Kirisurf: A Novel Censorship-Resistant Onion-Routing System (pre-print)

Yuhao Dong
University of Waterloo

November 15, 2014

Abstract

Nowadays, authoritarian governments such as that of the People's Republic of China increasingly use deep packet inspection (DPI) and application-layer firewalls to censor and monitor the Internet, utilizing advanced heuristics to detect protocols used by popular censorship circumvention and privacy protection software. More alarmingly, evidence suggest that filters only constitute a portion of the arsenal used by state-level censors. Large-scale surveillance of network traffic, aggregate analysis, and deployment of active and proactive attacks such as protocol probes produce extremely effective identification of circumvention and privacy tools.

In this paper, we present an experimental onion routing system, Kirisurf, with a novel obfuscation system, node graph layout and server distribution system. Kirisurf is designed from the ground up to counter large-scale, state-level censors and eavesdroppers. We discuss several attacks upon Tor and related systems, which are mitigated by Kirisurf's defenses. Finally, we discuss possible attacks upon Kirisurf and possible further improvements.

1 Introduction

1.1 Necessity of a new onion router

Onion routers, such as the famous Tor (The Onion Router) software, use repeated relaying and layered encryption to protect anonymity of users. This technique exposes a proxy-like interface to users, while making sure that no single server knows who the user is, and only one server (the exit server) knows what the user is looking at. A virtual circuit, or simply a circuit, refers to the path taken by the data through the nodes. The impact of compromise or foul play is then greatly limited. Tor, and certain experimental spinoffs like Tor-on-DTLS, has the following characteristics:

- Fully public node directory. Clients randomly pick nodes in the directory with which circuits are built.
- Nodes are divided into entry nodes, intermediate nodes, and exit nodes. As their name suggests, intermediate nodes are limited to the middle of a circuit, while entry nodes can be in the beginning, and exit nodes at the end.
- Standard communication protocols like TLS are used between nodes.

In the threat model of Tor-type systems, gigantic Internet-controlling entities are generally assumed to be rare, and it is assumed that most nodes are in areas where Internet controls are scarce. Additionally, knowledge that a certain user is using Tor is assumed to be public, and will not result in negative repurcussions for the user. Thus the huge node directory, on which Tor servers can be identified (and thus Tor users by the ISP), and distinctive Tor TLS certificates and traffic patterns, are not considered problems. Traffic analysis, which requires mass eavesdropping at the ISP scale, is also not considered part of Tor's threat model.

However, such an optimistic threat model seems increasingly unrealistic. Recent revelations suggest that numerous countries, including democracies such as the US and UK, perform semi-targeted surveillance based on data mining. More and more people in countries like China have demand for free access to overseas websites as self-censorship of domestic websites worsens. Additionally, the fact that Tor and related systems are very difficult to use in places like China pushes users to much less private solutions like Freegate, Ultrasurf, or subscription VPN services, giving users a false sense of security while giving adversaries a free honeypot to eavesdrop on.

In the face of this trend, it is extremely necessary to introduce a new onion-routing system that provides better assurances of privacy and security. Mass and indiscriminate censorship and/or surveillance is an especially insidious threat: it does little to protect against determined criminals, but ordinary citizens' rights of information and privacy are greatly violated.

1.2 New adversary models

Instead of the Tor adversary, which is limited in power and has no access to Internet-wide surveillance [7], we must consider adversary models which are far more powerful and far more unscrupulous. Since different adversaries have differing abilities and goals, we present the following ability categories, based upon the categorization by Houmansandr et al [12]. The objective of an onion router is to defeat adversaries as far down the list as possible.

- **LO**: A *local adversary*. Controls a tiny LAN or slightly more. Schools and offices are examples.

- **OB**: *State/ISP-level oblivious adversary*. State-level adversary. Limited resources, has traffic classification capabilities but cannot gather data from across country and use statistical methods such as traffic volume analysis.
- **OM**: *State/ISP-level omniscient adversary*. State-level adversary. Completely controls network connections within, to, and from a certain country. Every transmission on every public line in the country is visible to the adversary, there are resources for expensive offline analysis of captured traffic, and data collected from different locations can be aggregated.
- **GC**: *Great ISP Conspiracy*. Similar to OM, except on a global level. All WAN transmissions on the planet are visible to the adversary. Large fraction of planet's computing power can be devoted to attacks. Currently nonexistent.

As an example, China is a classic OM adversary actively using its powers to aggressively censor the Internet. China regularly uses advanced and expensive techniques such as traffic analysis and active probing of destinations marked suspicious by the analysis to block Tor [1, 4].

2 Obfuscation

2.1 General issues

2.1.1 Hiding traffic in plain sight

It is often desirable to hide the fact that you are using an onion router. For example, a whistleblower sending a secret live audio stream over the Internet would not want to be identified as using a whistleblower-protecting tool. Dissidents, or simply citizens wishing for access to censored information, in repressive states would not appreciate standing out from the crowd due to their unmistakably onion-routing Internet traffic. At the very least the traffic could be blocked, at the worst the traffic might invite a visit from the secret police.

Obfuscation is a way to accomplish this hiding. Traffic is recoverably encoded into a form similar to innocuous protocols in *imitative obfuscation*, attempting to fool firewalls into miscategorizing the traffic. In *direct obfuscation* on the other hand, the traffic is encoded into a form close to random noise. This attempts to fail firewalls' attempts to classify the traffic as any type.

Some famous imitative obfuscation systems include SkypeMorph, which mimics Skype traffic [9], and StegoTorus, which mimics HTTP traffic [11], both designed for Tor. Imitative obfuscation typically has high overhead, and as the imitation is never perfect, can instead invite classification as, say, "Fake Skype", exposing the fact that the "Skype" traffic in fact hides an onion router or some similarly sensitive payload [12]. In fact, these imitations can easily be blocked by a LO adversary [12], making them nearly useless. We consider this type of obfuscation completely infeasible.

Direct obfuscation involves using cryptographic methods to scramble the body. Essentially, direct obfuscation protocols are designed in a similar way to secure tunnels such as TLS, except all messages are carefully designed to be indistinguishable from random by a passive attacker. Tor has two direct obfuscation protocols in use, obfs2 and obfs3. obfs2 is easily broken by passive attackers, since it transmits the shared symmetric key in the initial handshake [6]. obfs3 is based upon a modified Diffie-Hellman handshake and is secure against passive attackers [5].

2.1.2 What traffic to obfuscate?

Tor only obfuscates selected "bridges" used by people in areas where Tor is blocked [13, 7]. This works for people using Tor for censorship circumvention, and assuming no OM adversary in areas where Tor is not blocked, privacy protection is still good. However, only obfuscating bridge connections will not work if we assume that "silent OM" adversaries exist even in parts where Tor usage is unrestricted. This is because with onion routing connections clearly identified, an OM adversary can correlate traffic entering and exiting the network. For example, the adversary observes large amounts of leaked documents traveling from an exit node via e-mail to, say, a media company. If at the same time, traffic with almost exactly the same timing, speed, and volume patterns is observed entering an entry node from a Tor user at 12345 Unnamed Street, the adversary can guess with reasonable accuracy that their whistleblower is now at that address, even though the latter traffic is encrypted.

Thus we must obfuscate node-to-node traffic as well to prevent an OM adversary from correlating traffic in different parts of the network. More importantly, obfuscation schemes such as obfs3 which preserve data lengths, timing, and even TCP write sizes [5] are clearly out of the window in our threat model. Our new obfuscation scheme for Kirisurf must be able to generate artificial overhead to obscure these parameters, while not introducing new identifying signatures.

2.2 Attacks against obfuscation

2.2.1 Packet length analysis

The packet length analysis statistics attack relies on the fact that certain communications protocols, such as that of Tor, produce IP packets of a very distinctive distribution. For example, the segment length of Tor is fixed at 512 bytes. This means that the vast majority of packets will be 512 bytes long, plus some fixed amount of packet overhead [8].

This type of attack is not mitigated by minimal-overhead systems like obfs3. Although the contents of each packet would be scrambled, the lengths alone will mark a stream as extremely suspicious. Active probes may not even need to be deployed; very few systems other than Tor consistently send 512-byte-long messages.

Many other published obfuscation schemes, such as ScrambleSuit and SkypeMorph, have reasonable defenses against packet length analysis and related attacks on other parameters such as the time passed between successive packets.

2.2.2 Time-volume analysis

A time-volume analysis attack is essentially an attack based on analyzing network flow rate at different times. Time-volume analysis has a different objective than packet length analysis: rather than trying to ascertain the protocol being encapsulated within the obfuscation (say, Tor), it attempts to discover the meaning of the data being transmitted (say, leaked files). Most data have a distinctive traffic pattern when transmitted due to network conditions around the transmitting server. In fact, it has been shown that using naive Bayesian classification, it is possible to discriminate, with extremely high accuracy, between flows from different websites without looking at the content [10].

These attacks become far more powerful when combined with honeypots. As an example, suppose an OM adversary wishes to capture a political whistleblower in its country. It could set up a server, masquerading as a place to retrieve incriminating corruption evidence. However, the server will transmit the data in a distinct pattern, such as "transmit for 1 second, pause for 3 seconds, transmit for 5 seconds, pause for 7 seconds, etc". The OM adversary, being all-powerful within the country, can then zero in on the one user who is receiving data in that pattern. Of course, with a GC adversary, the attack is even more powerful: the user could be in a different country, and the attack will still work.

Unlike packet length analysis, time-volume analysis is less covered in the literature. Dyer et al. [10] seems to be the only paper on the subject; it gives devastating statistical attacks using a classifier algorithm called VNG++, defeating all published obfuscation systems as of the time of writing. To the best of our knowledge, honeypot-based time-volume analysis is novel.

2.3 Kirisurf's defenses

Kirisurf's obfuscation is divided into two layers. The lower layer defines a transport layer for the higher layer, which defines a transport protocol for the encapsulated protocol.

2.3.1 Lower-level obfuscation

Kirisurf's obfuscation scheme, especially the handshake and packet length scrambling, is greatly influenced by that of ScrambleSuit, although with some tweaks. It is a direct obfuscation method, with data streams looking indistinguishable from random noise.

When one Kirisurf node wishes to communicate with another one, it first uses the following handshake to establish a temporary secret key:

- Initiator sends 512 bytes of random data, and then the HMAC of that data with a preshared secret key. (How this preshared secret key is distributed comes later!)
- If HMAC correct, responder responds with ephemeral 2048-bit UniformDH public key.
- Initiator gives its UniformDH public key.
- Both sides compute the Diffie-Hellman shared secret, and derive keys for both directions. RC4 stream cipher states are set up on both sides, and the first 8192 bytes of the keystream are dropped.
- Both sides then send the following message, encrypted with the stream cipher state:
  - 2 bytes: random 16-bit length ℓ
  - ℓ bytes: NUL bytes

The idea of using a preshared secret key comes from [8] and ensures that active probes without knowledge of that key cannot know that the server operates a Kirisurf node. When the HMAC is incorrect, the server stays silent instead of continuing the handshake, giving no incriminating evidence. In addition, recently used hashes are cached to detect and stop replay attacks.

After the handshake is complete, subsequently the bidirectional stream of data is simply encrypted with the RC4 state with no authentication, unlike ScrambleSuit. This is because Kirisurf handles securely authenticated encryption at the circuit level.

2.3.2 Higher-level obfuscation

At a higher level, the obfuscation has essentially two types of messages, data segments and junk segments [18]. Data segments carry the obfuscated payload, while junk segments are used defeat packet statistics attacks and volume analysis attacks.

The first part of Kirisurf's obfuscation is inspired by ScrambleSuit, and defends against packet length analysis. When the connection is established, a random probability distribution of 16 to 1400 bytes/packet is drawn on each side. This is represented by a large array of buckets. Whenever one side wishes to send, say, n bytes to the other end, the following procedure is followed:

- If $n > 1400$, simply send over a data segment of length n. This means the packet is part of a bulk transfer, and bulk transfers in the vast majority of cases involve packets of the MTU size, which is generally a bit over 1400. Tampering with these packets would ironically produce a distinctive feature (lack of MTU-sized packets).
- Otherwise, draw a length m from the probability distribution.
- If $n < m$, then send a data segment of length n with the data, and then immediately send a junk segment of length $m-n$. Both segments are sent in the same TCP write.
- If $n \geq m$, then divide the data segment into a segment of length m containing the first m bytes, and a segment of length $n-m$ containing the rest of the data. Send these two segments seperately, in different TCP writes.
- "Juggle" the probability distribution by randomly increasing some of the numbers in the buckets and decreasing numbers in other buckets.

We will now describe Kirisurf's second set of defenses, which we propose as a defense against time-volume analysis. It aims to obscure "coarse features" such as traffic graph, "burstiness", total bytes transmitted, and time of last transmission, which can accurately fingerprint websites even in non-honeypot situations using Bayesian techniques similar to those of the extremely powerful VNG++ [10]. Whenever one side wishes to send n bytes, in addition to the abovementioned obfuscation, the following is done:

- Draw a number i according to an exponential distribution with mean 3.
- Draw a number n from the probability distribution, but then multiply n by 4. (This is because most TCP *writes*, instead of packets, are around 4 kilobytes long)
- Schedule a junk message filled with n bytes of junk to be sent i seconds later.
- The junk packet will not be sent if the line is "too busy"; more precisely the junk will not be sent if another packet is in front of it in the send queue i seconds later.

The combined effect of this is to greatly "smear" the traffic graph horizontally, obscuring patterns. For seconds-level tasks such as loading a website, this produces large amounts of garbage that increase the overhead to 100% or even more. However, for bulk tasks like transferring a large file, the overhead is decreased since the last rule stipulates that junk packets are avoided on busy lines.

Thus, although the overhead may be very large for common, latency-bound tasks such as e-mail and browsing, in situations where overhead is most disruptive to users, it is avoided.

2.3.3 Multiplexing of encapsulated connections

A multiplexing protocol is run over the obfuscation [18]; this means that all connections between two endpoints are in fact mixed into one stream before obfuscation. This further increases the difficulty of traffic analysis, as without cracking the obfuscation it is difficult to isolate the stream the adversary wants to monitor (say, a honeypot) from unimportant streams (such as casual browsing) happening at the same time.

2.4 Testing results

We use a simulated honeypot. The honeypot does the following, pausing 3 seconds between each step:

1. Send random data at maximum speed for 1 second.
2. Send 5 16-kilobyte blocks of random data, pausing 1 second between each block.
3. Sending 25KB of data, 512 bytes at a time, pausing 0.2 seconds between each 512 bytes.
4. Send data at maximum speed for 1 second, pause 1 second, send data for 2 seconds, pause 2 seconds, send data for 3 seconds, pause 3 seconds.

Step 1 attempts to form a high area in the traffic graph for as close to exactly 1 second as possible. Step 2 attempts to form exactly 5 prominent spikes in the traffic. Step 3 tests how well packet length patterns are preserved: without obfuscation it should produce a stream of packets a bit more than 512 bytes long each. The final one is designed to see how well the obfuscation performs when subject to differing burst lengths.

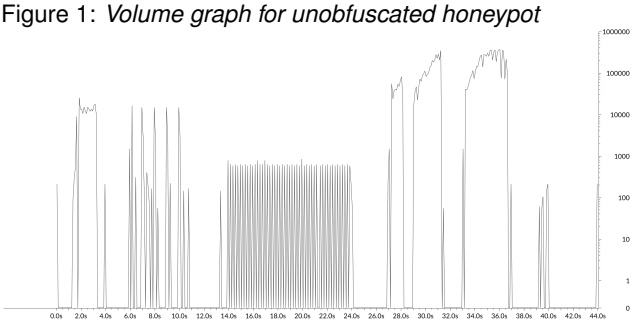
We will then examine packet length logs and look at the shape of the time-volume graph to qualitatively measure how easy it is to attack a system based on packet length analysis and time-volume analysis. Wireshark is used for this step, and the graph is graphed with time-step of 0.1 seconds.

Testing was done on Debian Unstable, 32-bit PAE, in July 2014.

2.4.1 No obfuscation

We directly connected to the honeypot from China to a server in Singapore. This distance simulates a typical-latency internet connection.

Figure 1: *Volume graph for unobfuscated honeypot*



From the traffic graph we can easily see the 1-second burst, the 5 bursts afterwards, and even each individual 512-byte write is visible.

Furthermore, packet lengths are extremely distinctive. From the packet length graph around step 3 we see that every single 512-byte write corresponds to a 573-byte packet.

Finally, the last step, involving increasing burst lengths and pause lengths, also creates a distinctive pattern: the pause lengths do progressively get longer, and the burst lengths do the same. The actual lengths are also roughly 1, 2, and 3 seconds.

Clearly, using the honeypot, an OM adversary could easily capture targets using either packet length or time-volume analysis.

Figure 2: *Packet lengths for unobfuscated honeypot*

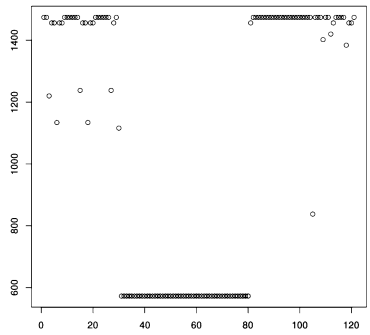
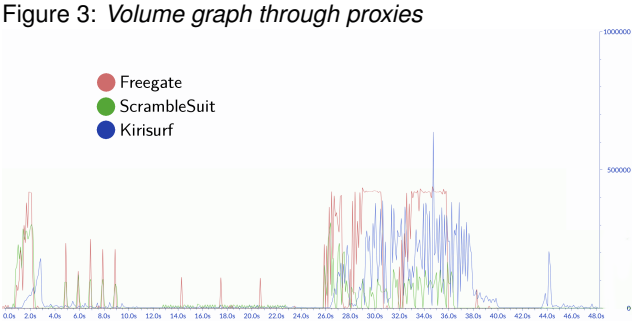


Figure 3: *Volume graph through proxies*



2.4.2 Proxied through Freegate

Freegate is a closed-source, but very popular, anti-censorship tool developed by Dynamic Internet Technologies (DIT) [16, 3]. It has partnerships with various dissident media [16], even including the Chinese version of Voice of America, which recommends the software on its website [15]. Based on our analysis, it uses a proprietary TCP-like stack running over UDP. It was never designed for obfuscation, yet countless people in China rely on it to freely access overseas websites.

We connected to the same honeypot server. Since Freegate produces lots of low-level control traffic, a linear rather than logarithmic scale is used on the volume graph.

Similar to the unobfuscated version, the first burst, the 5 small bursts, and the final series of bursts are obvious.

However, the 512-byte writes seem not to register well. Instead they are bunched into three large bursts, suggesting a periodically flushed cache. Furthermore, examination of packet length logs reveals no repetitive packet length.

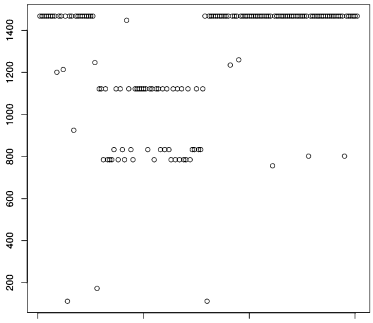
Volume-based attacks seem to work very well against Freegate; however, the caching may defeat the simplest of packet length analysis. Regardless, the proprietary UDP system is a sure-fire signature of Freegate; it is somewhat surprising Freegate works so well in China. Perhaps due to its relative lack of anonymity (all transmissions go through DIT-controlled servers), it is not seen as a large threat, unlike onion routers such as Tor.

2.4.3 Proxied through ScrambleSuit

ScrambleSuit [8] is an open-source, published obfuscation system. As mentioned before, many of Kirisurf's ideas borrow from those of ScrambleSuit. The system is built on the `obfsproxy` "pluggable transport" API from the Tor Project [14]; however, we use it directly to encapsulate our honeypot traffic instead of going through Tor. A linear scale is used on the volume graph.

The first spike is rather well preserved, as are the second series of exactly 5 large spikes. Rather surprisingly, the 512-byte writes leave a distinctive sawtooth pattern. The final three spikes and pauses are indistinct and quite well-obfuscated; this is likely due to ScrambleSuit's obfuscation of "inter-packet arrival time".

Figure 4: *Packet lengths for ScrambleSuit*



Even though the 512-byte writes remain distinct, packet statistics of those writes are drastically changed. Instead of a constant 573-byte stream, we see that some packets have length 1122, others 785, and others 833. Interestingly, no other packet lengths occur during step 3; the cause of this peculiar phenomenon is unknown.

In conclusion, the prominence of the first spike and the 5 large spikes, in addition to the exact count of the 512-byte writes, show that ScrambleSuit provides little or no protection against time-volume analysis. It does, however, contain reasonably effective protection against packet length analysis, and indeed it was designed for the latter type of attack.

2.4.4 Proxied through Kirisurf

We set up one Kirisurf node in Japan and use it to connect to our honeypot in Singapore from China. The use of a single hop on a single node is to limit the effects of incidental Kirisurf network topology and focus on Kirisurf's obfuscation component.

We note that the spikes have been notably reduced. Indeed, the large 16 kilobyte spikes from step 2, although still somewhat visible, are spread out to no more than 10 KB/s. Under less ideal conditions, this would easily disappear under the noise of other traffic on a typical Internet user's computer.

The 512-byte writes are completely indistinct in the graph, and the final bursts are well-obscured. Notably, the total

length of step 4 is hidden — junk segments drag out the bursts even after the actual payload has been sent. Packet lengths show no distinct pattern. One potential problem, however, is that there is a slight deficiency in MTU-sized packets during bulk transfers. An insufficient algorithm calculating when to refrain from sending junk segments is likely the culprit.

In conclusion, Kirisurf provides reasonable defenses against both packet length analysis and time-volume analysis. Unlike ScrambleSuit, it obscures volume patterns, and the packet length obfuscation is also improved due to a fortunate interaction with the volume pattern obfuscation.

2.4.5 Conclusion for tests

First and foremost, we note that none of the solutions completely defeat the honeypot. On the order of 10 seconds, the general form of the honeypot can still be seen. This implies that against honeypots, defenses against time-volume analysis still have a long way to go.

On the other hand, we see that it is indeed possible to obscure local volume features. Discrimination between different websites' traffic patterns, and honeypots based on, say, a few documents, would be much more difficult to pull off on Kirisurf. Unfortunately, currently deployed technologies such as ScrambleSuit or Freegate do not yet have meaningful protection against time-volume analysis.

Finally, we see that packet length analysis is much easier to defend against. Even Freegate, with no objective of obfuscation, inadvertently defended against the simplest form of this by using a periodically flushed buffer (which however does make it very unsuitable for applications such as SSH). ScrambleSuit protects against packet length analysis reasonably well. Of course, Kirisurf does the job even better, although it does not attempt to obscure inter-packet delays like ScrambleSuit does (we conjecture that the junk packets obscure this enough), and might be vulnerable to attacks on this and other packet statistics.

3 Network topology

3.1 Problems with traditional network topology

One large problem with existing onion routers is that each client usually needs to know the addresses of all of the nodes in the network in order to build circuits. However, this allows even a puny LO attacker to simply act as a client, get the addresses of all of the nodes, and add them all to a blacklist for censorship, or a watchlist for targeted surveillance. Tor presents a partial solution with "bridges", or special non-public entry nodes: people having trouble accessing "usual" entry nodes due to censorship can instead use bridges, which might even include rather sophisticated obfuscation [7, 13, 14]. Nevertheless, "silent" adversaries using watchlists without censorship, being extremely difficult to detect, would not prompt users under their attack to use bridges. These adversaries can still gather an accurate and comprehensive list of Tor users on their network. In many cases, this could result in targeted surveillance due to Tor's reputation as a privacy tool.

These silent adversaries become much more of a problem with adversaries with abilities belonging to OB and beyond. Mass surveillance of onion-router users presents a grave violation of privacy rights and net neutrality, but as long as the adversary does not use censorship, few people would even notice the surveillance. Users would then happily use unobfuscated onion routing, exposing themselves to eavesdropping. It seems clear that a new network topology must be able to:

- Give the client enough information to securely build onion-routing circuits
- Allow for low-latency and reasonably low-overhead communication
- Make gathering information about the addresses of each node difficult

The last bullet point is what separates Kirisurf's network topology from those of other onion routers.

3.2 Kirisurf's network topology

3.2.1 Node graph

Consider the sequence of nodes forming a circuit. In order for the circuit to be built, the following two conditions must be met:

- The client must be able to tell node n how to connect to node $n+1$.
- Node n must have the address of node $n+1$.

Note that crucially, node n does not need to know the addresses of nodes $n+2$, $n+3$, etc.

Kirisurf represents the entire network as a directed graph of online nodes. The graph obeys the following properties:

- Edge ij exists if and only if server i knows the address of server j.
- Servers are represented by their public key hashes.
- For all nodes i and exit nodes j, edge ij exists.

The first point ensures that each node, then, knows only the addresses of its direct successors, instead of everybody in the graph. The use of public key hashes means that clients can identify node $n+1$ to node n without knowing node $n+1$'s address. The last point implies that the addresses of exit nodes are public. The rationale for publicizing this information is because exit nodes' addresses are easily discoverable anyway by clients (by asking a destination what

the origining address appears to be); thus making the addresses public does not do any harm, but instead increases the amount of possible circuit paths and thus the difficulty of guessing which path sensitive information is going through.

As a side note, Kirisurf does not have "intermediate nodes". Either a node is exit or not — all other nodes are entry nodes. In this way, available circuits are increased, and more importantly, nodes cannot trivially find the addresses of its immediate predecessors, since incoming connections could also be ordinary clients building circuits.

3.2.2 Directory server

Kirisurf operates a central directory server trusted by all users by default. When a client connects to the server and asks for node information, the following steps are taken:

- Identify as client.
- Client is given entire graph of servers, with nodes labeled by public key hash.
- Client is identified by its IP address. This is hashed to produce an identifier.
- Based on bits in the identifier, certain servers' IP addresses are revealed.

The same process is repeated for servers, except that the servers identify themselves as normal or exit nodes. However, the directory then adds the server into the graph, with the revealed-IP servers from step 4 as its successors.

3.2.3 Circuit building

After receiving the node graph from the directory server, the client can then build circuits. The exact procedure is available on the online documentation for Kirisurf; we will give a high-level description here.

Before building the circuit, the client needs to find a valid path. Unlike in systems such as Tor, Kirisurf client cannot simply randomly select a few nodes — it is likely that no direct links exist between adjacent nodes in such a circuit. Instead, we use the following algorithm:

```
x <- random node with known address
circuit <- [x]
loop until circuit long enough AND
     last node of circuit is an exit

    last <- last element of circuit
    next <- random successor of last
    circuit <- append x to circuit

return circuit
```

"Long enough" is defined differently in Kirisurf than in Tor. In Tor, because each node in the graph is connected to every other node, the number of possible circuits is very high, and it is hard for one node in the circuit to guess the entire circuit. However, in Kirisurf, due to the limited number of each node's successors, guessing a circuit would be much easier. Thus in Kirisurf circuits of length 5 are default, while in Tor circuits of length 3 are default.

The next step is to actually build the circuit. Like all onion-routing systems, the circuit is consisted of layers of some encrypted transport system. In Kirisurf this is KiSS, or Kirisurf Secure Sockets, a system somewhat similar to TLS but vastly simplified and designed for easy, secure reimplementation. It uses Diffie-Hellman for key exchange and Blowfish for symmetric encryption; the details are available online [19]. When building a circuit, the client does the following:

```
wire <- connect to circuit[0]
wire <- KiSS handshake on wire
for each remaining element of circuit
  send "connect me to element" on wire
  wire <- KiSS handshake on wire
send "terminate circuit" on wire
return wire
```

"Connect" in this case uses the obfuscation system in part 2; note that the obfuscation is not layered and instead between each link. This reduces computational overhead and also provides additional obfuscation benefits: it is much harder to correlate incoming and exiting traffic on a given node since they are obfuscated with different random patterns. Furthermore, between two busy nodes, multiple circuit segments are multiplexed into one (see [18]) before being obfuscated, making it impossible to do attacks such as tracing circuit paths by counting changes in the number of incoming and outgoing circuits.

After "terminate circuit" is sent, multiple connections can then be multiplexed onto the circuit using the same protocol as that used to multiplex connections between the same communicating pair into one before obfuscation.

3.2.4 Circuit utilization

Tor tends to use the same circuit for the same destination host [7]. This does provide greater compatibility with the general web application assumption that a client would not randomly jump around the world within a session, but it has two subtle problems:

- Volume analysis of one node may reveal websites clients are accessing. In particular, if volume analysis is done on the entry node, websites particular clients are accessing could be discerned.
- Destination host, say, a website, can "tag" a user more easily even without the help of devices such as HTTP cookies — the traffic patterns of a certain IP address could reveal the presence of a certain user.

One may object that volume analysis should be defeated with obfuscation, and users should be careful about information leaks to the website they access. However, defense-in-depth

dictates that we should have multiple defenses against the same attack, to guard against vulnerabilities in one.

Kirisurf, on the other hand, maintains a buffer of 8 circuits. Connections on multiplexed onto them in a round-robin fashion. These circuits may have different exit nodes; to a destination host different connections would look like they are coming from different people. In addition, if, say, one entry node is under surveillance by the adversary, traffic analysis would be far less effective since the traffic only represents one connection to a website, which typically carries a small and unpredictable fraction of the website's data.

4 Bootstrapping problem

4.1 Chicken and egg

One annoying problem present in all censorship circumvention systems is the bootstrapping problem: a client must know some information in order to connect to the network, such as server addresses or node layout graphs. How should this information be sent to the client? Censors could easily find the source of the information and block it.

Some systems use the approach of embedding bootstrapping information, hard-coded, into the binary of the program; Ultrasurf, a system extremely similar to Freegate and made by the same consortium GIFC [17], partially uses this method [2]. However, as soon as one binary gets into the hand of an adversary, blocking of the tool would become trivial. For example, dumping the memory image of Ultrasurf and Freegate running on a disconnected computer reveals the exact hardcoded IP address list [2, 20].

Another way of attacking the problem is to use complicated proprietary protocols to negotiate bootstrapping info. For example, when Ultrasurf detects that all of its hardcoded servers are blocked, it uses a convoluted method involving specially formatted DNS queries to obtain new addresses [2]. Compared to the previous approach, this has the advantage of continuing to work (by sending new bootstrapping information) even if adversaries understand the protocol and obtain the bootstrapping info. However, these protocols are easily recognized by OM adversaries (or even OB if the protocol is simple enough), and the bootstrapping process itself could be blocked.

Finally, some systems, such as Tor, assume that bootstrapping is not blocked by adversaries for most of its users [7], and leaves censored users' bootstrapping as a problem to be solved by third parties, using systems such as "pluggable transports" [14]. Indeed, Tor encourages using ad-hoc methods for censored users to obtain bridge addresses for bootstrapping, such as e-mailing friends in uncensored countries [13]. There is no standardized way of distributing bridges.

The reason why bootstrapping is so difficult is that it presents a chicken-and-egg problem. To obtain the bootstrapping information, some sort of authenticated and hard-to-censor channel must be available. However, if such a channel exists, then there won't be a need for censorship circumvention in the first place!

4.2 Kirisurf's solution

To solve this bootstrapping problem, we observe that we have much looser requirements for the censorship-resistant tunnel used for bootstrapping than the censorship-resistant tunnel used for the actual encapsulation of onion-routing streams. In particular, the bootstrapping tunnel can have extremely low throughput, and only needs to distribute an infrequently changing piece of data instead of arbitrary streams.

These channels are ubiquitous on today's Internet. In fact, any HTTPS-enabled website with user-editable content fits this requirement. It is impervious to eavesdropping by adversaries, the only method of censorship involves great collateral damage (HTTPS websites can only be blocked in their entirety), and it is authenticated if we use a static URL. The URLs of these information drops are then hardcoded into the binary. If the information at these URLs is retrieved by the adversary and blocked, we can then simply edit the content at those locations to point at another, unblocked, directory server. Directory servers are obfuscated, so users' connections to it are hard to block directly, and authentication of the directory prevents entities hosting the URLs from changing the content to point to a fake directory controlled by the adversary.

5 Conclusion and further work

In this paper we have described a different threat model more corresponding with the real world, and innovative countermeasures to this threat embodied in the experimental Kirisurf onion-routing software. By using a novel system of obfuscation, a more diffusing circuit utilization strategy, and clever tricks to solve the bootstrapping problem, Kirisurf mitigates much of the powerful attacks our model adversaries could mount against traditional onion routers like Tor or anti-censorship software like Freegate.

Our further research lies primarily in how to improve our bootstrapping strategy. Currently, adversaries can still run the bootstrapping process repeatedly, blocking the directory server address obtained. Ultimately it becomes a cat-and-mouse game of adversaries' polling versus our changing of the info stored at the URLs. This is clearly unsatisfactory. In addition, a GC level adversary, or even an OM adversary if the users are significantly biased towards a particular nation, can appropriate large amounts of IP addresses and use them as originating addresses to probe the directory, obtaining a complete list of online nodes for blocking and monitoring. Cryptographically secure unique identifiers on each copy of Kirisurf may pose a solution; however privacy concerns abound as the identifiers could potentially identify each user, permanently, to the directory.

References

[1] Roger Dingledine, Jacob Appelbaum "How governments have tried to block Tor"

https://svn.torproject.org/svn/projects/presentations/slides-28c3.pdf

[2] Applebaum, J. (2012) "Technical analysis of the Ultrasurf proxying software" https://media.torproject.org/misc/2012-04-16- ultrasurf-analysis.pdf

[3] Dynamic Internet Technologies software version history http://dongtaiwang.com/loc/news.php

[4] Phillipp Winter, Stefan Lindskog (2012) "How the Great Firewall of China is blocking Tor" http://www.cs.kau.se/philwint/pdf/foci2012.pdf

[5] obfs3 protocol specification. The Tor Project. https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/blob/HEAD:/doc/obfs3/obfs3-protocol-spec.txt

[6] obfs2 protocol specification. The Tor Project. https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/blob/HEAD:/doc/obfs2/obfs2-protocol-spec.txt

[7] Tor FAQ. The Tor Project. https://trac.torproject.org/projects/tor/wiki/doc/TorFAQ

[8] Phillipp Winter, Tobias Pulls, Juergen Fuss (2013) "ScrambleSuit: A polymorphic network protocol to circumvent censorship" http://www.cs.kau.se/philwint/pdf/wpes2013.pdf

[9] Moghaddam, Li, Derakhshani, Goldberg (2012) "SkypeMorph: Protocol Obfuscation for Tor Bridges" http://cacr.uwaterloo.ca/techreports/2012/cacr2012-08.pdf

[10] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, Thomas Shrimpton. "Peek-a-Boo: I Still See You: Why Efficient Traffic Analysis Countermeasures Fail" http://kpdyer.com/publications/oakland2012-peekaboo.pdf

[11] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, Dan Boneh "StegoTorus: A Camouflage Proxy for the Tor Anonymity System" https://www.owlfolio.org/media/2010/05/stegotorus.pdf

[12] Amir Houmansandr, Chad Brubaker, Vitaly Shmatikov "The Parrot is Dead: Observing Unobservable Network Communications" http://www.cs.utexas.edu/~amir/papers/parrot.pdf

[13] Tor Project: Bridges https://www.torproject.org/docs/bridges

[14] Tor Project: obfsproxy https://www.torproject.org/projects/obfsproxy.html.en

[15] Access methods. Voice Of America Chinese http://www.voachinese.com/proxy.html

[16] Dynaweb Chinese http://dongtaiwang.com/loc/phome.php?v=0

[17] Global Internet Freedom Consortium http://internetfreedom.org/

[18] Specification for n to 1 stream multiplexing. The Kirisurf Project https://github.com/KirisurfProject/kirisurf/wiki/Specification-for-n-to-1-stream-multiplexing

[19] KiSS 1.0 specification. The Kirisurf Project. https://github.com/KirisurfProject/kirisurf/wiki/Specification-for-KiSS-1.0

[20] Reverse engineering notepad. The Kirisurf Project. https://github.com/KirisurfProject/kirisurf/wiki/Reverse-engineering-notepad