The Bitcoin Hunter: Detecting Bitcoin Traffic Over Encrypted Channels

Fatemeh Rezaei[1], Shahrzad Naseri[1], Ittay Eyal[2], and Amir Houmansadr[1]

[1] College of Information and Computer Sciences, Amherst
[2] Cornell University

Abstract. Bitcoin and similar blockchain-based currencies are significantly important to consumers and industry because of their applications in electronic commerce and other trust-based distributed systems. Therefore, it is of paramount importance to the consumers and industry to maintain reliable access to their Bitcoin assets. In this paper, we investigate the resilience of Bitcoin to blocking by the powerful network entities such as ISPs and governments. By characterizing Bitcoin's communication patterns, we design classifiers that can distinguish (and therefore block) Bitcoin traffic even if it is tunneled through an encrypted channel like Tor and even if Bitcoin traffic is being mixed with background traffic, e.g., due to browsing websites. We perform extensive experiments to demonstrate the reliability of our classifiers in identifying Bitcoin traffic even despite using obfuscation protocols like Tor pluggable transports. We conclude that standard obfuscation mechanisms are not enough to ensure blocking-resilient access to Bitcoin (and similar cryptocurrencies), therefore cryptocurrency operators should deploy tailored traffic obfuscation mechanisms.

Keywords: Bitcoin. Blockchain . Cryptocurrency.

1 Introduction

Bitcoin and similar blockchain-based currencies [39] have seen rapid adoption by consumers and industry because of their many applications in electronic commerce and other trust-based distributed systems. Bitcoin supports \$1–\$4.2B worth of transactions per day, growing steadily. Bitcoin and similar virtual currencies offer significant advantages compared to traditional electronic currencies, which include open access to a global e-commerce infrastructure, lower transaction fees, cryptographically supported contracts [3] and services [36], and transnational operations.

Given this significant importance of electronic currencies, they need to be resistant to embargoes by governments. That is, people investing in cryptocurrencies (by running businesses that rely on such currencies) should be assured that their Internet providers or governments are not able to prevent them from using their cryptocurrencies if they decide too. For the sake of argument, consider what happens if the Great Firewall of China decides to block all Bitcoin traffic overnight.

In this paper, we investigate the resilience of Bitcoin to blocking by powerful network entities, including ISPs and governments. Note that identifying standard (non-encrypted) Bitcoin traffic is trivial as Bitcoin messages use specific packet contents and formats. Therefore, a trivial countermeasure to prevent an ISP from identifying Bitcoin traffic is to tunnel Bitcoin over an encrypting tool, e.g., VPN, SSH, or Tor. However, previous studies [51, 24, 9] show that encryption is *not enough* to conceal the nature or even the content of communications. Such attacks are broadly known as *traffic analysis*.

In this paper, we investigate if and how Bitcoin's traffic can be identified through traffic analysis despite being tunneled through an encrypted channel. First, we characterize Bitcoin's traffic patterns such as rates, timings, and sizes. Comparing with other protocols, we show that Bitcoin has traffic patterns that are unique, because of the specific types of messages sent by Bitcoin peers. Leveraging such unique features of Bitcoin traffic, we design a toolset of classifiers in order to distinguish Bitcoin traffic over encrypted channels. We perform extensive evaluations of our classifiers by capturing Bitcoin traffic in the wild. Particularly, we use more than two month of Bitcoin and other protocols traffic over Tor [14] and three Tor pluggable transports [45], namely, FTE [16], meek [34], and obfs4 [44] to evaluate our classifiers. Our experiments show that while such obfuscation mechanisms modify Bitcoin's traffic by changing the sizes of packets, and changing packet latencies, they are not able to hide the presence of Bitcoin traffic.

In summary we make the following main contributions:

1. We evaluate Bitcoin's traffic and characterize its patterns such as its packet sizes and traffic shape. We compare Bitcoin's traffic patterns to other popular protocols showing its patterns to be unique.
2. Based on our characterization of Bitcoin traffic, we design a range of classifiers whose goal is to identify Bitcoin traffic despite being tunneled through an encrypted channel (like Tor) and in the presence of background noise (e.g., open browser tabs).
3. Using several months of Bitcoin traffic and other protocols, we perform experiments to evaluate their performance. We evaluate our classifiers when Bitcoin traffic is tunneled over Tor and three Tor pluggable transports of FTE [16], meek [34] and obfs4 [44], and in the presence of background noise. Our classifiers are able to identify Bitcoin traffic in all cases with only 10 minutes of traffic with more than 99% true positive and near 0 false positives.

2 Background on Bitcoin

Bitcoin is the most popular cryptocurrency. It uses a decentralized, peer-to-peer architecture [40], where each peer (e.g., client) is identified by her unique public key. Bitcoin clients exchange money through Bitcoin transactions, which are broadcasted on Bitcoin's p2p network.

To prevent double spending and similar violations, Bitcoin uses a public ledger called the *blockchain* to store all Bitcoin transactions. The blockchain is a chain of *blocks*, where each block contains a set of transactions and a *proof-of-work*. A proof-of-work is a piece of data which is time-consuming and costly to generate. However, verifying the proof-of-work is easy. Each block is valid if and only if all of its transactions and its proof-of-work are valid. A verified block is broadcast on the network to update all peers' local ledgers.

Bitcoin Protocol Messages. Bitcoin communications involve various protocol messages that are created by Bitcoin peers. We divide Bitcoin protocol messages into two classes: *synchronization messages*, which are used for propagating user addresses and transactions in the Bitcoin network, and *block-related messages*.

2.1 Synchronization Messages

These messages are aimed at keeping Bitcoin peers synchronized with the rest of the Bitcoin network.

`addr:`Each peer advertises the information and IP addresses of other peers via `addr` message in the network.

`inventory(inv):` Peers send `inv` to advertise their knowledge about the known objects, like transactions and blocks.

`getdata:` A peer sends `getdata` message in response to the `inv` to retrieve information about the content of an object, which can be a block or a transaction.

`tx:` This message describes a transaction in response to a `getdata` message.

2.2 Block-Related Messages

Such messages are used to exchange Bitcoin blocks among the peers. The current Bitcoin network is supporting two ways of propagating blocks, full block and compact block propagation.

Full Block Propagation: The sender node first validate the block completely, then it advertise the possession of block by `inv` message. The receiving peer which doesn't have the block, asks for it by sending `getdata` message. Finally, the sender node send the block via `block` message. Sending full block in the network is wasting network bandwidth since we are re-sending all of the transaction and nodes have some of transaction in their memory pool. The messages transmitted in this mode is:

`block:` It consist of block version information, previous block hash, merkle root of a Merkle tree collection which is a hash of all transactions related to this block. Sending the new block forwarded through all the network.

Compact Block Propagation: From the middle of 2016 in `0.13.0` version, Bitcoin protocol start to forward blocks as compact blocks which means instead of forwarding full blocks in network, only a sketch of block is sent. The sketch include 80-byte block's header, the short transactions IDs used for matching already-available transactions and select of transactions which sending peer expect that a receiving peer may be missing.

`sendcmpct:` This message informs the receiving peer about the mode of communication the sending peer has chosen (low or high bandwidth).

`cmpctblock:` This message introduced in the compact block relaying and is presenting a sketch of block.

`getblocktxn:` This message is introduced in compact block relaying and is used to request for the transactions that are missed by sending a list of their indexes.

`blocktxn:` This message is introduced in compact-block relaying and is used to provide some of the transactions in a block, as requested.

3 Threat Model

Here, we investigate the resilience of Bitcoin to blocking by powerful entities such as ISP or governments. This is an essential problem because censoring the Bitcoin traffic prevents the use of this currency in censoring regions, ruining the privacy of transactions.

4 Characterizing Bitcoin Traffic

4.1 Proportion and Distribution of Messages

Bitcoin peers generate various kinds of messages such as inv, block, tx, and etc. We show that the distribution and sizes of such messages are quite unique to the Bitcoin protocol, distinguishing Bitcoin traffic reliably from other protocols.

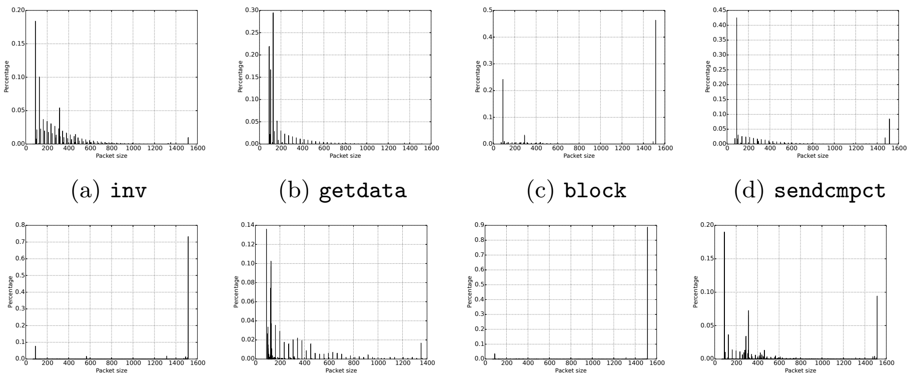
Distribution of Packet Sizes. Figures 1a to 1h illustrate the packet size histogram of different types of Bitcoin messages in our collected Bitcoin traffic. As can be seen, each type of message has a distinguishing traffic pattern. Note that through our experiments, we find out that `tx` and `inv` are dominating with 43.6% and 27.2% of all packets, respectively. Therefore, the characteristics of these messages will shape the pattern of a Bitcoin peer's traffic.

Histogram of packet sizes in aggregate traffic. Figures 2a to 2d show the histogram of packet sizes in the upstream and downstream directions, in compact and full block relaying. As mentioned before, `tx` and `inv` dominate the messages sent by a typical Bitcoin peer, therefore, their sizes (shown in Figures 1a and 1h) strongly shape the histogram of Bitcoin traffic, making them uniquely distinguishable from other protocols.

Comparing to other protocols: Figures 2e to 2l show the histogram of other popular protocols, collected as described in Section 6. Note that we look at the traffic after going through an encryption tunnel, e.g., a VPN or SSH tunnel, so the histogram includes the (small) TCP ACK packets.

As we can see, the packet size distribution of Bitcoin is uniquely different from these other protocols, since a Bitcoin connection is composed of unique messages with specific size distributions shown before.

Above we showed that the counts and sizes of packets in Bitcoin demonstrate a unique behavior. In addition, the shape of traffic in Bitcoin and the volume of traffic received over time is distinguishable from other protocols.



(a) `inv` (b) `getdata` (c) `block` (d) `sendcmpct`

(e) `cmpctblock` (f) `getblocktxn` (g) `blocktxn` (h) `tx`

Fig. 1: Packet size distribution of Bitcoin messages in compact block relaying

Full block relaying mode Figure 3a shows the traffic of a Bitcoin client operating in the full block relaying mode. As can be seen, the small protocol packets, mostly corresponding to `inv` and `tx` messages, appear uniformly over the time. On the other hand, the Bitcoin full blocks appear as large spikes of roughly 1MB at specific points in time, i.e., once a new block is generated in the network.

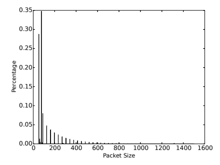
Compact block relaying mode In the compact block relaying mode, it would be harder to notice the block spikes, since only a sketch of the blocks is transmitted. As can be seen, compact blocks appear at smaller amplitudes than the actual block size, but the behavior is also nondeterministic, since it depends on whether the client has previously received some of the transactions in that block. This intuitively makes detection of compact blocks less reliable than full blocks, as shown later in our experiments.

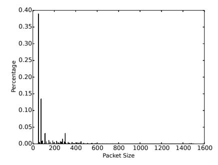
5 Designing Bitcoin Classifiers

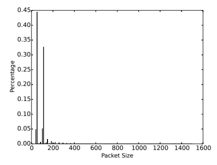
We use the features described above to build robust classifiers for Bitcoin traffic. We aim for our classifiers to work even in the presence of encryption and background traffic, e.g., when the machine running Bitcoin is used for web browsing and runs other applications, or when the Bitcoin traffic is tunneled over Tor.

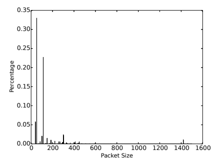
5.1 Size-based Classifier

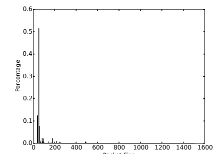
As noted in Section 4.1, the histogram of Bitcoin packet sizes has a unique pattern. Based on this, we designed classifiers to distinguish Bitcoin traffic from other protocols.

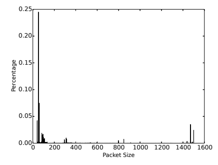

(a) Compact block, upstream

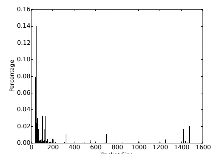

(b) Compact block, downstream

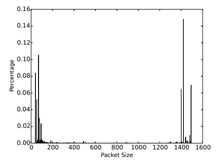

(c) Full Block, upstream

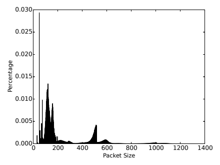

(d) Full block, downstream

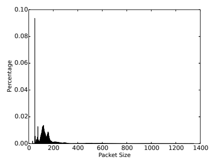

(e) HTTP, upstream

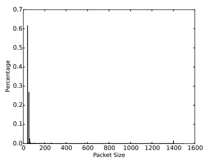

(f) HTTP, downstream


(g) SSH, upstream


(h) SSH, downstream


(i) VoIP, upstream


(j) VoIP, downstream


(k) BitTorrent, upstream

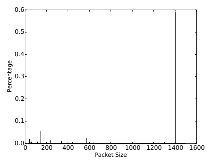
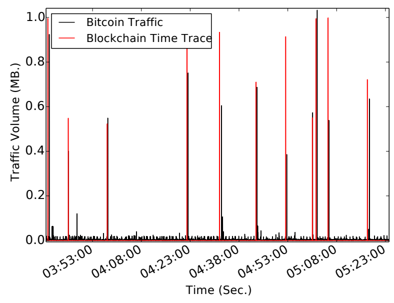
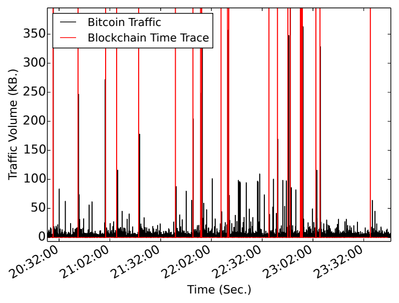

(l) BitTorrent, downstream

Fig. 2: Upstream and downstream packet size distribution of Bitcoin and several popular protocols



(a) Full block



(b) Compact block

Fig. 3: Comparing time of received blocks with time of blocks in the blockchain

Size histogram classifier (`SizeHist`) This classifier correlates the packet size histogram of a target user's traffic with the packet size histogram of Bitcoin traffic captured by the adversary. By histogram of packet sizes we mean number of each packet size from 1 to MTU size. To do so, the classifier first divides the given traffic into upstream and downstream directions, then it calculates the histogram of packet sizes in each direction. The baseline (real-time Bitcoin traffic) is also divided in upstream and downstream directions.

In the next step, the classifier calculates the cosine similarity between the histogram of the target traffic and several Bitcoin traces in both upstream and downstream direction. If both of the upstream and downstream averaged correlation values are above their thresholds, the target traffic is detected as Bitcoin traffic.

Tor-specific classifier (`SizeTor`): Tor [14] reformats traffic into constant-sized segments called *cells*. However, as the size of a cell is smaller than the MTU of IP packets, depending on the volume of traffic, multiple cells can merge into one single packet. This makes the number of single-cell packets and multiple-cell packets different for different protocols tunneled over Tor. Our `SizeTor` classifier aims at detecting Bitcoin traffic tunneled over Tor based on the distribution of single-cell packets. As shown in Appendix 4, Bitcoin traffic consists of a large number of small-size packets (due to frequent `inv` messages). Tor will add padding to these small packets to form cells. Therefore, the ratio of single-size packets in Bitcoin over-Tor is larger than regular traffic over Tor, e.g., HTTP-over-Tor. Based on this, our classifier compares the ratio of single-cell packets to all packets; if the ratio is larger than a threshold, the connection will be flagged as a Bitcoin connection. Note that our `SizeTor` classifier can be adjusted for other protocols that similarly change the size of packets.

5.2 Shape-based Classifier

The main intuition of our shape-based classifier is looking for changes in the traffic volume of a target user around the times of block announcements. Therefore, we assume that the classifier obtains the times and sizes of Bitcoin blocks, e.g., from the public `blockchain.info` repository, or even by running a local Bitcoin client.

For each confirmed Bitcoin block, the algorithm analyzes the volume of the target traffic two time windows with size ω_i around the block time, one before $(t_{block_i} - \omega_i, t_{block_i})$ and one after the block time $(t_{block_i}, t_{block_i} + \omega_i)$. The size of the window depends on the size of the block, the target client's bandwidth, etc., as evaluated later. For an actual Bitcoin traffic with no noise, the difference should be close to the size of the block. If the difference is within the bound (J), the algorithm considers that block to be *detected* in the traffic of the suspected user. The algorithm performs the same for a number of blocks and evaluates the ratio of such "detected" blocks. If the ratio is above a threshold η, the target user is declared to be a Bitcoin client.

Choosing the threshold η. The threshold should be chosen based on the target user's specific network conditions such as background traffic, network noise, and bandwidth.

To do so, the classifier generates N (e.g., $N = 100$) synthetic block series, which we call *ground false*s. The classifier then correlates the target traffic with each of the N ground false instances using the correlation function. Finally, the threshold η is chosen to be larger than the largest correlation value.

Choosing other parameters. The window shape classifier also needs to choose the values of the parameters ω and J. Parameter ω needs to be big enough to contain the most of the traffic of a block during block propagation. Moreover, Parameter J is chosen to take into account that some of the block propagation traffic might not be downloaded in that time window. This parameters needs to be selected based on user's bandwidth and the volume of background traffic, and therefore it needs to be chosen for each client specifically.

5.3 Neural Network-based Classifier (NN-based)

We use a neural network-based classifier to detect Bitcoin in presence of a more complex background noise, e.g., browsing more than one website simultaneously. In following, we explain the feature selection phase, and then describe the design of our neural network.

Feature selection To create each sample data, we divide time into intervals and use the volume of traffic in each interval as our features, which is presented in equation $\mathrm{vlm} = v_1, v_2, ..., v_n$.

Note that v_I is the volume of traffic in interval I. We choose 10 minutes as the *sample size*, which is the smallest length to have at least one peak of traffic. Furthermore, to choose the interval length, we try different values of 1, 5, 10 and 20 seconds. From our experiments, we find out that the interval length of 10 seconds results in the best performance. Therefore, we choose 10 seconds as the interval length (l). Since the length of each sample is 10 minutes (600 seconds), using equation $n = \text{sample size}/l$, we get an array of length 60 as our feature.

Designing the model For our neural network model, we use a combination of convolutional and fully-connected network that consists of an input layer, an output layer, and three hiden layers: one convolutional layer, and two dense layers. The input layer has $n = 60$ number of neurons, which is the size of each sample data, the hidden layers have $n_1 = 64$, $n_2 = 32$ and $n_3 = 16$ number of neurons, respectively, and the output layer has 1 neuron which represents if the sample data contains Bitcoin traffic or not. We use Relu as the activation function of the hidden layers and sigmoid [21] for the output layer. Also, we use binary cross-entropy as our loss function, and Adam optimization [30].

5.4 Combined Classifier

In this classifier, we combine all the attributes used in the above classifiers. More specifically, we are using the size histogram of the packets, downstream to upstream traffic volume ratio, and volume over time. Our final feature set has a length of 1576 (1 for downstream to upstream, 60 for volume over time, and 1515 for size histogram). Note that the feature that we use in sizeTor classifier is a subset of size histogram, and volume over time is capturing the shape of traffic that we utilized in the shape-based classifier. The model that we use has three convolutional layers and one dense layer with sizes: 1024, 128, 64, and 32 for the dense layer. We use the same activation functions and loss function as the NN-based classifier.

6 Experimental Setup

We use Bitcoin Core software[3] to run full node Bitcoin clients on 5 virtual machines on a campus network. Each virtual machine is connected to the Internet with high bandwidth. Before starting the experiments, we leave our Bitcoin clients for a few days to make sure they have downloaded an up-to-date blockchain ledger. The Bitcoin client is passively receiving blocks and participating in the Bitcoin network, but it is not doing any transactions. We capture Bitcoin traffic under three different scenarios on a Linux 16.0.4 virtual machine:

6.1 Datasets

Collecting Bitcoin traffic: We use Bitcoin version 0.12.0 to capture Bitcoin traffic in the full block relaying mode and Bitcoin version 0.14.0 to capture traffic in the compact block relaying mode. We capture Bitcoin traffic for each version for a period of a month.

Bitcoin tunneled through Tor We captured Bitcoin traffic behind Tor [14] for both compact and full block modes. We also captured Bitcoin traffic in the compact block mode behind Tor and popular Tor pluggable transports of obfs4 [52], FTE [16], and Meek-amazon [34].

Bitcoin with background traffic: We captured Bitcoin traffic in presence of HTTP background traffic by browsing the top 500 Alexa websites using the Selenium[4] tool while running Bitcoin software. We also collected Bitcoin traffic with HTTP background for the same set of websites behind Tor and its three pluggable transports using Selenium.

CAIDA background traffic: We use CAIDA's 2018 anonymized traces[5] as a dataset for additional background traffic.

HTTP traffic: We collect top 500 Alexa websites using Selenium tool. Also, we capture these websites over Tor, and three pluggable transports. Moreover, we use the dataset by [41] which has collected the top 50,000 Alexa websites over Tor.

[3] https://bitcoin.org/en/bitcoin-core/
[4] `http://www.seleniumhq.org`
[5] `https://www.caida.org/data/monitors/passive-equinix-nyc.xml`

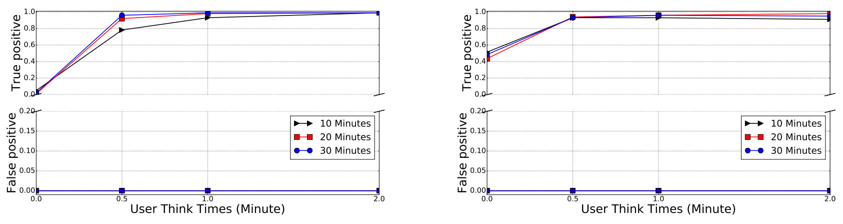
Table 1: Traffic class breakdown for CAIDA dataset

Traffic class	Port numbers	Number of connections	%of total
http, https	80, 8080, 443	745262	0.318
dns	53	1073758	0.457
smtp	25	2646	0.001
telnet	23	6958	0.003
ssh/scp	22	4928	0.002
other	–	511700	0.219
all	–	2345252	1.0

6.2 Metrics

We use following metrics to measure the performance of our classifiers:

- **True positive:** True positive shows the proportion of the data which contain Bitcoin, and our model correctly identified as Bitcoin traffic.
- **False positive:** False positive shows the proportion of the data which did not contain Bitcoin traffic, and our model incorrectly classified as Bitcoin.
- **Accuracy:** Accuracy shows the proportion of the data which was correctly classified.



(a) Compact Mode (b) Full Mode

Fig. 4: Result of `sizeHist` classifier on noisy Bitcoin traffic

6.3 Modeling Normal Users

In this section, we describe four different types of users' profile that we use to evaluate the performance of our Bitcoin classifiers against.

- Simple User: A simple user is a Bitcoin client with no background noise and traffic. This type of user does not generate any network traffic except the Bitcoin client application traffic. Therefore, all traffic of the simple user is Bitcoin traffic.
- Simple Noisy User: A simple noise user is a Bitcoin client who browses only **one** webpage.To control the background traffic, we introduce a parameter

named think time, T, representing the amount of time that the user spends on a particular website.

– Complex Web (Complicated/Sophisticated) User: A complex user is a Bitcoin client who browses **multiple** websites simultaneously.
– Complex CAIDA User: A complex CAIDA user is a Bitcoin client who is running 1-5 number of CAIDA applications, which is introduced in table 2 simultaneously in the background.

7 Results

In this section, we implement our classifiers to evaluate their performance on user profiles described in part 6.3 writing more than a thousands lines of code in Python. First, for each classifier, we declare the user profile(s) that we use for evaluation of its performance and the false data that we use for computing its false positive. Second, we describe the result of each classifier and give a summary and comparison of them at the end of this section.

7.1 User Profiles and False Data

For each classifier, we use a specific user profile and depending on that we choose the false data. As we explained above, false data is the base traffic that we use to compute false positive. Note that, we use same length of traffic for Bitcoin and the false data. In other words, it is the traffic that we compare our Bitcoin traffic with. For example, when we have 10 minutes of HTTP traffic, we are continuously browsing different websites for 10 minutes. We make these samples by concatenating the browsing of different websites. In the following, we describe these pairs for each classifier(s).

– For shape-based classifier, we use the simple user profile. Furthermore, we use HTTP which is the typical user behavior as the false data. This experiment evaluates if Bitcoin traffic can be differentiated from browsing an HTTP website.
– For the rest of the binary classifiers in this section, we use simple noisy user profile and attempt to detect the presence of Bitcoin. Note that, similar to the window-based classifier, we use HTTP for the false data. This experiment attempts to evaluate if browsing an HTTP website is enough to hide the Bitcoin traffic.
– For the neural network-based and combined classifiers, we use the complex web and complex CAIDA user for training and testing.

Note that, for these two classifiers, we evaluate our model using $10,000$ number of test data, and report the false positive, true positive and accuracy. For rest of the classifiers, we use 500 number of test data for evaluation. Also, the data is balanced, which means we have the same number of data for each category.

7.2 Size-based Classifiers

`SizeHist` Classifier We implement the `sizeHist` classifier on Bitcoin traffic in compact and full block modes using the noisy user model described in Section 6.3. Figure 4 shows the performance of this classifier. We control the noise using think time (T). Increasing T decreases the noise and enhances the classifier's performance. Figure 4 shows that we can reach more than 90% true positive and 0% false positive for both modes when we have 10 minutes of traffic and set T to 2 minutes. It is worth stating that we could reach similar results when we set T to 0.5 minutes and have 20 minutes of traffic.

7.3 Shape-based Classifier

To implement this classifier, we set J and ω introduced in Section 5.2 to 100 kilobytes and 20 seconds, respectively. To set η, we compute block detection rate for Bitcoin using ground false shown in Figure 5. We need to set η to be larger than all the detection rate values for Bitcoin using ground false. Note that each point for Bitcoin using ground false in the figure is the average for 25 different ground falses. Using this figure, we set η to 0.4. Moreover, Figure 5 shows the block detection rate for HTTP using ground truth and block detection rate for Bitcoin using ground truth too. Using the η, we can detect all Bitcoin traffic through (August 28 - October 5) as Bitcoin. Also, we did not classify any of the HTTP traffic as Bitcoin, which results in 0% false positive. Furthermore, the performance of shape-based classifier quickly diminishes in the presence of a small HTTP background noise. Also, we fail to detect Bitcoin traffic in compact mode because of small block sizes, which makes it impossible to distinguish them.

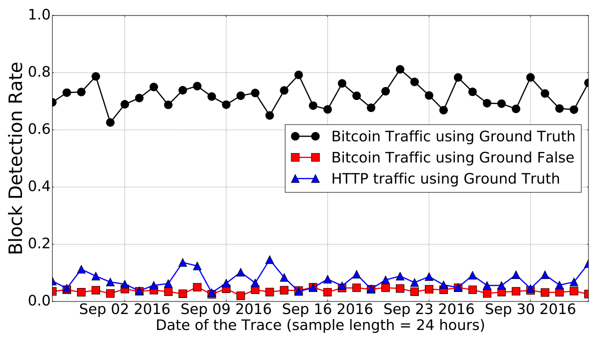


Fig. 5: Block detection rate using shape-based classifier

7.4 Neural Network-based Classifier

We implement the NN-based classifier using Keras [12] with Tensorflow [1] backend. We use complex web and complex CAIDA user to evaluate its performance.

Table 2 shows the result of NN-based classifier for different sizes of training data. As the table indicates, the accuracy of the classifier improves from 62% to 96% when we increase the size of training data from 1000 to 40, 000. Note that, true and false positive improves from 44% to 92% and 20% to 2% respectively when we increase the size of training data.

Table 2: Result of neural network classifier.

Training size	False positive (%)	True positive (%)	Accuracy (%)
1000	20	44	62
5000	11	80	85
10, 000	6	84	89
40, 000	2	92	95

7.5 Combined Classifier

To extend the neural network-based classifier, we defined combined classifier, which uses all of the attributes used in previous classifiers. Using this classifier, we reach 99.84% accuracy with false positive of 0 and true positive of 99.74% having 40, 000 of training data and sample size of 10 minutes. This result is very promising and shows that having enough data and using the attributes that distinguishes Bitcoin from other traffic, we are able to train a neural network model that gives us 0% false positive and more than 99% accuracy.

7.6 Summary and Comparison of the Results

The SizeHist and shape-based work only when there is a very small background noise or no noise (think time of 2 minutes). Therefore, they are not useful when Bitcoin traffic has a large amount of background noise. To distinguish Bitcoin traffic in the presence of larger noises, we employ NN-based and combined classifiers. The benefit of these classifiers is that they do not have the training phase required in NN-based techniques.

On the other hand, the NN-based classifier and combined classifiers result in a better performance in the presence of higher background noise (complex web and complex CAIDA). The combined classifier outperforms the NN-based by using more features during training. This classifier detects Bitcoin traffic using the complex web and complex CAIDA user explained in 6.3 with 0% false positive and much higher accuracy (99.84%).

8 Countermeasures

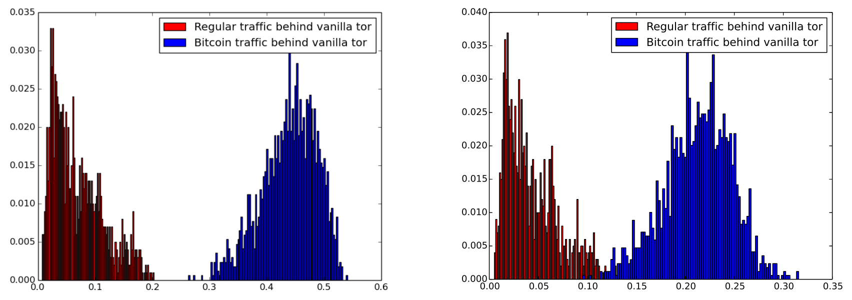
A possible countermeasure against Bitcoin detection is to tunnel Bitcoin traffic over an anonymity system like Tor. We tunnel Bitcoin traffic over Tor to disguise its patterns. We evaluate the invisibility that Tor provides by designing a new classifier (SizeTor) which is tailored for Tor. Also, we use our strongest classifiers (NN-based and combined) to evaluate if Tor succeeds in hiding Bitcoin traffic.

8.1 Bitcoin Over Tor

Tor sends traffic in *cells*. If the packets sizes is below the cell size, it will add padding to the packets to reach the fixed cell sizes. This modifies the traffic patterns of Bitcoin traffic, e.g., its packet sizes, therefore it may increase resistance to traffic classification. In the following we evaluate our classifiers against Tor and three state-of-the-art Tor pluggable transports [45] namely *obfs4*, *FTE*, and *Meek* explained in the following.

Pluggable Transports

- **Obfs4:** obfs4 is a widely used Tor pluggable transport, which is based on *ScrambleSuit* [50]. It differs with ScrambleSuit in public key obfuscation and its protocol for one-way authentication, which makes it faster.
- **FTE** The FTE transport re-encrypts Tor packets in order to match the regular expressions of a benign protocol like HTTP.
- **meek** Meek uses *domain fronting* [20] to tunnel traffic through public CDN or cloud platforms.



(a) upstream (b) downstream

Fig. 6: Histogram of one cell packet ratio for HTTP traffic and Bitcoin traffic

Bitcoin traffic has a larger ratio of one-cell packets compared to Tor. This is due to a large number of small Bitcoin messages (e.g., `inv`) that are each put into a single-cell packets. Figures 6a and 6b show the histogram of one cell size packets ratio in the upstream and downstream directions.

8.2 Evaluating Bitcoin over Tor

`SizeTor` classifier We implement `SizeTor` classifier on noisy Bitcoin on compact mode for Tor and its three pluggable transports. As it is displayed in Figure 7, we could detect Bitcoin traffic with high accuracy for the complex user model when the background noise is one open tab. As Figure 7 indicates, having a traffic size of 10 minutes is enough to detect Bitcoin traffic with around 90% true positive and 0% of false positive. Moreover, the Figure states that the result of classifier quickly diminishes when we increase the background noise from one to 2 or 3 open tabs. Note that this figure shows the average result of this classifier on Tor and three pluggable transports.

Neural Network-based classifier Table 3 presents the result of NN-based classifier for the complex web user over Tor for $10,000$ numbers of test data. As the table suggests, performance of our classifier improves when we increase the size of training data. More specifically, our false positives enhances from 11% to 4% when we increase the training data from 1000 to $40,000$. Moreover, when using $40,000$ training data, we reach 99% and 4% true and false positive, respectively (97% accuracy). Note that the reason we get better results from this classifier on Tor dataset is that this dataset only contains the complex web user since we did not have CAIDA application over Tor to use for our evaluation.

Table 3: Result of neural network classifier on Tor dataset.

Training size	False positive (%)	True positive (%)	Accuracy (%)
1000	11	98	93
5000	6	98	96
$10,000$	3	98	97
$40,000$	4	99	97

Combined classifier over Tor We apply the combined classifier on our Tor dataset, which consists of Bitcoin traffic with up to 5 number of open tabs on the background. Our experiments show that having $40,000$ of training data is enough to achieve more than 99.72% accuracy with 0.04% false positive and 99.5% true positive. Our combined classifier outperforms the previous classifiers (SizeTor and NN-based) on the Tor dataset as well. This is because we are using a more complex model and a more significant number of features.

9 Related Work

In this section, we overview previous work on classifying different protocols and discuss previous attacks on Bitcoin cryptocurrency.

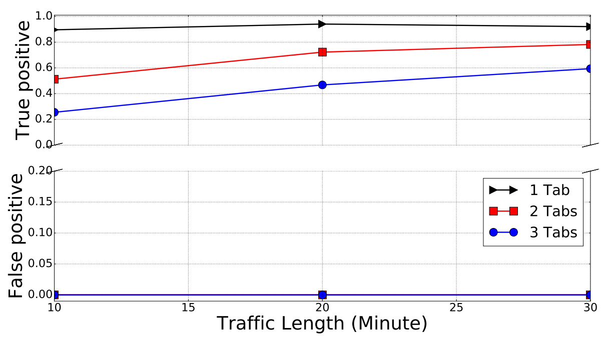


Fig. 7: Detecting Bitcoin compact mode traffic behind Tor (Tor and its three pluggable transports: meek, obfs, fte) using SizeTor

9.1 Protocol Classification

There is extensive work in literature trying to classify different applications or protocols in the network. Previously, researchers were focused on classification according to the port numbers [26, 25, 32, 49] and payload [49, 11, 22, 37]. Since many applications use uncertain port numbers [25], or some encrypt their payloads, researchers adopted new methods for protocol classification. Recent techniques use statistics such as packet sizes and timings for classification [6, 18, 27, 13, 17, 10, 53].

In some of the studies, researchers apply machine learning techniques on these statistics to classify different applications. For example, in [17], authors take a semi-supervised approach to classify a variety of applications such as FTP, HTTP, P2P.In [10, 53], authors use SVM classifier to distinguish different applications such as WWW, Mail, and FTP. Also, in [53], authors use SVM to classify a broad application category such as mail, buck traffic, service. In [38], Andrew W. Moore et al. use two refinements on the Bayesian technique and show that they can reach 95% of accuracy on Internet traffic classification. Moreover, in [5], authors use the Bayesian neural network to classify Internet traffic using just header-derived statistics. In [31], Wei Li et al. use C4.5 [46] decision tree to classify Internet traffic such as mail, gaming, database, and browsing. They reach an accuracy of 99.8, using 12 features collected at the start of the flow. In [42, 29], authors survey the papers on Internet traffic classification using machine learning techniques.

9.2 Attacks on Bitcoin Cryptocurrency

In this section, we discuss the previous attacks on the Bitcoin network. In [4], the authors discuss routing attacks, their impact, and possible countermeasures. They study partitioning and delay attacks by investigating node-level and network-wide attacks for both.

In [23], authors design eclipse attack in which the attacker isolate a victim from its peers by monopolizing its all incoming and outgoing connections, and filters her view of the network. Doing so, the attacker wastes her computing power on an outdated view of the network.

In [47, 2, 43, 35, 48], authors use Bitcoin transaction patterns to link users (or link transactions) using some side information.

In [7], the authors propose a technique to link the public key of a user to her address or link her transactions. They show that they could launch their attacks even when the clients are behind NAT using only a few number of machines.

In [28], they analyze the security of using Bitcoin for fast payments and show that the current Bitcoin system is not secure unless they integrate Bitcoin network with some detection mechanism. Also, they study double spending attacks on fast payments and implement a method to prevent it. In [19], the authors introduce selfish-mine in which a pool can obtain a revenue larger than its share of mining power by forcing the honest miners to work on the wrong block, and therefore, waste their computing power.

In [33], the authors propose a new attack on the Bitcoin payment system that exploits some authentication vulnerability, or some weakness on the refund procedure. They suggest a revision on the Bitcoin payment protocol, which prevents both attacks.

10 Conclusions

The reliable access to Bitcoin and similar cryptocurrencies is of crucial importance due to their consumers and the related industry. In this paper, we investigated the resilience of Bitcoin to blocking by a powerful network entity such as an ISP or a government. By characterizing Bitcoin's communication patterns, we designed various classifiers that could distinguish (and therefore block) Bitcoin traffic even if it is tunneled over an encrypted channel like Tor, and even when it is mixed with background traffic. Through extensive experiments on network traffic, we demonstrated that our classifiers could reliably identify Bitcoin traffic despite using obfuscation protocols like Tor pluggable transports that modify traffic patterns. In order to disguise such patterns, an obfuscating protocol needs to apply significant cover traffic or employ large perturbations, which is undesirable for typical clients.

References

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, and e. a. Michael Isard. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
2. E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 34–51, 2013.

3. M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure multiparty computations on bitcoin. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 443–458, Washington, DC, USA, 2014. IEEE Computer Society.
4. M. Apostolaki, A. Zohar, and L. Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 375–392, 2017.
5. T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Trans. Neural Networks*, 18(1):223–239, 2007.
6. R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty. Realtime classification for encrypted traffic. In P. Festa, editor, *Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings*, volume 6049 of *Lecture Notes in Computer Science*, pages 373–385. Springer, 2010.
7. A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin P2P network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 15–29, 2014.
8. A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.
9. X. Cai, X. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *CCS*, 2012.
10. J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang. An accurate traffic classification model based on support vector machines. *Int. Journal of Network Management*, 27(1), 2017.
11. T. Choi, C. Kim, J. Yoon, J. Park, B. Lee, H. Kim, H. Chung, and T. Jeong. Content-aware internet application traffic measurement and analysis. In *Managing Next Generation Convergence Networks and Services, IEEE/IFIP Network Operations and Management Symposium, NOMS 2004, Seoul, Korea, 19-23 April 2004, Proceedings*, pages 511–524, 2004.
12. F. Chollet. keras. `https://github.com/fchollet/keras`, 2015.
13. M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic classification through simple statistical fingerprinting. *Computer Communication Review*, 37(1):5–16, 2007.
14. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-generation Onion Router. In *USENIX Security*, 2004.
15. R. Durrett. *Probability: theory and examples.* Cambridge university press, 2010.
16. K. Dyer, S. Coull, T. Ristenpart, and T. Shrimpton. Protocol Misidentification Made Easy with Format-transforming Encryption. In *CCS*, 2013.
17. J. Erman, A. Mahanti, M. F. Arlitt, I. Cohen, and C. L. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Perform. Eval.*, 64(9-12):1194–1213, 2007.
18. J. Erman, A. Mahanti, M. F. Arlitt, and C. L. Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 883–892, 2007.
19. I. Eyal and E. G. Sirer. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, 2018.
20. D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson. Blocking-resistant Communication through Domain Fronting. In *PETS*, 2015.

21. I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.
22. P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: automated construction of application signatures. In *Proceedings of the 1st Annual ACM Workshop on Mining Network Data, MineNet 2005, Philadelphia, Pennsylvania, USA, August 26, 2005*, pages 197–202, 2005.
23. E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, Washington, D.C., 2015. USENIX Association.
24. D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Cloud Computing Security*, 2009.
25. T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos. Is P2P dying or just hiding? [P2P traffic measurement]. In *Proceedings of the Global Telecommunications Conference, 2004. GLOBECOM '04, Dallas, Texas, USA, 29 November - 3 December 2004*, pages 1532–1538, 2004.
26. T. Karagiannis, A. Broido, M. Faloutsos, and K. C. Claffy. Transport layer identification of P2P traffic. In *Proceedings of the 4th ACM SIGCOMM Internet Measurement Conference, IMC 2004, Taormina, Sicily, Italy, October 25-27, 2004*, pages 121–134, 2004.
27. T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Philadelphia, Pennsylvania, USA, August 22-26, 2005*, pages 229–240, 2005.
28. G. Karame, E. Androulaki, and S. Capkun. Double-spending fast payments in bitcoin. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 906–917, 2012.
29. H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the 2008 ACM Conference on Emerging Network Experiment and Technology, CoNEXT 2008, Madrid, Spain, December 9-12, 2008*, page 11, 2008.
30. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
31. W. Li and A. W. Moore. A machine learning approach for efficient traffic classification. In *15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2007), October 24-26, 2007, Istanbul, Turkey*, pages 310–317, 2007.
32. A. Madhukar and C. L. Williamson. A longitudinal study of P2P traffic classification. In *14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006), 11-14 September 2006, Monterey, California, USA*, pages 179–188, 2006.
33. P. McCorry, S. F. Shahandashti, and F. Hao. Refund attacks on bitcoin's payment protocol. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, pages 581–599, 2016.
34. meek Pluggable Transport. `https://trac.torproject.org/projects/tor/wiki/doc/meek`.
35. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. *Commun. ACM*, 59(4):86–93, 2016.

36. A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 475–490, Washington, DC, USA, 2014. IEEE Computer Society.
37. A. W. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In *Passive and Active Network Measurement, 6th International Workshop, PAM 2005, Boston, MA, USA, March 31 - April 1, 2005, Proceedings*, pages 41–54, 2005.
38. A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2005, June 6-10, 2005, Banff, Alberta, Canada*, pages 50–60, 2005.
39. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`, 2008.
40. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
41. M. Nasr, A. Bahramali, and A. Houmansadr. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1962–1976, 2018.
42. T. T. T. Nguyen and G. J. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials*, 10(1-4):56–76, 2008.
43. M. Ober, S. Katzenbeisser, and K. Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future Internet*, 5(2):237–250, 2013.
44. A Simple Obfuscating Proxy. `https://www.torproject.org/projects/obfsproxy.html.en`.
45. Tor: Pluggable Transports. `https://www.torproject.org/docs/pluggable-transports.html.en`.
46. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
47. F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011*, pages 1318–1326, 2011.
48. D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 6–24, 2013.
49. S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 512–521, 2004.
50. P. Winter, T. Pulls, and J. Fuss. Scramblesuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 213–224. ACM, 2013.
51. C. Wright, L. Ballard, F. Monrose, and G. Masson. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In *USENIX Security*, 2007.
52. Yawning. Obfsproxy4. `https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt`, 2015.

53. R. Yuan, Z. Li, X. Guan, and L. Xu. An svm-based machine learning method for accurate internet traffic classification. *Information Systems Frontiers*, 12(2):149–156, 2010.