training sets.

In Table 5 we show the effect of the training set size on the success rate for a server dataset S5 and a router dataset R1. We found that a larger training set often does not cause better scanning performance and can even make it worse. We believe the reason for what we experimentally observed is that more training IPs make the BN model better at adhering to already seen data vs. generating completely new addresses.

5.6 Predicting IPv6 Client Prefixes

The addresses in the sample client networks (C1-C5) extensively use pseudo-random interface identifiers, and thus there is no point in trying to guess the full address. For this reason, in this subsection, we turn to predicting the prefixes, *i.e.*, generating the candidates for the first 64 bits in the client addresses.

For that purpose, we constrained Entropy/IP to the top 64 bits, without any other modification to our method. We used a similar evaluation approach as in the previous subsection. We trained the model on a random 1K sample of prefixes seen on March 17th 2016 and tested them on the rest of data: for each of 1M candidate prefixes, we checked if it was observed on the same day and in the following week, as seen in the dataset.

In Table 6, we present the results. We were able to predict thousands of client /64 prefixes for each network, with success rates ranging ∼1% to 20%. For some datasets we found it easier to predict the prefix for the whole week vs. single day (*e.g.*, C1), while for others

Dataset	Predicted /64s		Success rate (7-day)
	Mar 17	Mar 17-23	
C1	12 K	54 K	5.4%
C2	2.0 K	11 K	1.1%
C3	7.5 K	8.3 K	0.83%
C4	37 K	120 K	12%
C5	150 K	200 K	20%
	210 K	390 K	

Table 6: Prefix prediction results for Clients (C*), for two time spans. Models trained on 1K real /64 prefixes and tested on 1M generated prefixes.

we did not find such a difference (*e.g.*, C3). We believe this demonstrates that operators have various addressing strategies for the top 64 bits of their IPv6 addresses, some of which are easy to predict. Moreover, we were able to repeat the same using the BitTorrent AT dataset for training. For example, by training a BN model using 1K prefixes of BitTorrent peers belonging to the network represented in C4, we were able to predict 120K prefixes from that dataset.

Finally, in Table 5 we briefly show the effect of the training sample size for C5. Again, using a too large training set can harm the predictive performance.

5.7 Validation

To qualitatively validate our results, we prepared a briefing for each of the five networks that we studied, at least one from each of the Servers, Routers, and Clients categories, mentioned in Section 3. The briefings contained both a link to the Entropy/IP interactive user interface with an analysis of the given network and subjective assessments based on our interpretation. We emailed briefings to four subject matter experts (SME), each responsible for one of those IPv6 networks, and asked them to confirm, refute, or otherwise comment on our results, promising to publish them in anonymized fashion (herein). [3] Summarizing their feedback:

- The first SME, from a mobile carrier network, wrote "In short, I don't know. :)" They reported that they assign prefixes to gateways from a common vendor for service providers in the US, and these gateways in turn assign the /64 prefixes and hint about an IID to the user equipment, *i.e.,* SLAAC for 3GPP cellular hosts [21]. Regarding a detail, they were surprised that our analysis showed 47% of the IIDs end with 01, *i.e.,* apparently not pseudo-random. They hypothesized these addresses are selected by some Android handsets.
- The second SME wrote "[Your] assumptions when it comes [to] infrastructure are wrong." This SME said that internally they use a RIR block that is not routed on the Internet, and that their routers are in a block that is only used for that purpose. In the briefing, we guessed that the lowest IPv6 address bits sometimes identified services. The SME confirmed this is the case, but did not comment on many other assertions including our guesses as to the uses of prefixes containing "vanity" hexadecimal strings that spell English words in the midst of their IPv6 addresses.
- Another network's SME wrote, "of your analysis [...]: nice reverse engineering of our address plan ;)." They also mentioned some confusion because the initial segment A in the conditional probability browser showed a /32 prefix that contained some addresses that are not theirs. This was a mistake on our part, due to "hard-wiring" a segment boundary at /32 and incorrectly assuming that the entire /32 was allocated to them. The SME asked if we used the RIR databases to verify the netblock owners.
- The final SME responded to the details that we reported regarding values in particular segments, saying it would require digging to confirm or refute. They reported that our subjective assessments were correct, *e.g.,* about which prefixes were dedicated to routers or clients, but also mentioned that was "rather obvious." They also asked questions and generously offered to discuss the results further.

With regard to the SME suggesting it may be Android handsets that use curious IIDs in their network, independent investigation confirmed this (Section 5.4). In answer to the question about using RIR databases, we did consult them, but apparently not carefully enough. This was useful feedback as it shows we can improve our system by better integrating RIR data and BGP-advertised prefixes. With regard to some of our assessments being labeled obvious, perhaps the SME meant they are obvious to an IPv6 operator or expert.

Overall, we find these responses interesting in their variability with respect to how forthcoming operators are, or are not, with details of their address plan. We appreciate their candid, collegial responses, such as one even admitting, essentially, that some network gear "just works," and that they did not study the resultant SLAAC-based address assignments to user equipment. This comports with one of the stated applications of our work: namely, to remotely analyze networks' addressing practices, assess, and report potential risks.

6. LIMITATIONS AND FUTURE WORK

Some limitations arise from out initial assumptions and choice of parameter values for segmentation, described in Section 4.2. First, as mentioned in Section 5.7, one network's SME identifies a problem with an assumption we made to "hard-wire" a break in segments at a /32 boundary, *i.e.,* after bit 32. Sometimes this results in segments that disagree with the actual subnet identifiers in a network's address plan. This could, also, result from requiring segment boundaries to be between 4-bit aligned nybbles or from algorithmic sensitivity to configurable parameters, *e.g.,* the threshold and hysteresis values, which influence the segmentation process. While these assumptions do not necessarily prevent us from observing effects of features smaller than 4 bits nor from observing phenomena involving boundaries at other bit positions, they do affect the resulting structure and candidate subnets. In future work, we might consider removing some of these fixed parameters or searching the parameter space with the hope of producing structures that match real subnet boundaries, *e.g.,* ground truth from operators.

Note that our Bayesian Network model captures dependencies *between* segments, and that we did not study dependencies across nybbles *within* segments. We in-

[3] We were unable to identify an expert for one of the networks, even after weeks with assistance from our network operations colleagues.

tend to do so in future research, possibly employing the concept of *mutual information*, or an entropy measure of the string of nybbles within a segment, where the normalization considers the length of that segment.

In this work, we have ignored the temporal characteristics of address sets, treating them as if they are a set of active addresses at one point in time. However, future work would benefit from integrating temporal considerations into our method, for instance with the hope of uncovering boundaries of sequential and random assignments of addresses from dynamic pools that we discovered in some networks [27]. Another consideration for future work is structural analysis in time-series, *e.g.,* to detect changes in network deployments.

Lastly, our evaluation supports our claim that Entropy/IP is effective at generating hit lists of candidate target addresses if one wishes to conduct a survey or census of the IPv6 Internet by active scans. This is an obvious choice for future work that we plan to pursue. Here, note that our tool can be used on datasets much smaller than the large sets we described in Section 3 and employed for evaluation. Thus, datasets known to the research community can be used for further discovery of IPv6 server farms, routing infrastructure, as well as client addresses of selected networks.

7. CONCLUSION

Comprehensive understanding of address structure is more difficult with IPv6 than IPv4. This is due to the features and the freedom that IPv6 offers in address assignment. To accelerate our IPv6 investigations, we developed a system—Entropy/IP—that automates network structure discovery by deeply analyzing sets of sample addresses gleaned by standard means, *e.g.,* server logs, passive DNS, and `traceroute`. We demonstrate the system's effectiveness in discovering structure in both interface identifiers *and* network identifiers, *i.e.,* subnets. While there is future work still to do, our initial performance evaluation suggests there is potential to surgically survey the vast Internet address space, as well as improve analysts' understanding in operation and defense of our increasingly complicated Internet.

Acknowledgements

We thank Keung-Chi "KC" Ng, Eric Vyncke, and the four network operators who replied to our solicitation for comments. We thank Johanna Ullrich for helpful discussion on their methods and datasets. We also express gratitude to our proof-readers: Jan Galkowski, Steve Hoey, Grzegorz Karch, Mariusz Słabicki, and our shepherd, Walter Willinger.

8. REFERENCES

[1] ARIN. Number Resource Policy Manual. https://www.arin.net/policy/nrpm.html#six521.

[2] K. Cho, R. Kaizaki, and A. Kato. Aguri: An Aggregation-Based Traffic Profiler. In *Proceedings of the Workshop on Quality of Future Internet Services (QofIS '01)*, Coimbra, Portugal, September 2001.

[3] J. Czyz, M. Luckie, M. Allman, and M. Bailey. Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy. In *Network and Distributed System Security Symposium*, Feb. 2016.

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

[5] M. Defeche and E. Vyncke. Measuring IPv6 Traffic in BitTorrent Networks. Internet-Draft draft-vyncke-ipv6-traffic-in-p2p-networks-01, 2012.

[6] N. Dojer. Learning Bayesian Networks Does Not Have to Be NP-hard. In *Mathematical Foundations of Computer Science 2006*, pages 305–314. Springer, 2006.

[7] A. W. Drake. *Fundamentals of applied probability theory*. Mcgraw-Hill College, 1967.

[8] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-Wide Scanning and its Security Applications. In *Proceedings of the 22nd USENIX Security Symposium*, Aug. 2013.

[9] C. Estan and G. Magin. Interactive Traffic Analysis and Visualization with Wisconsin Netpy. In *Proceedings of USENIX LISA*, San Diego, CA, December 2005.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Kdd*, volume 96, pages 226–231, 1996.

[11] Farsight Security. DNSDB. https://www.dnsdb.info/.

[12] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical Approaches to DDoS Attack Detection and Response. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 303–314. IEEE, 2003.

[13] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle. Scanning the IPv6 Internet: Towards a Comprehensive Hitlist. In *Traffic Monitoring and Analysis - 8th International Workshop. Proceedings, TMA 2016.*, Louvain La Neuve, Belgium, 2016. IFIP.

[14] F. Gont and T. Chown. Network Reconnaissance in IPv6 Networks. RFC 7707, 2016.

[15] D. Goodin. Using IPv6 with Linux? You've likely been visited by Shodan and other scanners. http://goo.gl/98vLdx, August 2016.

[16] Google. IPv6 Statistics. http://goo.gl/8FWGW9, August 2016.

[17] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291, 2006.

[18] S. Jiang, Q. Sun, I. Farrer, Y. Bo, and T. Yang. Analysis of Semantic Embedded IPv6 Address Schemas. Internet-Draft draft-jiang-semantic-prefix-06, 2013.

[19] E. Kohler, J. Li, V. Paxson, and S. Shenker. Observed Structure of Addresses in IP Traffic. In *Internet Measurement Workshop*, pages 253–266, 2002.

[20] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[21] J. Korhonen, J. Arkko, T. Savolainen, and S. Krishnan. IPv6 for Third Generation Partnership Project (3GPP) Cellular Hosts: Stateless Address Autoconfiguration, RFC 7066. https://goo.gl/EHcx0M, 2013.

[22] B. Krishnamurthy and J. Wang. On Network-Aware Clustering of Web Clients. *ACM SIGCOMM Computer Communication Review*, 30(4):97–110, 2000.

[23] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. In *Proceedings of ACM SIGCOMM '05*, Philadelphia, PA, August 2005.

[24] W. Lee and D. Xiang. Information-Theoretic Measures for Anomaly Detection. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 130–143. IEEE, 2001.

[25] D. Malone. Observations of IPv6 Addresses. In *Passive and Active Network Measurement, 9th International Conference, PAM 2008, Cleveland, OH, USA, April 29-30, 2008. Proceedings*, pages 21–30, 2008.

[26] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941, 2007.

[27] D. Plonka and A. Berger. Temporal and Spatial Classification of Active IPv6 Addresses. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 509–522. ACM, 2015.

[28] L. Rabiner and B. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[29] Rapid7 Labs. Forward DNS Records (ANY). https://scans.io/study/sonar.fdns, Aug. 2016.

[30] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[31] SI6 Networks. SI6 Networks' IPv6 Toolkit. https://goo.gl/MJXyTz, Aug. 2016.

[32] W. T. Strayer, C. E. Jones, F. Tchakountio, and R. R. Hain. SPIE-IPv6: Single IPv6 Packet Traceback. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 118–125. IEEE, 2004.

[33] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl. On Reconnaissance with IPv6: A Pattern-Based Scanning Approach. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pages 186–192. IEEE, 2015.

[34] A. Wagner and B. Plattner. Entropy Based Worm and Anomaly Detection in Fast IP Networks. In *Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on*, pages 172–177. IEEE, 2005.

[35] B. Wilczyński and N. Dojer. BNFinder: Exact and Efficient Method for Learning Bayesian Networks. *Bioinformatics*, 25(2):286–287, 2009.