Reference:	OPC-M/TECH.B/61
Date:	13 June 2011
Copy no:	

A potential technique to deanonymise users of the TOR network

[redacted]
OPC-MCR, GCHQ

Summary
A new technique to deanonymise users of the TOR network is demonstrated. It is shown that the majority of a small truthed set of data can be deanonymised without any false hits anywhere in eight bearer-hours of data.
For this algorithm to be further tested we must run some TOR exit nodes and collect data from these. SIGINT packet logging of guard node traffic is also required.
Demonstration software is available in the form of an R package from [redacted]

Copy	Distribution
1	NSA R1 [redacted]
2	NSA R4 [redacted]
3	NSA CES [redacted]
4	CSE [redacted]
5	DSD [redacted]
6	SEQUOIA [redacted]
7	ICTR [redacted]
8	JTRIG [redacted]
9	TEA [redacted]
10	GTE [redacted]
11	OPC-HQ [redacted]
12	OPC-ALTO [redacted]
13	OPC-CNE [redacted]
14	OPC-MCR [redacted]

OPC-M/TECH.B/61
[18 pages]

THIS PAGE IS INTENTIONALLY LEFT BLANK

OPC-M/TECH.B/61

A potential technique to deanonymise users of the TOR network

[redacted]
OPC-MCR, GCHQ

13 June 2011

1 Introduction

The Onion Router (TOR) [1] is used by individuals and organisations that want to hide the originating IP address of their communications.

A TOR client chooses the circuit their traffic will follow through a set of TOR routers before contacting the destination server. The first hop after the client is to the subset of the routers designated as "guard" nodes. The final hop in the TOR network is from a subset of routers which choose to be an "exit" node. There can be any number of TOR routers between the guard and exit nodes but typical clients choose to have one router.

There are many features that mean it is hard to track traffic through the TOR network:

- Traffic is encrypted in multiple layers between the client node and each TOR router (leading to the "Onion" analogy). Hence data between each TOR router has different ciphertexts. Unencrypted traffic is only seen between the exit node and destination server.
- TOR splits all traffic up into standard size "cells". Therefore packet sizes can not be used to follow traffic.
- Each connection between TOR routers will typically multiplex many circuits' traffic effectively masking any particular user's traffic.
- To ensure fairness of service TOR has a per-circuit rate-limiting store-and-forward buffer in each router. This buffer tends to flatten timing features in traffic. [3]

The aim of this work is to find the client IP address associated with unencrypted traffic between an exit node and a destination server. This paper achieves this aim given the following constraints:

- We must own the exit node. This constraint means that we can demultiplex traffic by TOR circuit and thus get a cleaner signal. See section 5 for more detail.

- We must be able to log the packet times of the traffic from the guard node to the client.
- The TOR communication must be long and structured. We will demonstrate the technique against a single user browsing the web via TOR.
- The client must not be running a TOR router of their own – otherwise we can not separate the client's own traffic from other TOR traffic.

The attack we will present is based on correlating exit node and guard node traffic and does not require tracking communications through any intermediate link in the TOR network. This approach should help maximise the chance of a successful attack despite incomplete SIGINT coverage.

2 Test data

We are going to work with two sets of data:

1. Truthed data where we have matched guard node and exit node traffic for a single user web browsing.
2. Bulk traffic logs for guard nodes from four SIGINT bearers.

We will try to develop techniques that can correctly match up the truth data but not false alarm anywhere in the bulk SIGINT logs.

2.1 Truthed data

TOR is designed to make it hard to link client and exit node traffic together. In conjunction with ICTR-NE and JTRIG we came up with a way to collect the exit node traffic from our own web browsing. As illustrated in figure 1 we used a virtual private server (VPS) as an intermediate destination for our traffic. We then run packet loggers on our client machine and the VPS.

We run an open HTTP proxy server on the VPS. We want to run an open proxy server to ensure that there is minimal impact on the data flows (which, for example, authentication may introduce) whilst still being able to conduct representative web browsing. A risk of an open proxy server is that other internet users may use it thus potentially leading to unlawful traffic interception. To avoid this danger we changed our "User-Agent" string to a non-standard value and the proxy server was configured to only respond to this user agent. Furthermore the proxy server was only active for the brief duration of the experiment. This setup was approved by JTRIG. No traffic due to other users was detected in the packet logs.

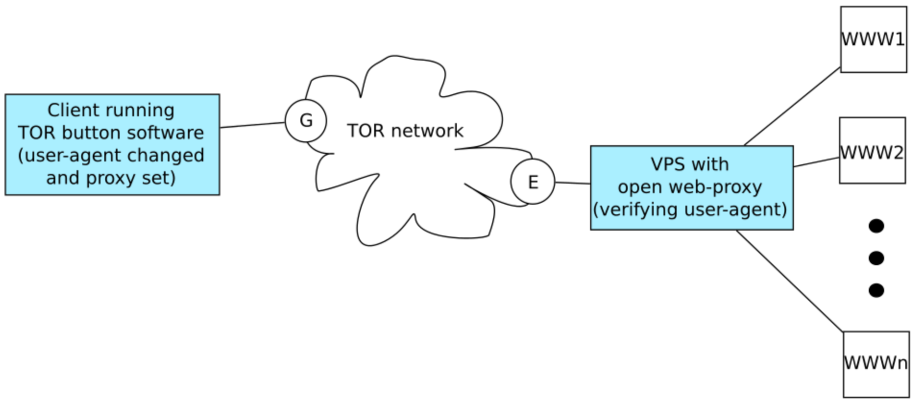
Client running TOR button software (user-agent changed and proxy set)
G
TOR network
E
VPS with open web-proxy (verifying user-agent)
WWW1
WWW2
WWWn

Figure 1: Our test data collection infrastructure. We control the two shaded hosts and run packet logging on these hosts. We connect to the public TOR network (guard node "G" and exit node "E" are shown) using default settings of the TOR button. We then browse public web sites.

On our client we used the standard "TOR button" web-browser extension to access TOR with two minor modifications. Firstly, we changed the internal TOR button proxy (polipo) to use our open HTTP proxy after transiting TOR. Secondly, we changed the web-browser User-Agent to match that accepted by our open HTTP proxy.

Four experiments of about ten minutes each were conducted:

1. "News": Search on Bing for news, followed by browsing BBC News and then the Washington Post.

2. "TOR": browsing the TOR website and then using a privacy checking website.

3. "Download": visit to SlashDot followed by downloading a large PDF file.

4. "Forum": Search on Google followed by browsing a PC technical help forum.

Packet logs were collected for each of these. In each experiment one guard node and one exit node was used apart from in experiment 2 where TOR set up a new path during the experiment with a change in both guard node and exit node. Therefore we will generally split experiment 2 into the two circuits "2a" and "2b".

2.2 Bulk data

To prove a low false alarm rate of any algorithm we need bulk data. We used IP address dumps from four internet bearers (two were "client-server" bearers and two were "server-client" bearers). The data was kindly collected by GTE with their high-speed data recorder. The timestamps are microsecond accurate but the data comes with no packet size, protocol or port information. Each capture was two hours long totalling eight hours of collection.

We identify TOR traffic by use of "consensus logs" [4] which are imported by ICTR-NE. The TOR network decides what nodes are part of the network and what their roles are. We filter the SIGINT packet traces to packets that satisfy:

- One IP address being a guard node as identified in the consensus log closest to the data collection period, and
- The other IP address not being seen any the consensus log from a month surrounding that period.

This approach will inaccurately identify traffic between clients and guard nodes; the inaccuracy comes from the fact that the guard nodes may be doing other non-TOR communication. We therefore believe that we will approximately filter down to a superset of the wanted traffic.

We note that these filtering rules mean that we will ignore any clients that also run a TOR node. However this constraint will still apply for a potential operational scenario where we also run guard nodes; without full knowledge of the client one can not tell whether a circuit is terminating in the client or being relayed through its TOR node.

3 Why we are not tracking a circuit through the TOR network

When we started this work, we considered attacks that tried to follow data through the TOR network. However an initial experiment based on JTRIG browsing logs showed that SIGINT visibility was too low for a significant chance of success.

We will now describe the experiment conducted but the rest of the section is not required reading.

We compare two sources of data:

- Communication between TOR servers seen in SIGINT. TDSD deployed an ICTR signature (in Squeal) to detect communication between TOR nodes across the SIGINT fleet for two days in December 2010. To remove signature false hits we filter down to hits where both IP addresses are known to be TOR routers (by consensus logs [4] from the same period).
- TOR links used by JTRIG for a similar period. JTRIG log the usage of TOR connections. Each of their circuits typically have two links between TOR routers.

There were 1958 TOR routers in the relevant consensus files of which 1893 had signature hits. We consider links between TOR routers as undirected and an all pairs calculations allows us to estimate that there are 1.79 to 1.92 million possible links in the TOR network (we do not know whether we do not see a router due to it being off or invisible to us).

However we only see 6185 links between TOR routers in the SIGINT logs. Therefore we are seeing about 0.3% of possible TOR links in SIGINT. Note this percentage is likely to be an small underestimate of the visibility of TOR links as most TOR circuits will consist of two links within the TOR network: one link involving a guard node and one link involving an exit node. Based on the consensus logs there are 1.65 million links that satisfy this constraint: this link count raises visibility to 0.4%.

The JTRIG logs use 8294 links between TOR nodes of which we see 13. Therefore we could only see 0.16% of JTRIG used TOR links in SIGINT.

The JTRIG link visibility percentage and the population link visibility percentage suggest a significantly different underlying distribution (p-value between 0.005 and 0.01). TOR is claimed to use a server's geolocation to choose circuits, perhaps this circuit choosing algorithm hinders our visibility.

If we assume the visibility of each link in a 2-path within the TOR network is independent then the chance of seeing any particular 2-path is between 1 in 100,000 and 1 in 400,000. In the JTRIG logs of 2935 paths we would expect to observe both links of one or more paths with probability 0.7% to 3%. We saw one so either we were lucky or link visibility is correlated in a path.

To complete a circuit trace you would also need to see the client-to-guard-node traffic and the exit-node-to-server traffic which would lower the probability of visibility yet further. We believe such probabilities are too low for a useful attack and thus we do not consider circuit tracing attacks further.

4 Correlating guard and exit node traffic

We will use the correlation in the timing of data packets between guard and exit nodes to deanonymise TOR. There are two features that we need to remember about TOR:

- TOR repacketises the data which means that packets and packet sizes can not be exactly followed through the network
- Each TOR node has a rate-limiting store-and-forward methodology which will affect packet timings

These two factors means that a burst of packets entering the TOR network will be smeared out over time into cells that do not directly map to the input packets and comparing histograms of guard and exit traffic is hard [3].

Our main insight is to consider cumulative distributions rather than histograms. In particular we will consider cumulative packet count and cumulative TCP payload bytes. We hope that over a long time window that there will be approximate conservation of these quantities and the impact of rate limiting will be less significant (rate limiting will just impose a maximum gradient).

In figure 2 and figure 3 we show cumulative packet counts and cumulative TCP payload sizes for our truth data. It can be seen that a lot of features are preserved – it seems sensible to consider that comparison of these cumulative plots between guard and exit node traffic could lead to a deanonymisation technique.

Considering figure 2 in more detail one can see steps in these plots replicated between guard node and exit node traffic for all the plots apart from perhaps "CtoS 4" – perhaps client-to-server traffic is harder to deanonymise. You can also see that there are changes in scale (for example in experiment 3 there are more guard node packets than exit node packets).

Figure 3 of cumulative TCP payloads shows some similar features. All the server-to-client graphs show preserved steps. However the client-to-server graphs do not show very clear relationships. In particular looking at "CtoS 3", you can see a big difference during the download of the large PDF file. The exit node is sending TCP ACKs with no payload back to the server but the client is sending larger TOR acknowledgement cells back which leads to a large difference in graph shape.

Therefore we will consider deanonymisation based on cumulative packet counts. As well as looking like a generally cleaner picture than using TCP payload, such an approach requires less data to be collected by the SIGINT system. This easier data collection is exemplified by our bulk data logs – these logs do not contain packet sizes, let alone TCP payload sizes.

5 Why we want to own the exit node

SIGINT gives us two added complications that the above truth data analysis above has ignored:

1. Each exit node is serving many clients which will add unwanted data into the traffic for a single user. TOR users are encouraged to run privacy-preserving proxies (e.g. privoxy or polipo) and these proxies try to normalise the traffic so that everyone shares the same HTTP header format. The impact of this is that SIGINT proxy demultiplexing techniques can not reliably be used. It is potentially possible to try and demultiplex traffic via content analysis (e.g. looking at what links are included in webpages) but such a technique would be complex and fragile to HTML formatting errors and require all web browsing to be non-encrypted.

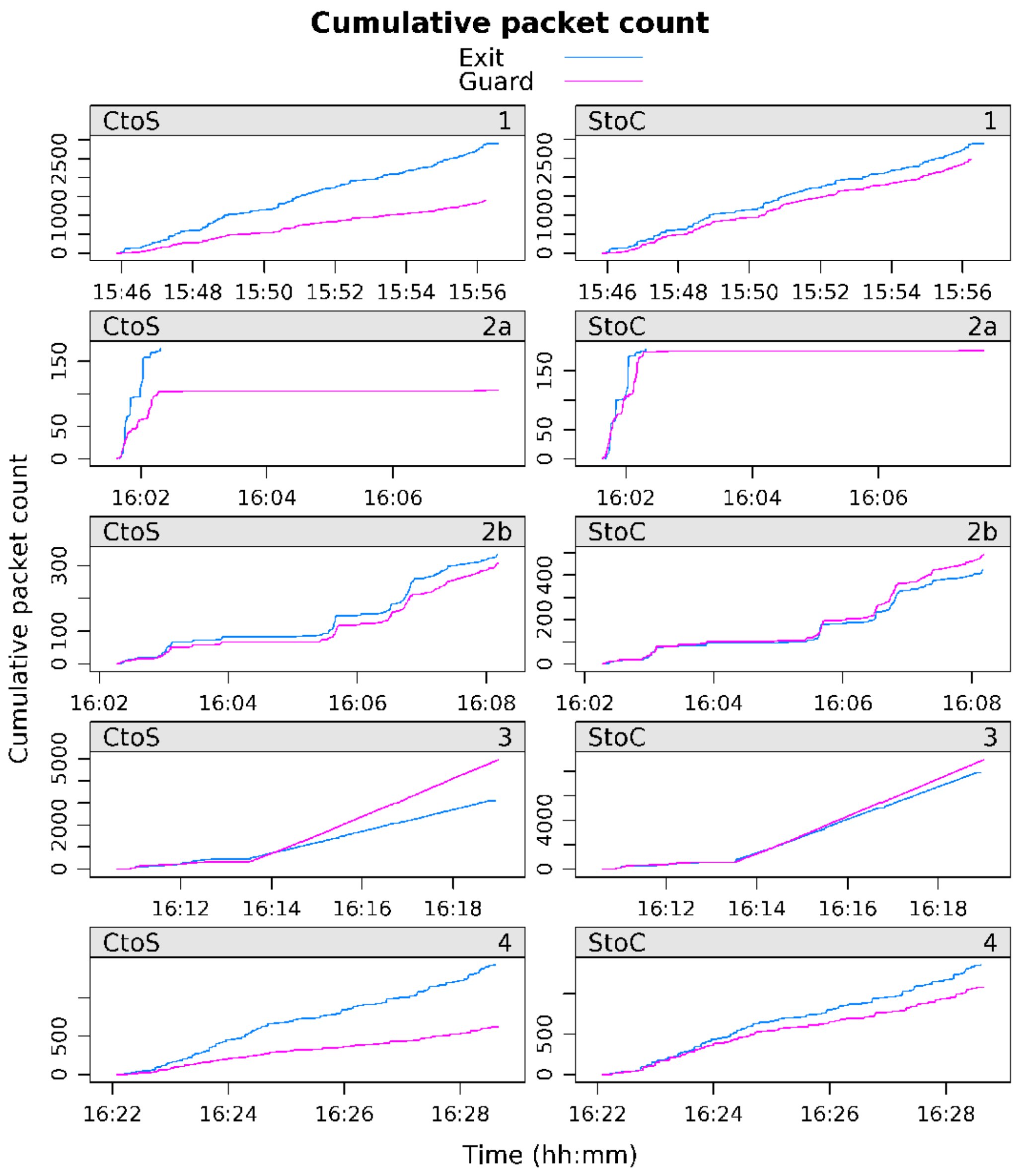

Figure 2: Cumulative packet count for the four truthed sessions. On the left-hand side the client-to-server traffic is shown. On the right-hand side the server-to-client traffic is shown. The second truthed session is split into the two significant circuits used.

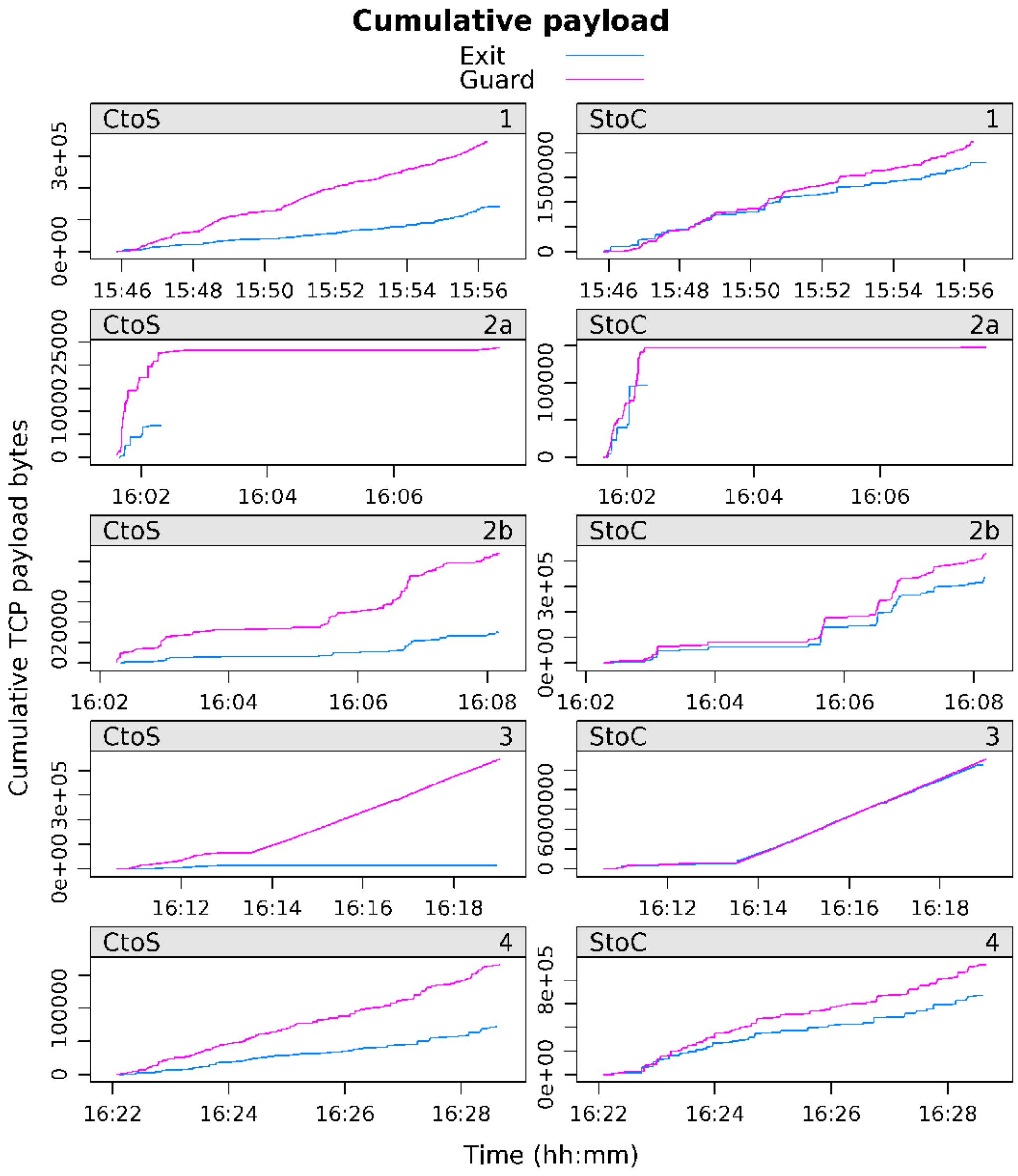

Figure 3: Cumulative payload bytes for the four truthed sessions. On the left-hand side the client-to-server traffic is shown. On the right-hand side the server-to-client traffic is shown. The second truthed session is split into the two significant circuits used.

2. Limited SIGINT coverage will mean we will only see a subset of the exit node traffic. The exit node could be contacting servers anywhere on the internet and unless we have collection very close to the exit node we are likely to only see a subset of these connections.

A potential approach around these problems is to demultiplex by HTTP server. We hope that (in a time window) there might only be one user browsing a particular website so demultiplexing by server will therefore demultiplex by user too. We also hope for complete collection of some of these streams as packets will hopefully follow a small number of routes between the exit node and the HTTP server.

We show the influence of demultiplexing by HTTP server in figure 4. It can be seen that the clear picture in the previous section is no longer present; a modern webpage is typically made up of content from many different HTTP servers. By eye it can be seen that it is hard to link many of these fragments of traffic per HTTP server to the guard node traffic. Indeed using the mathematical technique we will describe in the following section only four of the server-to-client HTTP traces could be successfully linked to guard node traffic.

We therefore recommend that we analyse traffic from exit nodes that we control. Such an approach has several desirable features:

- We can demultiplex exit node traffic by client as the TOR exit node can associate all traffic with a circuit. Complex (and potentially inaccurate) demultiplexing algorithms are not required.
- We collect all exit node traffic for each client circuit (as opposed to SIGINT which may only give fragments of the traffic). We therefore have the cleanest possible data.
- We hope to deanonymise all traffic for a circuit rather than just fragments of the traffic which may not contain the data of interest to an analyst.
- A resultant database of TOR traffic would give a simple place to experiment and deploy TOR deanonymisation analytics.

The NSA have already demonstrated the demultiplexing of traffic in a TOR exit node [2].

6 Score

We compare an exit trace and a guard node trace using basic linear regression (building on a suggestion of [redacted]). We bin time up into 1 second bins ($i = 1 \ldots T$);

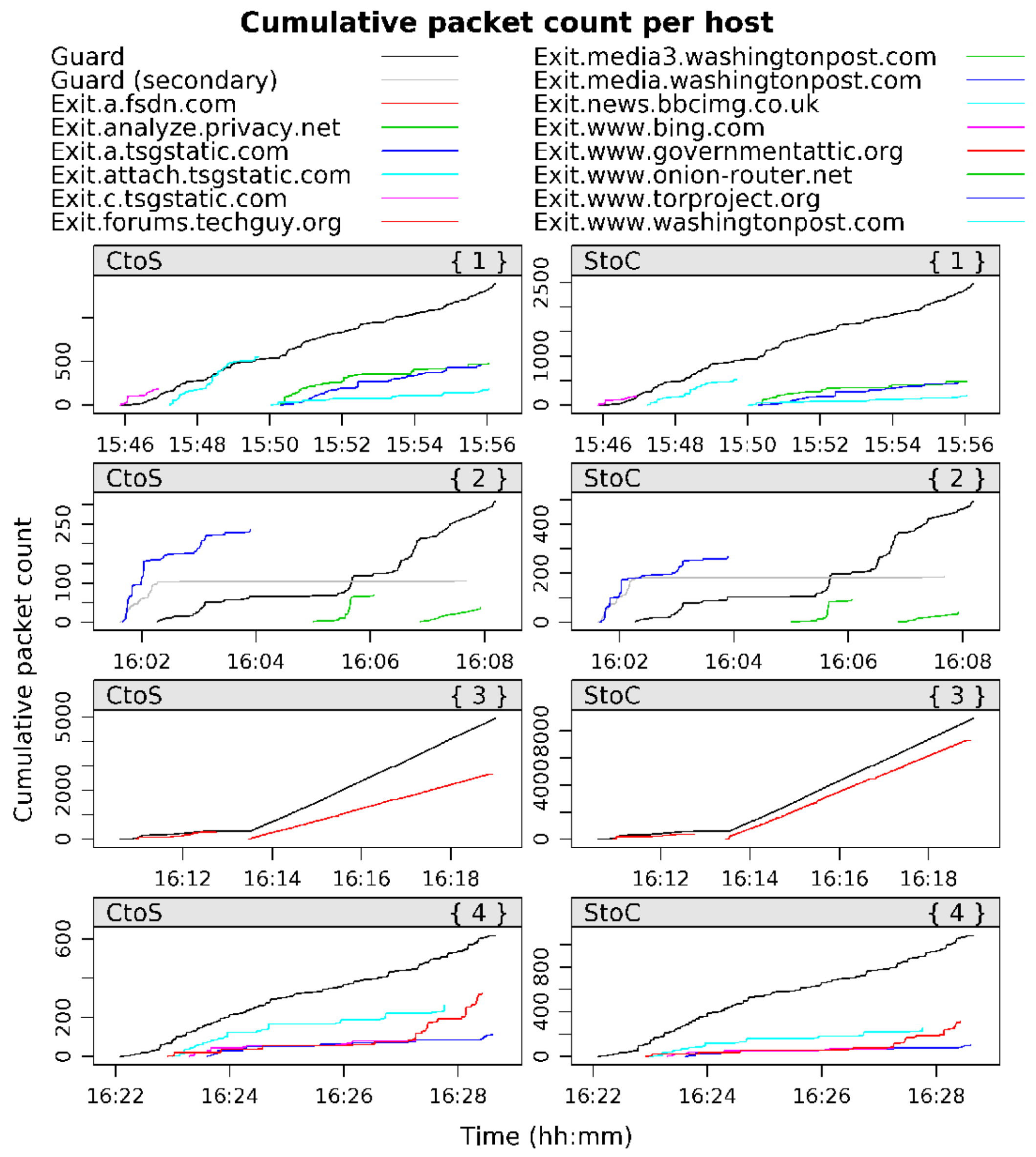

Figure 4: Cumulative packet count for the four truthed sessions but with exit traffic split across the main hosts contacted. On the left-hand side the client-to-server traffic is shown. On the right-hand side the server-to-client traffic is shown.

this choice is slightly arbitrary but does match the timing precision reported by most SIGINT systems. In each bin we then have a pair (E_i, G_i) where E_i is the cumulative exit node packet count at time i and G_i is the cumulative guard node packet count at time i.

Our primary measure of association is the correlation between E_i and G_i as measured by the Pearson product-moment correlation coefficient r. We expect that when the two traffic streams are associated that a very high correlation coefficient will result. We note that high correlation coefficients are generally expected as we're comparing two increasing sequences.

To remove false positives we also fit the following basic linear model:

$$G_i = \alpha + \beta E_i \tag{1}$$

We then use β to discard bad matches. We expect that a similar number of packets are expected in both traces – we only consider traces where $1/2 < \beta < 2$.

A key consideration is how to handle guard node and exit node traces that do not precisely overlap. We are assuming that our exit node trace is the complete log of a TOR circuit. On the other hand, the guard node trace may contain traffic for other TOR circuits. We therefore truncate the guard node trace to the period of the exit node trace. We also pad the guard node packet counts with zero if required to span the exit node trace.

In addition we consider score all possible time shifts of the guard node trace against the exit node trace – i.e. trying out different timing offsets. We do this as we do not know the clock offset in our truth data set and to be as generous as possible in finding false positives in the bulk logs. It is not known whether such a sliding approach may prove useful in practice with known clock offsets; it is possible that such an approach may also combat unknown delays in the TOR network.

7 Results

We first show results of the proposed technique on our truth data set and then consider false positive rates.

As mentioned above we will slide the putative guard node and exit nodes traces against each other to allow for an unknown clock offset between the client and HTTP proxy in the truth data set and to give us the maximum number of comparisons when trying to assess the false positive rate.

7.1 True positives in the truth data

The first experiments are to understand the behaviour of the algorithm on the truth data. In figure 5 we show plots of r^2 and β as the time offset is varied. It can be seen

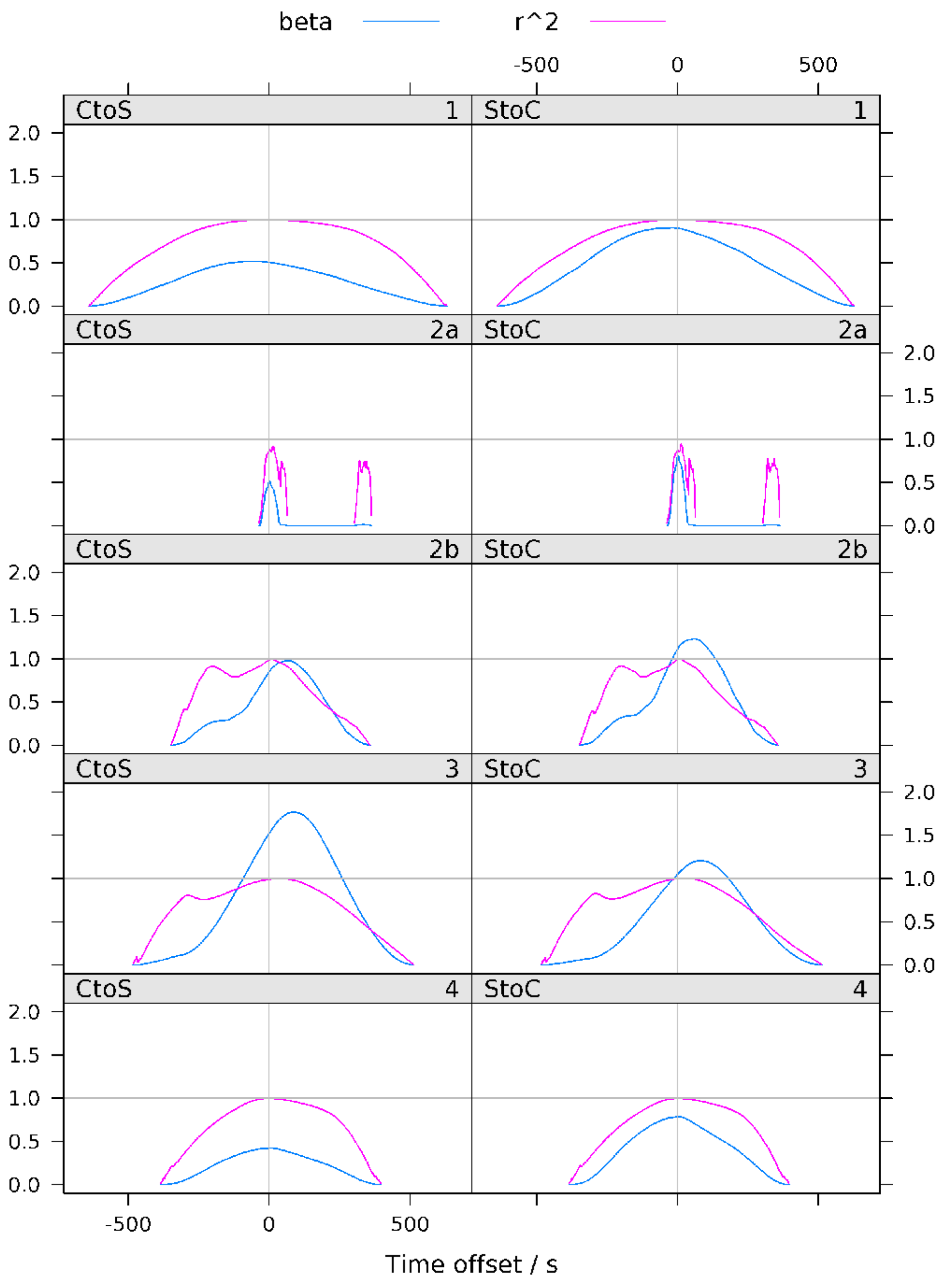

Figure 5: The correlation r^2 and the linear regression coefficient β as the time offset between the guard node and exit node traffic is varied.

Sample	Client-to-server	Server-to-client
1	0.9977	0.9991
2a	0.9238	0.9464
2b	0.9957	0.9983
3	0.9952	0.9998
4	0.9975	0.9983

Table 1: The maximum values of r^2 achieved on the truth data

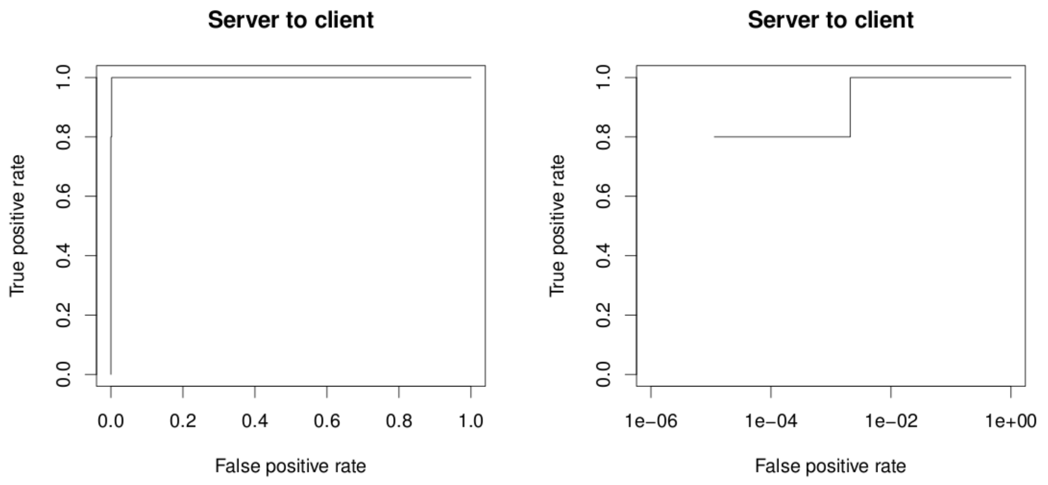

Figure 6: The ROC curve for the server-to-client direction (false-positive rate shown on a linear axis in the left plot and on a logarithmic axis in the right plot).

that correlations close to 1 are achieved at time offsets close to 0. The short circuit "2a" fails to achieve a high correlation. Sensible values for β also result, in particular in the server-to-client direction. The maximum value of the correlation is also shown in table 1 and it can be seen that higher correlations are achieved in the server-to-client direction.

7.2 False positives in the bulk data

We now compare our truth exit traces to the bulk data set and see if we find any false positives. As previously mentioned we're sliding time to allow as many traces as possible to be included.

In the server-to-client direction, we can choose a threshold of 0.998 such that we only miss one true positive whilst having no false positives. The one circuit we miss is "2a" where the session is short (the main activity is only over 33 seconds). The ROC curve looks very good, figure 6.

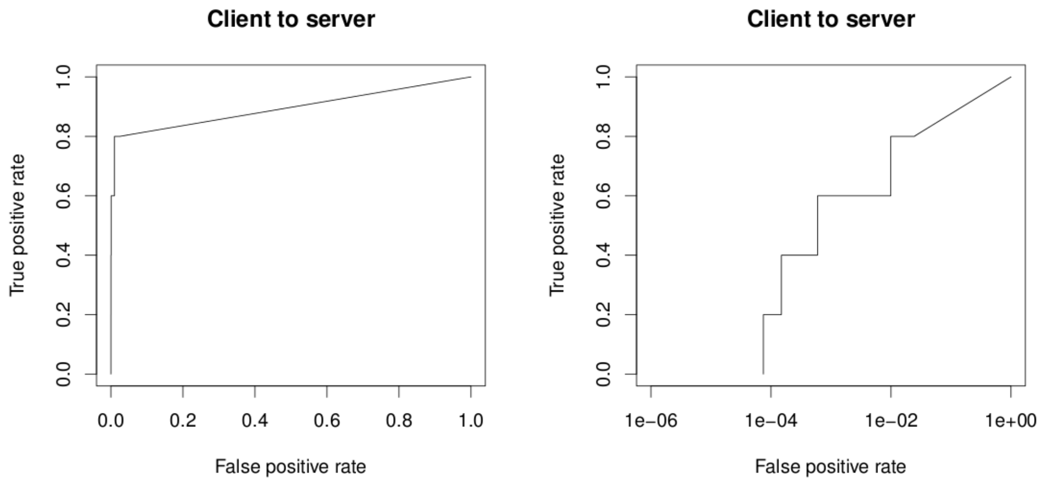

Figure 7: The ROC curve for the client-to-server direction (false-positive rate shown on a linear axis in the left plot and on a logarithmic axis in the right plot).

In the client-to-server direction, the true and false cases are not so well separated, see figure 7. We are not able to detect any true positives without some false hits. This poor separation is probably due to there being less distinctive structure in the client-to-server direction; HTTP requests share a similar size.

In the server-to-client direction a threshold of 0.998 finds the true cases (except "2a") without any false hits. We can try to extrapolate what this false hit rate means to SIGINT collection. It is likely that one would restrict the time slide; we would imagine a maximum of 5 seconds may be sensible as opposed to the 2 hours in each bearer we have allowed. If we imagine restacking our 5 circuits × 4 bearers × 120 minutes of collection into 288 circuits × 100 bearers × 10 seconds you could imagine it is possible we could run 288 test circuits against an aperture of 100 bearers without false hitting when allowing ourselves to slide the time window by 5 seconds. With our current test data we can't promise a lower false positive rate and further experimentation with sustained collection would be required to better evaluate a false positive rate.

8 Software

We provide a reference implementation of the algorithm as an R package "flowcompare". Three functions are provided:

formatflowdata Formats data to that used internally in the algorithm.

scoreflowpair Scores a pair of flows at all time offsets or at a limited range of time offsets.

findbestmatchingflow Find the guard flow that best matches an exit flow.

We also include our truth data as a data frame called tortruth. This package is available from [redacted]

9 Conclusion

We have a shown a technique that can deanonymise TOR web-browsing given packet times between the client and guard node and packet times from the exit node filtered to a single circuit. The false positive rate looks low enough to suggest this technique should be carried forward.

The required data is not collected at present. For this technique to work the following additional data feeds will be required:

- Second-accurate packet logging at TOR exit nodes we control with packets labelled by a unique circuit identifier.
- Second-accurate packet logging of sessions between TOR clients and TOR guard nodes. This data could be obtained by SIGINT or by running guard nodes. The SIGINT solution would require an up-to-date feed of TOR “consensus” documents; TOR IP addresses could then be extracted from the “consensus” documents for filtering by the SIGINT system.

At the time of writing JTRIG are investigating the collection of the exit node data and ICTR-FSP are trialling a feed of guard node data from research bearers.

Wider testing is recommended to better characterise the false positive rate. A first test case (both because we have truth data and for InfoSec purposes) may be to try and deanonymise JTRIG TOR usage.

References

[1] Various TOR resources linked off [redacted]

[2] NSA TOR capability reported by [redacted]

[3] [redacted] Options for further research into the Tor network. B/7694BA/5001/3/104. 2 February 2010. DISCOVER 6722909.

[4] [redacted]

THIS PAGE IS INTENTIONALLY LEFT BLANK