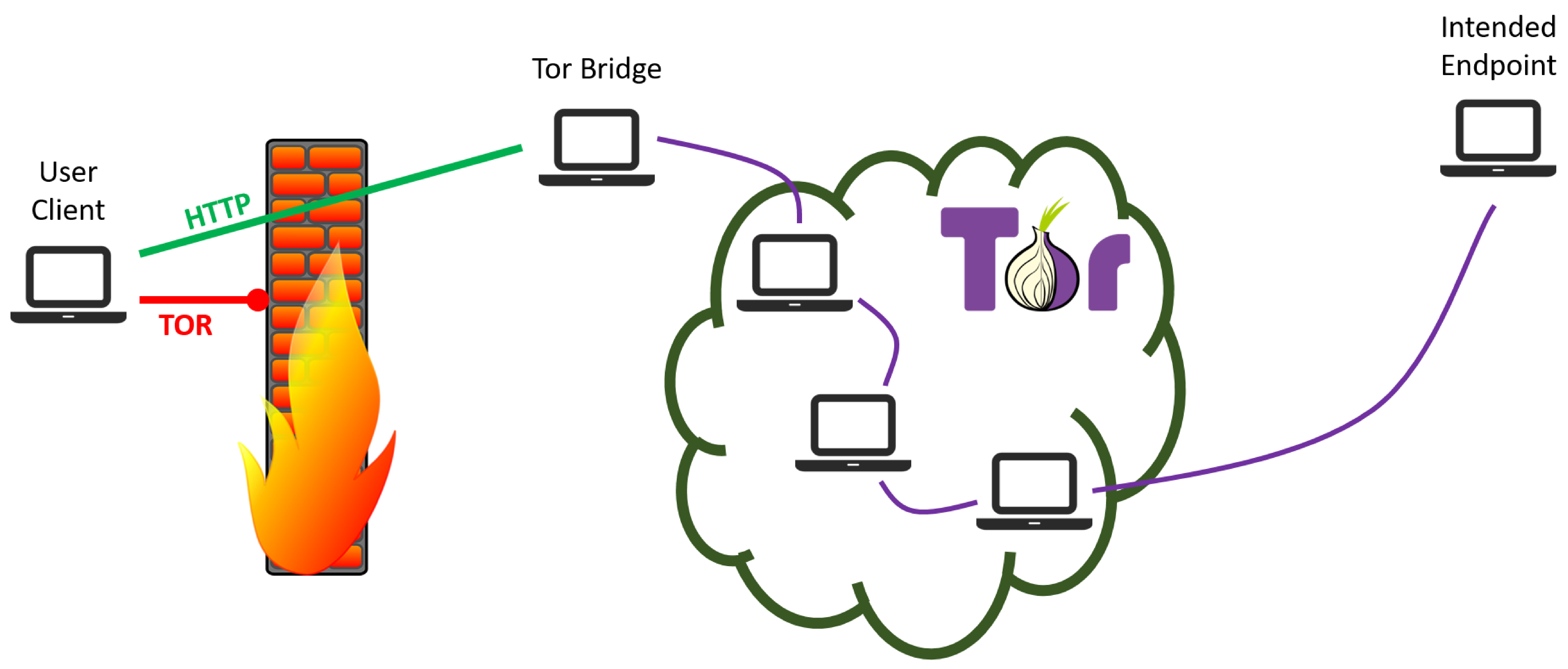
Innocuously Distributed Format Transforming Encryption

Samuel Russell

Professor Bogdan Warinschi



Introduction

I have been looking in to censorship evasion techniques with the aim to improve the availability to all on the internet. There are several situations in which censorship occurs but most poignantly there are state sponsored situations that controversially aim to block citizen's access to content deemed unaligned with state values. One of the recent developments is to use Format Transforming Encryption to hide banned data such as Tor packets inside ciphertexts that look like other protocols such as HTTP. I am extending this to rather than only create valid packet, to create packets that follow the normal background distribution as so pass innocuously through the network.

History of Format Transforming Encryption

The idea of Format Abiding Encryption is quite old and first was conceived to allow in place encryption of data in database fields. In this case, the ciphertext has to have the same format as the plaintext so that it abided by the database schema, however, the field has grown to allow mapping to separate format languages, leading to the name Format Transforming Encryption.
As it stands, the most efficient way of doing this is to rank the plaintext language such that each word in the language was mapped to a integer rank. This element of $\mathbb{Z}_n$ could then be enciphered using any standard block cipher before being un-ranked in to the output (possibly different) language. This rank-then-encrypt technique is efficient and allows any regular language to be targeted.

Application of Format Transforming Encryption

Encryption of a plaintext message usually leads to a ciphertext that is obviously not a plaintext. This may be since the encryption scheme outputs a tuple like in Elgamal $(g^r, h^r \cdot m)$ or the ciphertext is even just a group element in some group with strict notation such as elliptic curves. Instead, what FTE intends to do is by formatting these ciphertexts as innocuous traffic to fly under the radar of Deep Packet Inspection. Other hand-crafted solutions have been proposed that hide the ciphertext more discretely and also use traces to provide realistic packet distributions. However, since these are hand crafted once the particular details of the embedding is known it is easy to create (even low powered) adversaries to recognise the fakes.

Tor

One of the most popular tools for evading network surveillance is The Onion Router which wraps packets in multiple layers of encryption and passed them through a relay network which unwraps a layer at a time to anonymize the sender-receiver combination. Since this prevents outsiders from knowing what sites an individual visits, censors such as the Chinese Government have resorted to a complete ban on the service. Tor packets are very recognisable due to their predictable size and structure but some improvement have been made to adjust it's fingerprint to match TLS as much as possible.
Tor works by allowing users to connect to one of the publicly known entry guard relays however, since these IP addresses are public they are easily detected and interfered with by in-line systems. One way this is done is by sending reset packets to both parties telling them to close the TCP stream. To circumvent this, users instead connect to private bridges in uncensored zones (such as the US) so that the indented destination of the packets was hidden. However, with protocol fingerprinting by deep packet inspection (DPI) and the use of active probes censors can still catch these unlisted bridges in under 15 minutes of use! The use of FTE bridges has been able to fool these DPI systems for now since they fool the common protocol classification systems which use regular expressions also. However, as deep learning becomes more prevalent it is known that this will not be enough.

Contributions

Although, previous schemes of FTE produce syntactically valid words of the target language, the packets produced are very different to normal traffic patterns. This is partly because of the simple regular expressions used to describe the format languages, but mainly since the regular expressions do not give any information about the distribution of words. I am experimenting with different description formats of this distribution to produce much more "normal" looking packets.