Dissertation Type: Research

DEPARTMENT OF COMPUTER SCIENCE

Innocuous Ciphertexts

The DE-CENSOR Scheme

Samuel Russell

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Thursday 3rd May, 2018

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MEng in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Samuel Russell, Thursday 3rd May, 2018

Contents

Chapter 1

Abstract

Format transforming encryption is an extension of standard encryption that enforces a format onto the ciphertexts. This can be used to cause protocol misclassification of traffic by contemporary deep packet inspection tools.

Research Hypotheses

My research hypothesis is that statistical methods could be used by network based adversaries to distinguish between the use of format transforming encryption schemes and normal traffic. I also hypothesise that a practical replacement scheme will be able to produce innocuous ciphertexts whose distribution has undetectable divergence from the distribution of normal traffic.

Research Goals

- Critically discuss the downfalls of using an FTE system to encode ciphertexts.
- Build a statistical tool for distinguishing fake traffic produced by FTE from normal traffic.
- Define a new cryptographic primitive – DTE – which enforces an arbitrary distribution onto ciphertexts.
- Construct a new scheme – DE-CENSOR – that instantiates this new primitive to overcome this detection.

Chapter 2

Definitions and Acronyms

Novel

FTE	:	Format Transforming Encryption
DTE	:	Distribution Transforming Encryption
Normal distribution	:	Laplace-Gauss distribution
normal traffic	:	Traffic generated from a normal person browsing their normal sites on their normal web browser.
DE-CENSOR	:	Distribution Encoding of Ciphertexts to Emulate Normal Surfing for Onion Routing
Record Layer	:	Layer in the network stack that receives data from the application layer, encrypts it, fragments it to an appropriate size and sends it on to the Transport Layer.
SOCKS	:	Socket Secure – Internet Proxy Protocol
Statistical Indistinguishability	:	Cryptographic demand of DTE, for ciphertexts to be indistinguishable from the target distribution, by statistical methods.

Standard

AES	:	Advanced Encryption Standard
DES	:	Data Encryption Standard
DFA	:	Deterministic Finite Automaton
NSA	:	National Security Agency of the United States

Acknowledgements

I would like to take this opportunity to thank my supervisor – Bogdan – for his expertise and I would like to mention – Karl – for his machine learning incites. Also, for correcting my flaky grammar and phrasing, thank you to Jamie, Mike, mum and dad.

Chapter 3

Introduction

There is a strong belief among the creators and users of the Web across the globe, that it should be a space of freedom and expression, built for the many, not for the few. People globally, now use the Internet as their main source of information. It allows citizens who are otherwise cut off, by geographical or physical factors, a voice and access to the ever increasing pool of shared public knowledge. In this way, it is a big driver for equality across the world, as it can provide equal access to resources, such as online learning. There are economic as advantages as well as these social ones, providing a unencumbered, competitive space for businesses to collaborate and grow.

Conversely, the sharing of personal or false information, which might be in breach of privacy rights or against the law, could be unpleasant, misleading or even harmful to peopleâĂŹs lives. It is widely accepted that blocking on-line access to child pornography would be beneficial to society and this is now a legal requirement in many countries: forcing websites and Internet Service Providers (ISPs) to block this type of content.

However, this type of blocking can be taken further, with ISPs blocking the content they do not want others to see. Examples of this are the state-sponsored, censorship regimes of Iran and China, who block content of the Web in a bid to further their government's political agendas.

Therefore, there is currently an arms race between these network-based Internet traffic censors, and schemes to bypass their detection and blocking mechanisms. A technique called deep packet inspection can provide valuable information to system administrators managing network traffic, by informing protocol use and traffic content. However, use of some of these tools, especially on a state-wide scale, is controversial since it can be used to snoop on citizens and block content – as stated previously.

One of the core and proven systems for avoiding censorship is The Onion Router (TOR). Although originally built to primarily provide anonymity of users to the sites that they visit, it has now become important in also providing users with anonymity to network-based surveillance. However, because of this, censors such as Iran and China have resorted to blocking the entire protocol. Therefore, users cannot access the TOR network directly. To get around this problem, the concept of bridges have been implemented to hide the traffics destination, but even still this does not hide the protocol sufficiently.

Currently, the majority of the protocol detection schemes in use are based on regular expression parsers. As a result, there has been work to produce schemes that exploit this fact by: in the case of obfs4 [26] removing all formatting to avoid black-list detection; or in the case of Format Transforming Encryption (FTE) formatting the ciphertexts to match the definitions of white-listed protocols. These have had success in practice at supporting arbitrary web access for users in oppressive regimes and, currently, these systems are working. However, censors' abilities and techniques for detection are improving all the time. So, looking to the future, I am analysing a potential avenue for detection and pre-emptively finding a solution.

During this thesis, I explore the statistical analysis techniques that can be used to distinguish traffic produced using an FTE proxy from normal traffic. This begins with discovering a way to practically define a description of an arbitrary distribution like that of normal traffic, and then discusses techniques of how to compare two of these distributions. Then, I formalise a new type of encryption scheme to counter these detection techniques, called Distribution Transforming Encryption (DTE), by defining

its syntax and security requirements. Next, I explore how to sample from arbitrary distributions to produce normal-looking ciphertexts and then define my concrete scheme – DE-CENSOR – using my chosen method. After proving the security of my construction, I discuss building an implementation and evaluate the practicality and performance of using the scheme in a proxy situation.

Chapter 4

Motivation

Tim Berners-Lee said that since its conception, the World Wide Web has been built with the principles of openness and privacy [20]. These key values, that also lie at the heart of democracy, can ensure that if a website is made available on web, anyone can connect to it – no matter who they are or where they live. Traditionally, the Internet has been built as a distributed entity, to share the technological burden, but also to share responsibility and provide options – the very concept of a network can provide many routes for information to move from source to destination.

However, as organisations throughout the world realised the Internet's importance, in a bid to try to control it, they caused a trend of centralisation. For instance many VOIP systems are no longer pure peer-to-peer systems but route all traffic through a central server. Large Internet service providers control the backbone of the Internet and are in charge of handling the movement of information around the world. The world has become reliant on search engines to help us navigate the expanse of knowledge available to us, but this role is dominated by few big entities. Therefore, the internet today, especially for certain sub-networks in some parts of the world, is under the control of just a few powerful actors.

Furthermore, although the Internet started as a place of openness and accpectance of all, recent policies have been made which challenge this status quo; certain content on the internet is now subject to censorship laws in some places like the United Kingdom or China. The motivations behind this censorship can be well-meaning, for example, the blocking of illegal content such as child pornography with the intention of protecting the innocent lives of children. A further attempt to reduce damage to people's lives was the implementation of search engine de-listing of personal content petitioned by the right-to-be-forgotten movement [10]. However, this decision sparked much controversy since people believed that this deletion of history could lower the quality of the Web.

However, other cases of censorship are on a much larger scale and are not universally accepted as positive. One example is certain nation states blocking access to content that they deem to be inciting opposition to their political standpoints. This can involve blanket restrictions to foreign websites or restricting sites which may be hosted inside the nation, but are deemed to perpetuate negative views. Many countries ban hate-speech, or exclude it from their freedom of speech laws [23] and some countries have Libel rules [16] which ban spreading of false information. However, there are a number of nations that actually hide information about certain world events or ideologies from their citizens, depriving their civil rights [4]. This is dangerous for the world as it creates environments that can breed hatred directed at whomever the censors wants.

There is no doubt that the western economy and society has benefited greatly from the success of the web and its ability to connect people and share information. What ensures the web's success is its status as a permissionless space that leads to creativity, innovation and freedom of expression. This allows any new, small start-up companies and individuals to use this commodity to build services and communities without the need for legal approval or business deals, leaving their projects to be free to grow and prosper. Proposed Net Neutrality regulations would allow Internet service providers to prioritise certain site's traffic. This does not permit full censorship but even the slowing down of traffic could reduce the usability of certain sites so that users would avoid them. This would fundamentally coerce people's browsing behaviour to direct them away from certain topics.

Taking an existing example; in China, which is a state with among the most stringent Internet censorship practices, the American Chamber of Commerce in China [32] says that four out of five of its member companies report a negative impact on their business from Internet censorship. This gives financial incentives to ensure the Internet remains open.

4.1 Network Surveillance and Censorship

Censorship can be categorised into two modes. The first kind involves entities on the Internet such as websites taking down content so that I cannot be viewed. This is most poignant when the content has been made by others, such as on a micro-blogging site like Facebook. Although this is an issue and currently there is a debate about the balance of such censorship on these social media platforms, this is not what I will be looking at. This is because the data is gone at the source, rather than being blocked. Of course, there are tools that archive content that maybe later retracted or deleted such as The Internet Archive's project: Wayback Machine [2], and access to these resources is what the second type of censorship is about.

Informally, the Internet consists of each party's computer equipment connected together with a network of wires (and other mediums) and some management machinery to route communications to their destination. Communications are packet based and are directed along the connections according to the Internet Protocol (IP) [25]. This protocol header defines the destination of the packet for the purposes of determining the most appropriate route to follow. It also contains the address of the source so that error messages can be directed, to the right place, appropriately. Among other information, like version numbers and flags, there is also a field that tells the intermediary machinery when to delete the message, but this is only to handle cases where the destination is unreachable.

The second type of network surveillance and censorship sits at these intermediate points in the network and deals with how these packets are handled. By default, IP, and the transport layer on top of it (TCP) does not encrypt or sign the payload data. Therefore, it is possible for the intermediary parties to read and tamper with the packets and their payloads before they are sent or even dropped from the network.

From their observations of the Chinese Internet filtering systems, Zittrain and Edelman deduced several methods that were being used [12]. Filtering on the IP address of a resource can be done by analysing the destination field of the packet. This is good for blocking direct access to static sites since it is one of the simplest and therefore fastest to implement, though it relies on comprehensive black or white listing coordination.

The most common method of blocking is for the routers to drop these black-listed packets, but as Clayton et al. found in their practical experiments [9], for protocols that communicate in a streaming fashion over time, the network machinery sends TCP reset packets to both parties to interrupt the session and stop communication. This type of attack can be easily countered by ignoring all reset packets since, due to how the system is architectured, the original packets are allowed through. They also observed a less frequent technique which was the injection of a forged packet with random synchronisation numbers. When this reaches the destination, the server detects the incorrect value and validly triggers the sending of a reset packet. Now, if the reset packets are ignored then the two real endpoints become out of sync and the communication breaks down. Luckily, as it stands, the forged packets are poor imitations and so are easily detectable.

Another method used in practice is DNS poisoning, which involves incorrectly resolving URLs to redirect users either to a censorship notice or just a random – and useless – address. This only affects the use of URLs and so the use of raw IP address is completely unaffected, although many sites use them internally so it creates difficulty and work for users through setting up of a local host file.

Finally, packet inspection has become very common place. This can be content-based detection such as searching for trigger words in HTTP request paths, or in the HTML content returned. A good example of this is search terms included the query parameters of search engine requests. Although these snooping attacks only apply to clear unencrypted HTTP traffic and using an encrypted TLS transport layer hides all this information, sites that do encrypt, risk having the whole site IP blocked as above. This happened to Wikipedia in China when the site switched to enforcing HTTPS on all traffic. When the site did this, the Golden Shield Project – in charge of China's network surveillance – could no longer selectively block the specific culprit pages, instead, a blanket ban on the service was enforced.

The packet inspection and classification is also done on a protocol basis. Censors are usually tolerant to users accessing web pages over HTTP or video calling over VOIP protocols, however, protocols that allow the transfer of data in an opaque way such as VPN tunnelling, The Onion Router and TLS (less so) are often blocked completely.

4.2 A Possible Solution: The Onion Router

One of the most effective tools for evading network surveillance is The Onion Router (TOR) which not only hides the contents of packets using encryption like TLS but also obfuscates their destination. It does this by maintaining an opaque network which decorrelates the movement of packets in and out. As a result, it is hard to work out where traffic that enters the network at an entry node will exit, and then therefore what its final destination is. To realise this, each packet is encrypted three times (on top of standard TLS encryption) and, as the packet moves through the network, a layer of encryption is removed at each relay node. This means, as it mixes with other packets inside a relay's buffer, it is hard (computationally indistinguishable) to identify and match what is arriving with what is sent on. Paths through the network are randomly allocated and keys are exchanged securely using a Diffe-Helman protocol. This is done one node at a time through the existing secure link such that nodes only know the identities of their predecessor and successor in the chain.

However, there has been a lot of work and success around de-anonymising the TOR protocol to gather information on users. There has been research into both passive and active attacks giving varying levels of success but with varying levels of effort. Website fingerprinting attacks such as Cair et al. [8] only need to eavesdrop at a single point but suffer from a high number of false positives. Traffic confirmation attacks, which require access to both ends of the communication, such as Shmatikov et al. [29] need long periods to gain statistical significance. Active attacks, such as watermarking through delays, are a lot quicker but they involve a lot more work for the attacker since they need to maintain compromised relays in the network to apply the subtle modifications to the packets. However, government bodies with large influence and resources such as the NSA are interested in this field, so the use of these attacks is not out of the realms of possibility. Even so, if an adversary only controls a small proportion of the network, the chances of being allocated a fully comprised relay chain is small. Recently, Arp et al. released a side-channel attack for TOR called Torben [3], which is more reliable but less intrusive than other methods that use browser exploits. In fact in their research, Nithyanand et al. found that very high levels of active circuits were potentially vulnerable to autonomous and state level adversaries [22]. To counter this, they have built a TOR client that is aware of possible adversaries and is able to more intelligently select relay paths. This is successful in reducing the rate of affected connections but, as with all these anonymising techniques, sacrifices performance, adding latency to the circuit generation before any data can be sent.

Although TOR has these issues, these attacks are quick intensive so it makes the surveillance of users' browsing astronomically harder. In a presentation, released in the Edward Snowden leaks, entitled TOR Stinks, it reveals that the use of TOR makes wide-scale, continuous surveillance of users by the NSA impossible [13]. Instead, only directed, manual analysis can reveal the identity of users. Therefore, censors such as the Chinese Government have resorted to a complete ban on the protocol. Originally, TOR packets were very recognisable due to their predictable size and structure. For example, the encryption modes it used did not match those implemented in TLS. Therefore, simple, targeted attacks could easily detect TOR packets. There has been some improvement to adjust its fingerprint to match TLS as much as possible. However, this could not be done completely, so more had to be done.

Tor works by allowing users to connect to one of the publicly known entry guard relays. However, since these IP addresses are publicly known, they are easily detected and interfered with by network censor systems. As mentioned, 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 United States – so that the indented destination of the packets is 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 fifteen minutes of use! The use of obfuscation and FTE (described in section 6.3) bridges have been able to fool these DPI systems for now since they fool the common protocol classification systems which use regular expressions. However, as deep learning becomes more prevalent it is known that this will not be enough.

4.3 Summary

In this chapter, I have explained how there are motivations for wanting to access online resources with out the controllers of the network being able to determine the content of what you are accessing. I have discussed TOR's roll in this but also, that access to the TOR network is limited and that even TOR bridges are vulnerable since the traffic patterns are identifiable. Therefore, there is a need for encodings that cause misclassification of TOR traffic to other innocuous protocols.

Chapter 5

Classical Cryptographic Background

After language was invented to share knowledge, humans soon needed a way hide it from those whom it was not intended. The first schemes used word or letter substitutions or the shuffling of characters to make the meaning of a message hard to obtain. These techniques are still used as the basic building blocks today; for example in the AES block cipher, a very popular encryption scheme, of which there are hardware implementations baked into many of the major computer chips. Early schemes, however, relied on the secrecy of the method to protect the privacy of the message which meant that they cannot be reused by others. This lead to the introduction of parametrised constructions that use a key to alter the encryption such as the Vigenère cipher. Modern schemes obey Kerkoff's principle; that the security of the system only depends on the security of the key, allowing the same encryption method to be used by all.

The term – security through obscurity – is in stark contrast to the Kerkoff doctrine and advocates not sharing security system information. However, a well known saying is that "security through obscurity is an illusion" since; (a) there is no way of proving this type of security, and (b) schemes that are not in the public domain cannot get any where near the amount of scrutiny as those who are published, discussed and tested with others. These days, a lot of the research into cryptography is done by staff at universities, where publishing and collaboration is the norm.

What defines modern cryptography is the requirement to prove the hardness of all problems, or at least compare it to the difficulty of another problem. Information theoretic security is a level of security that is based on information theory. Proving this does not rely on any unproven assumptions but implies that there is just not enough information in the ciphertext to recover the plaintext. Perfect security is a subtype of information theoretic security in which implies there is not enough information in the ciphertext to obtain any information about the plaintext. One example of this, in the one-time-pad which involves using a key just as big as the plaintext and exclusive-oring (Xor) them together bitwise. Although this scheme has perfect security, it relies on sharing the same random key (of the same size as the message) between the two parties who wish to communicate without leaking it to anyone else. This feature of the scheme is unsustainable since for every message that is sent the two parties must meet up and share more key material Therefore, it is hardly ever used in practice.

Instead, schemes are used that require a much smaller secret. This secret still must be long enough so that it would take a *long* time to guess, but due to exponential scaling this can be quite manageable. For example, for a key of length 256 bits, it could take up to many times the age of the universe to guess. Also, the schemes shuffle and mix the ciphertext such that to work out what the key is, by mathematical constraints, many plaintext-ciphertext pairs are needed.

To be a bit more formal, encryption schemes are built from building blocks called blockciphers. A blockcipher is a set of keyed permutations each of a fixed uniform length n and an example of one is AES. These constructions, encrypt plaintexts of length n into ciphertexts of length n so to build a cipher that works over arbitrary length data, they need to be combined in some fashion. There are bad ways of doing this such as electronic code book mode (ECB), which suffers from block independence so certain patterns in the plaintext are obvious in the ciphertext. There are also good ways, such as cipher block chaining (CBC) which XORs the previous ciphertext into the next plaintext to make all ciphertext blocks dependent on all previous plaintext blocks. This has performance implications since to decode one block

you have to decode all previous blocks in the message so there are other modes tailored to that but, this also means the last block is a calculation that depends on all other blocks.

This same mode can then be used with a different key to *encrypt* the ciphertext, only keeping the last block. This last block can then be sent with the ciphertext as proof that the ciphertext is not changed while in transit, since recalculation at the destination should result in the same message authentication code (MAC). A third party could replace or edit the ciphertext, but without the second key, they can't reproduce the correct corresponding MAC. Hash MAC (HMAC) is when a compression function involving a hash is used to compress the input before a final encryption instead of many encryptions needed in a CBC-MAC. A hash function is a deterministic function that is infeasible to invert (first pre-image resistance), mixes the data such that a small change occurs in the input, a large change occurs in the output. It should be hard to find two inputs that have the same output (collision resistance).

The cryptography systems mentioned so far are symmetric in that they involve a single key, that is known to both parties by no-one else, to encrypt and decrypt. This is not practical in a lot of situations. Instead, we use an asymmetric scheme that uses different keys to encrypt and decrypt. This means that the decrypt key can be kept private but the encrypt key can be made public to everyone, such that anyone can encrypt a message and send it, but no-one, even the original author, can decrypt it, apart from the one with the private decrypt key. We name these keys; private and public respectively. The way the schemes work is based on a trap door function which is easy to compute in the encryption direction but the inversion for decryption is hard unless you know the private key. The major difference here is that you can calculate the private key from just the public key, but the system is built such that it would take a lot of computation and so if infeasible in real life. This partially solves the problem of key distribution, since it is all right to send the public in the clear, however, there is now a problem of ensuring the key a sending party is using to encrypt is the correct one. This authentication problem can be solved by more asymmetric cryptography called signatures which are the asymmetric version of MACs.

Although many symmetric encryption schemes produce ciphertexts that are indistinguishable from random bits, asymmetric schemes do not. This is because these systems are based on mathematical equations to link the public and private key. In many asymmetric schemes such as in the well known RSA cipher, the mathematics is based on group theory and the keys are definitions of groups and ciphertexts are made from elements inside them. Therefore, they might have some structure to define the separate elements, for instance, an ElGamal ciphertext is $(g^y, m \cdot h^y)$ where, $m, g, h \in \mathbb{G}$ which is the group. Furthermore, since the sizes of the groups $\mathbb{G}$ are not a power of two, the elements representations will not produce uniform bit patterns.

Steganography is the practice of concealing one message inside another. Most examples of this are just security through obscurity such as invisible ink or hiding text in an image by altering pixel values. Honey encryption which we introduced in 2014 by Juels et al. [19] has a similar tone to steganography, though instead of the resulting encoding appearing like innocuous data, it involves as ciphertext that can be decrypted to the original message or into an innocuous one. The approach was actually targeted for situations with low entropy keys, such that brute force attacks are possible. To counter this, the definition enforces that for any number of incorrect keys the scheme produces plausible-looking but bogus plaintexts.

Though similar, my distribution transforming encryption, like the format transforming encryption, is not exactly a steganographic scheme since the data is not embedded inside innocuous other innocuous data, but it is the innocuous data. This distinction means that it is more general and more efficient since well-looking cover traffic does not need to be produced to encode each new message.

Chapter 6

Previous Work

6.1 TOR Transports

The TOR community understands the necessity for disguising the TOR protocol as innocuous traffic. They have already produced several schemes that claim to help.

ObfsProxy simply encrypts the entire packet with a streaming cipher. Earlier versions used fixed parameters so although expensive, decryption could be done to obtain the standard traffic but now even that is not possible since a key exchange produces ephemeral keys. This procedure produces ciphertexts that are computationally indistinguishable from a random string so intensive computational statistics would be able to differentiate this from a background distribution of standard traffic. Also, due to its complete lack of structure, it would not pass any protocol white-listing filters.

StegoTorus is a system that uses carefully handcrafted embeddings for ciphertexts. In the HTTP mode, it selects randomly for pre-recorded traces and hides the chopped up ciphertext inside Javascript, PDF and SWF files and header fields like Cookies. Since, the masquerading protocol is HTTP, blocking encrypted protocols will not stop this. However, it relies on distributing large volumes of these traces or everyone recording their own, otherwise, censors could just block all of them specifically. This means the overhead in communication but mainly storage of traces is large

SkypeMorph transforms Tor traffic so it looks like a Skype Video Call. In a review of TOR transports Houmansadr et al. find that since its imitation of the Skype protocol is incomplete, a low-cost, passive attack that can always detect this encoding [18]. To be fair, the Skype ecosystem is very complicated as it deals with dependencies between multiple sub-protocols in different streams, talking to Skype directory servers and the VOIP recipient. Houmansadr suggests that one solution would be to run the whole Skype stack and then encode the data in the audio stream, with careful consideration of the lossy transmission. This, however, adds a complicated and intensive overhead to the communication.

In summary, although, each of these systems is being successfully used today, they each have floors that could be attacked in the future, which inhibit them from being universal solutions. Therefore, as invited by the TOR community, new systems are needed.

6.2 Format Preserving Encryption

In 1981, NIST [24] outlined a mode of DES that was able to restrict the ciphertext of an encryption to a string of a fixed, finite alphabet such as decimal digits $\Sigma = \{0, 1...9\}$. This allows plain texts such as phone numbers of form $X \in \Sigma^N$ to be encrypted to a ciphertext $Y = \Sigma^N$ that is also a phone number. Although that FIPS document is now withdrawn, it sparked further work such as what was done by

Brightwell and Smith [7]. They first specify the particular use case of encrypting data in place as inside a database where the ciphertexts must abide by the same type constraints of the database schema. The paper then goes on to describe their Datatype-Preserving Encryption scheme that again works with DES in a streaming mode and works over an indexed finite alphabet. However, the algorithm has redundant elements, such as shuffling the alphabet's ordering, that are not backed up with reasoning and there is no proof of security. More recently, Black and Rogaway [6] provide three constructions in their paper. They analyse the security of each one against the standard adversary Adv^{prp} and also discuss the practical considerations of scaling asymptotically and in practice.

The field has become a highly practical one, with companies such as Voltage moving to take advantage of the concept. They are using it to upgrade older systems by adding a layer of security that is invisible to the applications that sit on top since does not change the structure of the data. Therefore this vastly reduces the costs of this retrofitting since the need to completely redesign the system is usually eliminated. As of 2018, HPE [30] are still pushing it as part of their SecureData product to help maintain legacy software.

To keep up with the work in industry, Bellare et al. wrote a comprehensive paper [5] that formally defines the goals of FTE. They did this in the form of adversary based games. They redefine the standard Pseudo Random Permutation (PRP) Security game which challenges an adversary to distinguish the encryption scheme from a random permutation. The paper also gives weaker notions they think are more appropriate. Single Point Indistinguishably (SPI) requires the adversary to be unable to distinguish between the encryption of any chosen plaintext from the specified language and a randomly selected ciphertext of the same language. The Message Recovery (MR) game gives the adversary the frequency distribution of plaintexts and asks for a decryption of the ciphertext of a randomly selected item from the plaintext space. The adversary is given adaptive access to an encryption and verification oracle. Finally, they define Message Privacy (MP) as the inability of an adversary to compute a function of a plaintext given the ciphertext; Note in this game the adversary can select the function it hopes it will have the most success with. They state the following strength hierarchy for these games, showing PRP does give us the Message Recovery property we ultimately want.

$$PRP \implies SPI \implies MP \implies MR$$

The first implication is trivial since a random function will produce a random element when given the message. The last implication is also trivial since MR is just a special case of MP – where the function is fixed to be the identity. For the proof of $SPI, \implies MP$ the paper shows how an adversary B for SPI can be constructed using an adversary A for MP with sufficiently large advantage. In this construction, B is able to run A and satisfy its calls to the encryption oracle using its own oracle in turn.

6.2.1 Rank and Encrypt

In the second half of the paper [5] Bellare et al. then proposed a framework called Rank then Encipher. This framework supports and enhances the methods already discussed since it enables any encryption scheme that supports arbitrary size alphabets to be used to encrypt more complex languages. It simplifies the complexity of language features such as a checksum character, by mapping words to a simple index so that the encryption scheme used can be generic. It does this by partitioning the language into slices, each containing the words of a given length. That is, for the language $\mathcal{X} = \{\mathcal{X}_N : N \in \mathcal{N}\}$ for lengths in $\mathcal{N}$, each slice $\mathcal{X}_N = \{X : X \in \mathcal{X}, |X| = N\}$. Now the elements in each slice can be arbitrarily ordered as $\mathcal{X}_\mathcal{N} = \{X_0, X_1, \cdots, X_{n-1}\}$ and a bijective mapping can be defined $rank_N :: \mathcal{X}_N \to \mathbb{Z}_n$. Then for the whole language, we can define

$$rank :: \mathcal{N} \times \mathcal{X} \to \mathbb{N} \cup \bot$$
$$rank(N, X) = \begin{cases} rank_N(X) & \text{if } |X| = N \\ \bot & \text{otherwise} \end{cases}$$

$$unrank :: \mathcal{N} \times \mathbb{N} \to \mathcal{X} \cup \bot$$
$$unrank(N, i) = \begin{cases} rank_N^{-1}(i) & \text{if } i \in \mathbb{Z}_{\mathcal{X}_N} \\ \bot & \text{otherwise} \end{cases}$$

Then if we have an encryption scheme $Enc :: \mathcal{K} \times \mathcal{N} \times \mathbb{N} \to \mathbb{N}$ that works over arbitrary domains to encrypt values into $\mathbb{Z}_{|\mathcal{X}_N|}$, we can define our scheme $\mathcal{E}_k^N = unrank_N \cdot Enc_k^{|X_N|} \cdot rank_N$. Using this

construction, they state how the security properties of the original encryption scheme, over the finite domain, is inherited by the rank then encrypt scheme over the language.

They provide a ranking and un-ranking algorithm for regular languages inspired by the work done by Goldberg in '85 [15], but to make this efficient, pre-computation is needed. For each word length, a table of path numbers has to be created which can be done through dynamic programming.

Furthermore, using Mäkinen's [21] algorithms, if a language can be described an unambiguous context-free grammar then ranking can be done in linear time and unranking can be done in $O(n \log n)$ time, if we allow $O(n^2)$ time and space preprocessing. This is since the left Szilard language of the grammar can be more easily ranked and unranked. Conversion back to the original grammar is as simple as linearly applying the rules and conversion from a word to its left Szilard counterpart depends on the efficiency of parsing, which, is not always linear, but for unambiguous grammars is still efficient.

6.3 Format Transforming Encryption

Inspired by the rank and encrypt method from Bellare's work, Dyer et al. outlined in their paper [11] a new cryptographic primitive that extends standard symmetric encryption to produce ciphertexts that follow a selected format. This is done by chopping the front off the scheme $\mathcal{E}_k^N$ defined above and interpreting the input bit stream as a number that represents the ranking.

Their scheme is targeted against the regular expression based deep packet inspection (DPI) schemes that are commonly used in industry. These high end DPI tools are deployed by network administrators all over the world as they are able to label traffic by protocol using the regular expressions as definitions. They are attractive because they are fast, due to the speed of finite state machine automatons and can single out suspect types of traffic for further analysis of payloads or outright blocking.

They show in their paper that FTE leaves these systems vulnerable to protocol misclassification by targeting the allowed formats that can either be extracted from the enterprise software or learned by taking network samples. They also show that the formatting encryption expansion factor for realistic regular expressions is low enough to enable web surfing over the TOR protocol.

Formally, the scheme takes a formatting regular expression grammar $\mathcal{F}$ and lets $L(\mathcal{F})$ be the language defined by it. Then, using an existing authenticated encryption scheme Enc_k, the original message M is encrypted and the unformatted ciphertext Y is interpreted as an element of $\mathbb{Z}_{|L(\mathcal{F})|}$. The encoding function $unrank :: \mathbb{Z}_{|L(\mathcal{F})|} \to L(\mathcal{F})$ is then used to format it to specification. To make `unrank` and its inverse, decryption counterpart `rank` efficient the same precomputation is done as by Bellare and only a subset of the formatted language with fixed length is used. This is optimisation is key to be able to use this in practice. This turns the use of the scheme in to a block cipher used in Electronic Code Book mode. However, this lack of diffusion does not lead to pattern leakage since a random IV is used for each large block.

They have implemented this scheme and integrated it in to a TOR transport. Although, the regex based parameters make the scheme flexible so any protocol (described by a regular expression) can be targeted they have focused on hiding inside the HTTP protocol. To be precise they use the regular expression in figure 6.1. This hides the ciphertext inside the path of a `GET` request but due to the lack of structure – any alpha-numeric string – the packets are not visually convincing as seen in figure 6.2.

Although as a TOR proxy, the FTE scheme has had success in bypassing China's Golden Shield Project (GSP) censorship mechanisms as is listed as a official TOR transport.

```
^GET\ \/([a-zA-Z0-9\.\/]*) HTTP/1\.1\r\n\r\n$
```

Figure 6.1: Regex pattern to describe the HTTP protocol

6.4 Attacking FTE

Although FTE produces validly formatted ciphertexts which satisfy many current filtering systems, the ciphertexts look in no way normal. Informally, this is in the same way auto-generated C source code

```
GET /EopwqPHDbCy8WfNPEO3BPB.UUXEJ7xe2NoW/fSZQny4.x/jacDVkxjZQ4uSqgZu7.N2AGbaYeFqr/DEh
bHSMhLjVf2cZy7z7iBGOhwsgtCegXuBNZn//VGXFLUmG55Hlk5bsMMhoO4tqF.mYApGxAd2cOG/goOnZLivQB
.qTAzDFwxPtxS1i.bw2Oud2BQ7RtZaLoAcBJ4OfjLwsLi7v18Oi9Q6cUqTqHKTMsVWijB9/kh HTTP/1.1
↩
↩
```

Figure 6.2: Example packet generated by FTE

looks very different to any C code that a human would write. This means that statistical analysis such as deep learning techniques will discover that these streams are in fact fake.

To do this, however, the problem comes in defining the distribution of normal traffic. Many distributions can be defined easily, even those over countably infinite or continuous domains. The key to the tractable definitions of these distributions is the structured shape that the frequency density graphs have. This means that the frequency values can be defined using a formula. For example, the Gaussian's probability density function (PDF) ϕ has the classic bell curve shape and the following succinct definition.

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{1}{2}x^2}, f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{1}{2}(x-\mu)^2}$$

Furthermore, the Gaussian can also be used as a parametrised distribution f to produce uncountably more succinctly defined continuous distributions over the choices of σ and μ.

However, as seen in figure 6.3 the normal traffic distribution does not seem to follow a neatly defined distribution. Therefore, this distribution must be completely defined by values for every possible observable outcome. We can fix a maximum length of the output space as practically the network protocols have packet size limits.

However you encode it, to fully define the probability density function for a bit stream of even a reasonable fixed length n, you need intractable exponential space. For a concrete example, let us calculate the space requirement for storing the distribution of the possible values of a block of the payload of size 10 bytes long could take. Assume a 4 byte fixed point or floating point value is used to store the proportions, depending on the variation of the values. The storage size would be an intractable $4 * (256^10) = 4 * (1024^8) = 4$ yobi bytes.

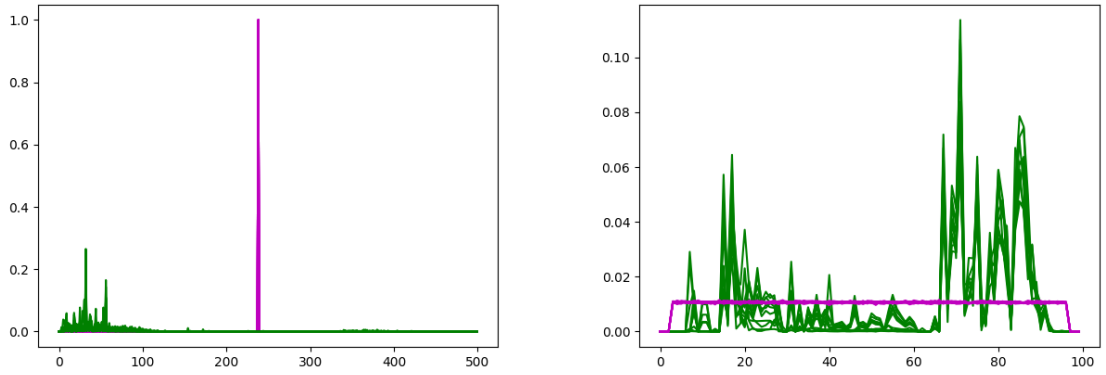
6.4.1 Heuristic based Distributions

Therefore, obviously simpler distribution descriptions were needed – i.e. heuristics. The first target is to design a distinguisher for the existing FTE scheme from normal traffic. The first heuristic was targeted at the fixed width that Dyer scheme produces. It can be clearly seen in figure 6.3a that compared to normal traffic the difference is obvious. Even a uniform random distribution is quite different.

In this case, the probability of the packet being exactly 256 bytes is overwhelmingly greater for the FTE scheme than for normal traffic. Of course, legitimate packets may also be this length so filtering them all out would give false positives. This might be okay if when the TCP layer re-sent the replacement for the missing, blocked data it would be of a different length and would be let through but in fact, it sends an exact copy each time. Therefore, to amplify the probability of not catching false positives, the system must take a sample from the stream and check them all as a batch. This could be done on a small running window basis and when triggered could add the IP address to a temporary ban. This is like what happens in currently deployed keyword filtering systems such as in China.

Another heuristic for defining the distribution is pattern frequency. This can be done at varying levels of granularity; the obvious level of abstraction is the character level. However, there are pros and cons of changing the measured pattern length. Short pattern sizes such as a couple of bits are extremely quick but do not provide much deep insight into the higher level patterns of the traffic. Longer patterns reduced independence since larger chunks are taken in context and so can catch more complex patterns, however, the complexity increases dramatically, as per our exponential calculation previously.

Other more less specific specific heuristics include `inter-slash-distance` and ways to extract pattern distribution information from only a subset of the URL such as the `initial-pattern` or `random-pattern`.



(a) Normal vs. FTE fixed width

(b) Normal vs. random width

Figure 6.3: Length Distrbutions

These less powerful measures can all be dealt with in the same way to parametrise an encoding.

6.4.2 Detection

For less obvious differences in heuristic values between normal and fake distributions more comprehensive comparison of the distributions are needed.

For testing distributions that are known to follow fixed-shape but parametrised models, such as a Gaussian, the Z-test can be used by testing the difference in means test statistic. The T-test is more appropriate for small sample sizes, where the population's standard deviation is hard to estimate since the Z-test relies on this to calculate the significance level. Although, due to the central limit theorem, these measures could be used, the arbitrary structure of the distributions in question it could result in false positives in the measurements if only the means of the samples were taken into consideration.

An exact test, such Fisher's exact test, will calculate the exact probability of obtaining each observation in the sample. Assuming the independence of the observations the overall probability can then be computed and then compared with a significance level threshold. However, in this situation, the independence assumption is unfounded and furthermore, the computational cost of this method is large, so for large sizes of our samples, other tests are recommended.

The Chi-square test and G-test are more appropriate for these large observation counts. In both of these tests, a test statistic is calculated and its value is then compared with the chi-square distribution to estimate the probability of obtaining that value. The subtle difference between them is that the Chi-square test's statistic is the squared differences of each frequency bin, whereas the G-test calculates the sum of log differences. Therefore, the chi-squared test penalises a larger difference more heavily than multiple smaller ones, whereas the G-test forgives these large differences by a log scale, so the opposite is true.

There is also a class of tests that measure the divergence between two distributions. These are not distance measurements since they do not obey the triangle inequality, however, they are useful for binary comparison. The divergence calculation, that Kullback and Leibler outline in their foundational paper, is a measure of entropy between distributions [28]. This measurement deals with the Shannon Entropy of the systems and quantifies the information gain needed to transform from one distribution to the other and is actually related to the G-test by,

$$G = 2\Sigma_i O_i \cdot \ln \frac{O_i}{E_i} = 2N\Sigma_i o_i \cdot \ln \frac{o_i}{e_i}$$

, where O_i and E_i are the observed and expected frequencies when N observations is made and o_i and e_i are the observed and expected proportions. However, it does have problems in application, such as asymmetry and possible infinite values. In response to this, Jensen and Shannon took a different tack for

their divergence measure, which instead measures the similarity between the distributions. This measure is symmetric and since it measures shared information it can only produce finite values.

6.4.3 Bringing it Together

We have seen that the standard implementation of FTE, is glaringly obvious to even casual detectors through its fixed length of output ciphertexts. Although this system could be updated to output several sizes of ciphertext, the pre-computation needed is different for each length value, so to add more than a couple of variations becomes massively inefficient. This makes the detection slightly harder, and so forces more test samples to be taken, but even so, the same targeted attack to check for those specific lengths will be successful.

In a more generic attack, using other heuristics, statistical and divergence tests can be used to measure how close the sample test to be tested resembles normal traffic. The Kullback-Leibler divergence test is commonly used in experimental designs and since its asymmetry is not an issue in this setting the proposed system will adopt it to measure the distance between the observed test distribution and pre-recorded reference distribution.

Deciding whether this divergence value is significant enough to trigger the detection can be done by comparison to a threshold. I discuss a method of calculating this threshold from positive and negative samples in section 8.4. However, this threshold can be adjusted by the censor to vary the false positive, true negative rate as applicable to their situation.

Chapter 7

Distribution Transforming Encryption

Informally, my scheme will produce ciphertexts that look the same as a stream of normal browser HTTP traffic. In practice, the scheme will draw samples from a pre-recorded reference distribution to emulate the target distribution. This encoding can be flexible to target different frequency distributions, for instance, to hide inside different protocols, but I will define a DTE scheme to target a fixed distribution.

Syntax

Let $\mathcal{D}_1$ represent the message source distribution and $\mathcal{D}_2$ represent the arbitrary target population distribution.

A DTE scheme $\mathcal{E}^{\mathcal{D}_2}$ that targets the distribution $\mathcal{D}_2$ can be defined as a 3-tuple: ($Kg^{\mathcal{D}_2}$, $Enc^{\mathcal{D}_2}$, $Dec^{\mathcal{D}_2}$), where,

$Kg^{\mathcal{D}_2} :: () \xrightarrow{\$} (k_e, k_s)$, produces a set of symmetric encryption and signing keys for a block cipher.

$Enc^{\mathcal{D}_2} :: ((k_e, k_s), M) \rightarrow C$, where M is a concatenation of the messages in a stream, and C is a concatenation of the ciphertexts produced. Over $\forall M \in Support_n(\mathcal{D}_1)$ this ciphertext stream looks like it is in $Support_n(\mathcal{D}_2)$.

$Dec^{\mathcal{D}_2} :: ((k_e, k_s), C) \rightarrow M$, where M is the original stream of messages the produced C.

Here, I am using $Support_n(\mathcal{D})$ to define a distribution of streams of length n that are the concatenation of messages sampled from $\mathcal{D}$.

Security Demands

A DTE scheme $\mathcal{E}^{\mathcal{D}_2} = (Kg^{\mathcal{D}_2}, Enc^{\mathcal{D}_2}, Dec^{\mathcal{D}_2})$ must uphold standard message privacy security, but also a new requirement – statistical indistinguishability.

Privacy I demand indistinguishability of ciphertexts under an adaptive chosen plaintext attack. Standard modes of authenticated encryption such as the cipher block chaining and encrypt the HMAC mode of AES provides higher levels of security. Though due to publicly known normal traffic distribution the produced ciphertexts are malleable, making chosen ciphertext attacks always possible. This must be inherited by the overall scheme to provide the standard assurances.

Statistical Indistinguishability This is a new demand that I am defining to ensure the closeness of the output distribution to the target distribution. This requirement does not refer to the computational indistinguishability that requires reduction to a hard known problem. Instead, it asks that for any well-behaving distribution comparison function Δ,[1]

[1] though does not need to be symmetric, or obey the triangle inequality,

Over $(k_e, k_s) \leftarrow Kg^{\mathcal{D}_2}, M \in Support_n(\mathcal{D}_1), N \in Support_n(\mathcal{D}_2)$; Measurement of $\Delta\left(Enc^{\mathcal{D}_2}((k_e, k_s), M), N\right)$ is small.

This quantification of *small* is left up to the creator of the construction. It can refer to a negligible function in one of the parameters but does not need to. The exact value is down to the priorities of the application. Adversaries that are willing to tolerate that some ciphertexts will get detected at the expense of a simpler encoding will want to relax the requirement. Yet, in other settings, where even one detection would cause large detrimental effects, it makes sense that you would tighten the bound on this constraint.

7.1 Exploration of Techniques

To create this DTE scheme, there must be a way to encapsulate not just the valid language but its frequency distribution. Firstly, I looked to the finite state machine produced by the regular expressions as used in FTE. Sadly, augmenting the node with weights according to the visiting frequency does not produce the desired effect. The adding of these weights to the DFA's state transitions would still produce a graph that could be ranked using a slightly altered algorithm. However, these weights would not be meaningful, since they would not be taking into consideration the previous path through transition graph and therefore previous symbols in the stream. In some graphs, where every different input stream parses through a different subset of the DFA states, this approach is valid. However, efficient DFAs do not do this since it has an exponential blow up, and specifically not the DFAs for the languages that we are targeting.

I also looked into ways of obtaining the structure of normal traffic using tools such as sequitur [31] to infer a grammar, but this too has the same problems.

As discussed in section 6.4 about attacking FTE, since the distribution of normal traffic is highly unstructured describing it in full is intractable, for the censor and here for the DTE scheme. Therefore, the suggestion was to instead concentrate on the distribution of values of certain heuristics which describe the packets. That idea can also be applied here, to generate packets that when tested against those heuristics will match the normal traffic distribution.

Different heuristics were discussed; such as lengths and pattern frequencies. Pattern frequency analysis can be done at varying levels of granularity to give varying results. The obvious level of abstraction is at the character level, which if used to base an emulation from, will only produce strings containing valid characters since they are what is extracted from normal traffic. Since patterns are recorded and generated independently, any lower level of granularity could produce byte values for non-valid characters such as control characters. This makes detection easy for a censor, as these values are not allowed in the protocol, so they can reject without even considering their frequency of use in normal traffic. Longer patterns reduced independence since larger chunks are taken in context, however, the complexity increases dramatically, as per our exponential calculation previously. Longer patterns have pros and cons; more character dependence, leading to more realistic URLs to the eye. Sequences of semantically correct content appear such as words and data structures like query string parameters. On the other hand, the complexity is increased dramatically, as per our exponential calculation previously. I will evaluate the effects of differences in granularity in section 9.1.

7.1.1 Sampling Methods

Rejection sampling

Rejection sampling is a technique for generating observations from a given distribution. It works by first sampling over the two-dimensional space of the proposed frequency density distribution and rejecting those that do not lie below the line. A classic, well-known, real-life example of this is the Buffon Needle experiment, in which the participant randomly drops needles on to a sheet of paper with evenly distributed parallel bars drawn across it. The needles that overlap a bar are counted versus those who do not and this proportion can be used to estimate pi as $\pi \approx \frac{2l}{t \cdot P}$ where l is the length of the needles, t is the gap between bars and P is the proportion. In this case, the first distribution is 3 dimensional; two spatial dimensions and one angular, and the second mapped distributions is binomial. The transformation formula uses

some elementary calculus and involves pi, so by knowing the outcome of the experiment, one can work backwards to estimate its value. However, the major downfall of this method is that the efficiency is greatly affected by the closeness of the two distributions, as more samples have to be rejected.

Inverse Transform Sampling

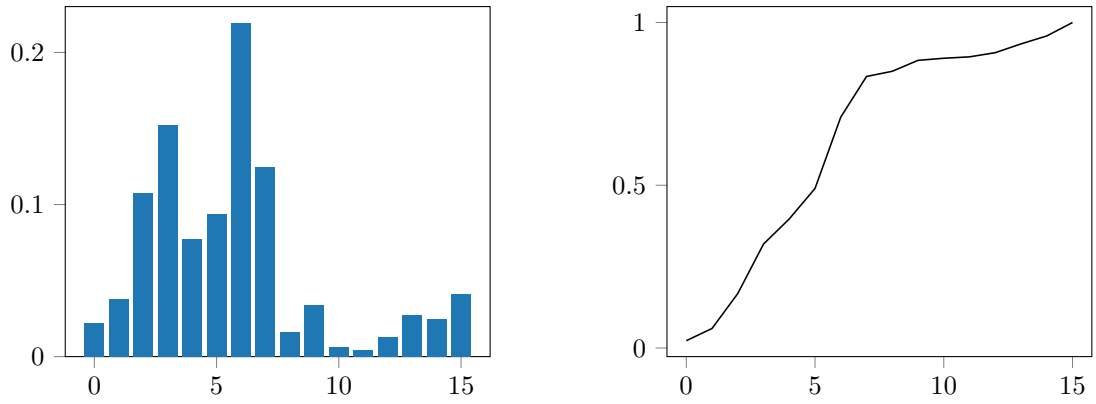
Another more efficient sampling method is called inverse transform sampling, which is commonly used for generating random numbers from arbitrary distributions. It is simple and efficient and can be used to generate values from the Normal distribution when given values from the uniform sources that are gathered by computer's random generation implementation. It works by generating samples of a different distribution and then transforming them using a formula. However, it works by calculating the inverse of the cumulative frequency distribution function so only works on distributions where this is invertible.

Fully Connected Neural Networks

Fully connected neural networks with a single hidden layer are theoretically able to model any continuous function. Therefore, they should be able to convert distributions. To test this, in early 2018, Horger et al. wrote a paper on training neural nets to generate samples from arbitrary distrobutions [17]. They trained the fully connected neural net to map uniformly generated inputs to a chosen probability density function by minimising Jensen-Shannon divergence. Their model was able to produce results that have comparable goodness of fit to other classical methods and after training was competitive in performance. Large amounts of data are needed to train the system, which if is to be done with realistic performance, needs special hardware to cope with the parallelism. Although, in the not so distant future this may become ubiquitous.

7.1.2 The Proposed Scheme

My proposed scheme works by generating encoded ciphertexts from a heuristic which can take a finite number of discrete values $\mathcal{X}$. For this, the scheme will use an inverse transform sampling due to its efficiency and ease of computation. The heuristics follow a distribution D_n defined by a frequency histogram, like shown in figure 7.1a. Since this is discrete, forming the cumulative distribution is simply the case of iteratively adding the bin frequencies and linearly interpolating the intervals as shown in figure 7.1b. Since we have a finite number of intervals, the inverse can simply be the reverse lookup function $\mathcal{F}_{D_N}^{-1} :: [0, 1) \to \mathcal{X}$. Now, assuming our input is uniformly random, sampling over the observations and applying this inverse transform $\mathcal{F}_{D_N}^{-1}$ will obtain values for our heuristics that are normally distributed. These values can then be used in the encoding to produced normally distributed ciphertexts.



(a) Probability Density Function

(b) Cumulative Probability Density Function

Figure 7.1: 4-Bit Pattern Length Distributions

7.1.3 Invertibility

To map the discrete input, un-encoded ciphertext values onto the continuous domain of $F_{D_N}^{-1}$ some quantisation must be done. However, the actual distribution structure does have an effect on the degree of quantisation (or more simply put, the amount of splitting) which can be done. Let s be the number of splits of the continuous space $[0, 1)$ and so $d = \frac{1}{s}$. To ensure that the encoding is invertible, as all encodings must be, we must ensure that our discrete version of $\mathcal{F}_{D_N}^{-1}$: $F_{D_N}^{-1}$ is injective and surjective.

Define:

$$F_{D_N}^{-1} :: \mathbb{Z}_s \to \mathcal{X}$$
$$F_{D_N}^{-1}(x) = x' \xleftarrow{\$} [x, x + d); \mathcal{F}_{D_N}^{-1}(x')$$

Surjectivity of $F_{D_N}^{-1}$ follows from the fact that $\mathcal{F}_{D_N}^{-1}$ is surjective by, $\forall y \in \mathcal{X}, \exists x' \in [0, 1)$ *st.* $\mathcal{F}_{D_N}^{-1}(x') = y$ and *let* $x = \lfloor \frac{x'}{d} \rfloor$ so $x \in \mathbb{Z}_s$ and $F_{D_N}^{-1}(x) = y$ so surjective.

However, injectivity is not provable in general. It depends on the exact instance of the distribution histogram. In fact, it is very unlikely for no overlaps between output quantisation and input quantisation to occur. Therefore, I made a tweak to make the system usable. This was to shrink the input divisions slightly so that they matched up to the nearest output division. This preprocessing step is outlined in implementation. It is important to note that if a correction makes the lower bound greater than the higher bound the process is invalid and the splitting rate must be reduced.

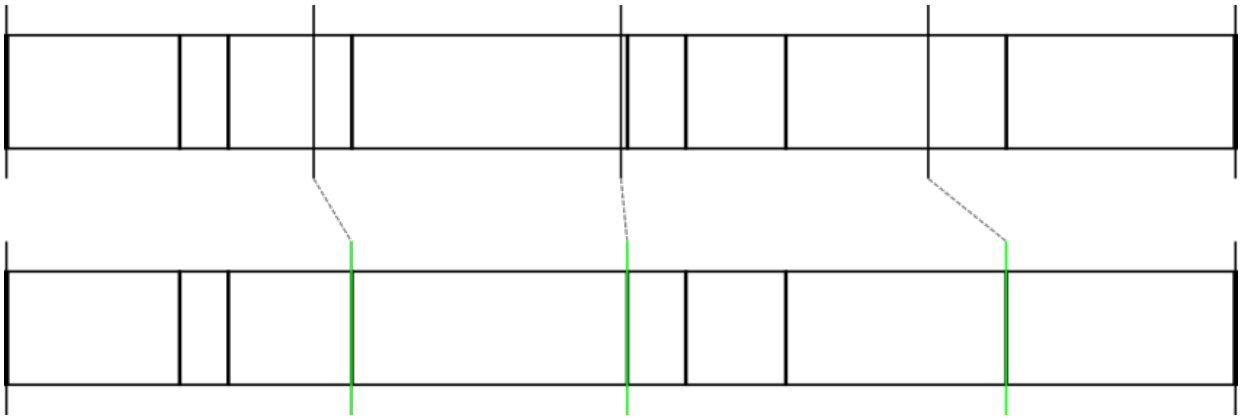


Figure 7.2: Showing how the input splitting is corrected to create an injective function.

In the diagram in figure 7.2, the smaller lines represent the output histogram boundaries and the longer lines represent the splitting of the input to the function $F_{D_N}^{-1}$. The green lines are the corrected splittings to force the function to be injective. This preprocessing creates an array of boundaries B. Augment B by setting $B = [0] + B$

Redefine:

$$F_{D_N}^{-1}(x) = x' \xleftarrow{\$} [B[x], B[x + 1]); \mathcal{F}_{D_N}^{-1}(x')$$

This new function still is surjective by roughly the same proof as before but its injectivity also holds. Proof: Firstly, the correction moved the boundaries of the inputs to match the boundaries of the outputs. The divisions started as non-overlapping and during the correction, this did not change,

$$\forall i \neq j \in \mathbb{Z}_s, \big[B[i], B[i + 1]\big) \cap \big[B[j], B[j + 1]\big) = \emptyset$$

Also since the boundaries match, the pre-image of a output value is contained by only one input value,

$$\forall i \neq j \in \mathbb{Z}_s, \alpha, \beta \in \mathcal{X} \text{ such that } \alpha \in Img_{F_{D_N}^{-1}}\big(\big[B[i], B[i + 1]\big)\big) \text{ and } \beta \in Img_{F_{D_N}^{-1}}\big(\big[B[j], B[j + 1]\big)\big)$$
$$\text{then } \alpha \neq \beta$$

Therefore, if we take arbitrary $i, j \in \mathbb{Z}_s$, such that letting$\alpha = F_{D_N}^{-1}(i)$, and $\beta = F_{D_N}^{-1}(j)$, then $\alpha = \beta \in \mathcal{X}$ then by the contrapositive above $i = j$, and so injectivity holds.

7.1.4 Towards a Concrete Scheme

The two most promising heuristics for encapsulating the traffic's distribution are length and pattern frequency. I am targeting length to demonstrate the advantage of my scheme over FTE. Pattern frequency over the whole ciphertext will also fool statistical analysis looking for the more specific heuristics that I mentioned earlier such as `inter-slash-distance` and `initial-pattern`. Therefore, I chose to encapsulate both of these in my scheme.

A simplification I have made is, like the FTE scheme, to restrict the encoding to only worrying about the path of an HTTP GET request. The scheme cannot work on the packet as a whole since the pattern independence of the scheme would not be able to produce the correct legal formatting. Instead, each field of the protocol needs to be dealt with individually according to its own distribution or heuristic and combined using a formatting algorithm like a regular expression. It could also be possible for there to be heuristics to decide whether fields such as `Cookies` should be included at all. However, each heuristic would be dealt with in the same way, so to limit the scope of the project to a proof of concept, I will only target the path field. The mechanics of the scheme will use the same structure as FTE. First, an authenticated encryption scheme is used to encrypt the original data using a symmetric key. This is then passed to the emulator encoding module, which uses the length and pattern heuristic distributions to produce normally distributed ciphertexts one heuristic at a time and shown in figure 8.3. The undistributed encryption is constructed from an IV, encrypted blocks, and an HMAC. Therefore, over a choice of random encryption and signing keys, it is indistinguishable from random. Therefore, no matter what the underlying data is, the uniform distributions that our encodings rely on is upheld. The inverse operation must use the same key and heuristic distributions, so can invert each encoding layer and then decrypt using the cipher. I refer to this scheme as DE-CENSOR: a Distribution Encoding of Ciphertexts to Emulate Normal Surfing while Onion Routing.

Pattern Encoding

The mechanics of the pattern encoding is basically a base transformation from base 0x100 (characters) to a base of the number of divisions of the pattern heuristic distribution. Once this new list of digits is produced, the scheme can apply our discrete inverse transformation function $F^{-1}_{D_N}$ to each one to map this to a list of output patterns which can be joined to form the normally pattern-distributed ciphertext.

The un-encode mechanism splits the string of patterns up and then applies the forward transformation function. The bases of the digits can then be changed back to bytes before forming the original character string once more.

Length Encoding

If length encoding took in arbitrary length ciphertexts, it would need to include the length of the original message into the encoding, such as pre-pending the data with its length as a number of bytes. However, whatever way this is done, the resulting output will not satisfy the uniform random requirement for the next level (pattern encoding). To counter this, another level of encryption could be done with another set of keys. However, this is undesirable due to the increased ciphertext length expansion and computational work. Another solution is to fix the whole scheme to only work on fixed sized blocks as input. TO enforce this messages can be padded before encryption using a secure padding scheme such as defined in ISO 10126 [14].

7.1.5 Formalising the Construction

DE-CENSOR is an encrypt-then-encode framework, and so the encryption and decryption operations are made up of 2 stages each. Let us define these formally.

$\mathcal{E}_{crypt} = (encrypt, decrypt)$ is the underlying encryption scheme.

$encrypt :: ((k_e, k_s), M) \to C$ is the encryption function of the authenticated encryption scheme.

$decrypt :: ((k_e, k_s), C) \to M$ is the decryption function of the authenticated encryption scheme.

Let, H be a description of each heuristic and D its value's frequency distribution. Then we can encapsulate $\mathcal{D}_2$, as $\mathcal{H} = \{(H_0, D_0), \cdots, (H_m, D_m)\}$ containing the information about each of the m heuristics used in the encoding.

$\mathcal{E}_{code}\mathcal{H} = (encode^{\mathcal{H}}, decode^{\mathcal{H}})$ is the underlying encoding scheme.

$encode^{\mathcal{H}} :: C \to N$, is the function that uses the heuristic distributions to transform ciphertexts C in to normal traffic patterns N.

$decode^{\mathcal{H}} :: N \to C$, is the function that again uses the heuristic distributions, but now transform normal traffic N back in to ciphertexts C.

Now we can use these to implement the DTE specification.

$$\begin{aligned} Kg^{\mathcal{D}_2} &: (k_e, k_s) \xleftarrow{\$} \{0,1\}^n, \text{ for key length } n. \\ Enc^{\mathcal{D}_2}((k_e, k_s), M) &: encode^{\mathcal{H}}(encrypt((k_e, k_s), M)) \\ Dec^{\mathcal{D}_2}((k_e, k_s), N) &: decrypt((k_e, k_s), decode^{\mathcal{H}}(N)) \end{aligned}$$

Privacy

To ensure the DTE demand of privacy, the construction must provide IND-CPA security. To ensure this, it will use an authenticated encryption scheme. Therefore for the underlying encryption scheme: $\mathcal{E}_{crypt}$, it will implement a cipher block chaining mode with HMAC using AES256 for encryption and SHA256 for authentication. Therefore, assuming the underlying block cipher – AES – is secure, this mode provides indistinguishability under a chosen plaintext attack.

This security guarantee is only for the encryption layer, however, since the encoding only has access to the IND-CPA ciphertext, no information leakage about the plaintext is possible. Therefore a reduction of IND-CPA security of the encryption layer can be reduced to the IND-CPA security of the whole scheme.

Here is the reduction:

Assume the existence of $\mathcal{A}$ to be the adversary that can distinguish ciphertexts of DE-CENSOR scheme $\mathcal{E}^{\mathcal{D}}$ with access to a plaintext oracle.

Define an adversary $\mathcal{B}$ to be the adversary that can distinguish ciphertexts of the underlying encryption scheme by,

- The game is randomly initialised: $(k_e, k_s) \xleftarrow{\$} Kg^{|k|}$ and $b \xleftarrow{\$} \{0,1\}$
- To produce m_0, m_1, $\mathcal{B}$ simply calls $\mathcal{A}$ and returns the same plaintexts, since they have the same input domain.
- The game then returns $C = encrypt((k_e, k_s), m_b)$
- To distinguish, $\mathcal{B}$ encodes C with $encode^{\mathcal{D}}$, using the public distribution description $\mathcal{D}$ to produce a normally distributed ciphertext N. $\mathcal{B}$ then runs $\mathcal{A}$ with N to get b^* and returns b^*. $\mathcal{B}$ is able to answer the oracle encryption queries of $\mathcal{A}$ for m' by using its own encryption oracle to obtain $C' = encrypt((k_e, k_s), m')$ and again using the public distribution description $\mathcal{D}$ to encode m' with $encode^{\mathcal{D}}$ and returning.
- $\mathcal{B}$ wins iff $b* = b$

Since $\mathcal{B}$ wins whenever $\mathcal{A}$ wins then,

$$Adv_{\mathcal{B}}^{IND-CPA} \geq Adv_{\mathcal{A}}^{IND-CPA}$$

Therefore, since the encryption scheme is IND-CPA secure,

$$Adv_{\mathcal{A}}^{IND-CPA} \leq negl(|k|)$$

where $negl$ is a negligible function and $|k|$ is the length of the keys.

then the DE-CENSOR scheme is IND-CPA secure.

Statistical Distinguishably

The construction will use the encryption scheme $\mathcal{E}_{crypt}$, as outlined in the privacy section. For this proof, we will only consider the simpler system with the boundary correction of $F_{D_N}^{-1}$ removed.

Also, for this construction requirement, we rely on a notion of security defined in Rogaway's paper called IND$-CPA which ensures indistinguishability from random bits under an adaptive chosen-plaintext-and-IV attack [27]. This abstraction of lifting the IV initialisation from the mode of operation such as CBC, to the user, produces a deterministic scheme. However, using a scheme that is IND$-CPA means that if the IV is treated as a nonce – a value, like a counter, that is used at most once within a session – the output bits are indistinguishable from random.

Therefore, if a scheme is built that,

- Chooses the IV also from random
- Uses an IND$-CPA scheme with the random IV to encryption M to C
- returns $IV||C$

we have a scheme whose ciphertexts are also indistinguishable from random. The underlying encryption scheme of DE-CENSOR: $\mathcal{E}_{crypt}$ is one of these schemes.

This means that,

$$encrypt\left((k_e, k_s), Support_n(\mathcal{D}_1)\right) \sim U_n$$

The discrete inverse transform sampling function $F_{D_N}^{-1}$ samples randomly within the division. Furthermore, since the division is uniformly selected by the uniform distribution generation by the encryption, the input to the continuous function $\mathcal{F}_{D_N}^{-1}$ is a continuous random real inside the domain. Therefore, by the definition of $\mathcal{F}_{D_N}^{-1}$, it will generate heuristic values that follow the distribution of normal traffic of $\mathcal{H}$.

So, by hybrid argument,

$$encode^{\mathcal{D}_2}\left(encrypt\left((k_e, k_s), Support_n(\mathcal{D}_1)\right)\right) \sim encode\left(U_n\right) \sim \mathcal{H}$$

Therefore, any comparison function Δ that is based on the heuristics $\mathcal{D}_2 = \mathcal{H}$,

$\Delta\left(Enc^{\mathcal{D}_2}((k_e, k_s), M), N\right) = 0$

Chapter 8

Implementation

I have written an implementation of the Censor Detection System (CDS) and also the DE-CENSOR scheme. The CDS's implementation consists of a measurement program that calculates the statistical similarity of batches of paths to a pre-recorded reference histogram of normal traffic. It compares the test statistic to a threshold – pre-calculated to separate normal from malicious requests. The DE-CENSOR scheme takes in messages, encrypts them and then encodes them using the selected heuristic emulator, that outputs an expanded ciphertext but one that is distributed according to its reference histogram. This can be the same reference distribution as the censor, but a high success rate is also possible using a separately recorded histogram of the same normally distributed background traffic. This is important since practically, it would not be possible to extract the used distribution from in-use instances of the tool.

All the code was written in `python2.7` for quick prototyping but, for performance improvements, the core could be re-written in `C/C++` to remove the overheads of the python interpreter. I conducted my implementation and testing on an Ubuntu 16.04 operating system, on VMware virtual machines running on a standard i5 Intel laptop.

8.1 Collection

Packet collection is used for both parties (CDS and DE-CENSOR) since both the emulator and censor need to record a reference distribution to work against. The CDS is also required to read the packets for classification. Since both my censor and de-censors are running on Ubuntu machines I used the PyShark a python wrapper around the Tshark non-interactive command-line tool. Tshark is also able to parse and filter the packets similar to the abilities currently deployed DPI systems. I used this feature to extract the packets I cared about using the filter in figure 8.1.

```
http && http.request.method == "GET"
```

Figure 8.1: Tshark packet filter

The filter shows that we only targeted GET requests. This is because HTTP is a very one-sided protocol, with most of the content moving from the web server to the client. Therefore, it is harder to hide messages inside the requests due to there normally low data size. The response encoding should be more straightforward since files returned from servers are usually much less structured. GET requests, as opposed to POST requests, take up a larger proportion of web traffic and so are the obvious choice to hide inside, although, in future work, a heuristic to produce a distribution of both could improve the scheme.

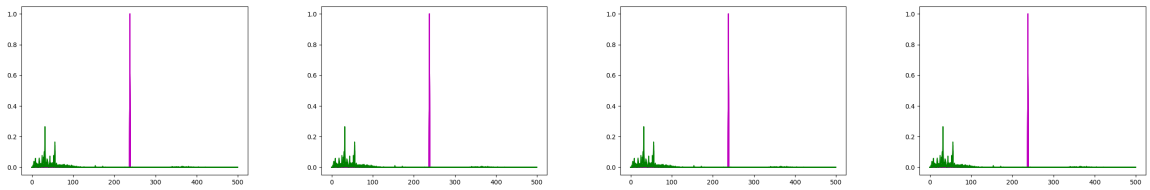
In this experiment, I have fixed all other HTTP fields apart from the path to simplify the implementation. I have done this since this is only a proof of principle and expanding the scope to create more complexity is left to a future goal.

To collect real web traffic data, I built a web-crawler using the FOSS Selenium web automation framework. To gather a realistic distribution I needed to emulate the whole browser environment since a large amount

of the requests from modern sites are dynamically initiated from inside the scripts. I crawled recursively over the Alexa top 50 sites [1] that support unencrypted traffic. This gave me a sample of 10,000 paths to create reference distribution.

8.2 Histogram creation

Like the recording of reference material, the calculation of distribution histograms is not time critical since they can be done offline. The process is straightforward frequency binning to count the occurrences of each option. I tried several methods as explained below.



(a) Path length (b) Character based (c) First Letter (d) Random Letter

Figure 8.2: Distribution Histographs

Path length – this was targeted at the FTE scheme that used a fixed length. It shows very few examples would be needed to differentiate the origins.

Character distribution – This graph exhibits the underlying encryption schemes uniform randomness, whereas in real traffic not all characters are created equally.

First Letter – This method was attempted to see if the same distinguishing effect would be possible only using a fixed subset of the URL.

Random Letter – like the previous this uses a limited sample by randomises its location to see if this improved on the above results.

8.2.1 Exponential Scaling

As the length of the patterns increase, the number of frequency bins increases exponentially, and therefore, if they are backed by a fully allocated array the space needed to record the frequencies quickly becomes unmanageable. For instance, even at pattern size of four characters, it will consume 4 GiB in memory. Next, we observe that very little of the large space containing the all the possible patterns is ever used since the samples collected are very small in comparison.

Therefore, instead of allocating a giant chunk of contiguous memory, the system will use a hash table. This dynamic structure handles the allocation of memory to match the number of items actually stored. Since we can estimate the expected number of items to be used during the analysis, we can also pre-resize the hash table to avoid the expensive operation of resizing in operation to avoid collisions.

8.3 Inverse Transform Function

As described in section 7.1.3, the inverse transform function $F^{-1}_{D_N}$ maps values drawn from uniform random to values that have a distribution defined by a frequency histogram. To implement this, I first transformed this into a cumulative frequency distribution and then extracted the upper and lower bounds for each of the output bins which represent each heuristic value. Then, sampling consists of sampling uniformly over the range of the whole cumulative distribution and finding into which bin's bounds the value falls.

This can be done by linear search but as the number of bins grows, this dominates the computation. Since the bounds are ordered we can replace the lookup with a binary search whose logarithmic complexity slows the increase of computational work.

8.4 Thresholding

The detection system's goal is to decide whether a sample of packets is from normally distributed traffic or from a stream of fake ciphertexts such as that is generated from an FTE proxy. It does this by calculating a corresponding statistic value for the batch of packets in question and applying a threshold to obtain the binary decision. The threshold is the point that has an equal probability of belonging to each distribution and to calculate this, the distribution types and parameters must be decided. By the central limit theorem, we can model the statistic's value as belonging to a Gaussian distribution, that parameters of which can be estimated from a sample. I took a sample of 10 streams of normal and fake traffic to get examples from each population and calculated the sample mean and variance. I was then able to estimate the population mean and variance and construct the following equation to constrain the threshold's value.

$$(\sigma_2 - \sigma_1)x^2 + 2 \cdot (\sigma_1 * \mu_2 - \sigma_2 * \mu_1)x + (\sigma_2 * \mu_1^2 - \sigma_1 * \mu_2^2 - \sigma_1 * \sigma_2 * \log(\sigma_2/\sigma_1)) = 0$$

This can be solved for x using the standard quadratic formula to obtain the statistic threshold.

8.5 Proxy Embedding

I embedded the DE-CENSOR scheme inside the same proxy framework as FTE. This enables point to point innocuous channel communication using a client-server model. A user runs a client application on their machine, which listens on a local port. Any application can then use this port as a SOCKS proxy, which channels the traffic through the innocuous channel to the server. The server decodes and decrypts the data and follows then SOCKS protocol to forward the traffic on to its ultimate destination. The keys and distributions must be pre-shared to allow the original traffic to be recovered, though I have included an example distribution as the default. The encoding happens inside a custom DE-CENSOR record layer which works on a series of buffers to transform the data before it is sent.

8.5.1 Encoding

1. Firstly, data is placed into a plaintext buffer as it arrives.
2. The encryption stage then consumes a fixed length block of the plaintext from the plaintext buffer and produces a ciphertext. This is placed in the encrypted-buffer.
3. The length encoding stage then transforms these encrypted fixed length messages into blocks of normally distributed lengths for the length-buffer.
4. Each length of message data is then picked by the pattern encoder one at a time, where it undergoes the final transformation into a normally distributed GET-request paths.
5. These paths are wrapped into an HTTP GET packet which is then sent across the network over an unencrypted TCP channel to the listening proxy server.

8.5.2 Decoding

1. As the server receives them, HTTP packets are placed into a packet buffer.
2. The path of each packet is then extracted and fed into the pattern decoder, which reverses the pattern encoding according to the pattern distribution.
3. The length encoding stage then joins the messages back into strings of fixed equal length.

4. The decryption stage then deciphers the ciphertexts and removes the padding to recover the original messages.
5. The SOCKS protocol headers are then used to pass on the original requests to the intended address.

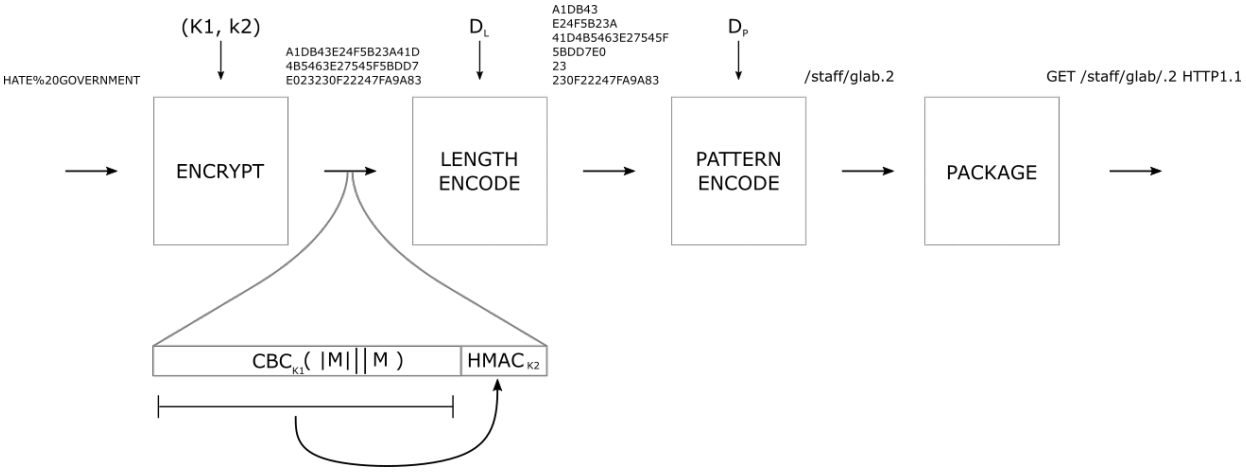
See figure 8.3 for an overview of the process.

8.6 Libraries

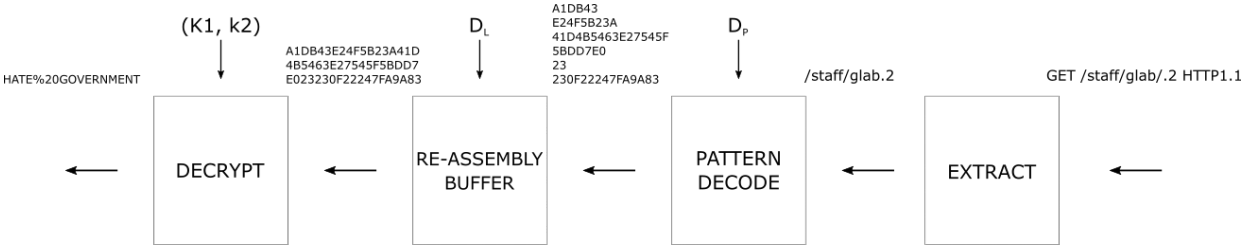
I used the Cryptography implementation from the PyCrypto library.This provides an AES block cipher, SHA256 Hash in a Merkle-Damgård construction and counter mode framework. The IV is chosen randomly.

The statistical distribution comparison calculations are done by the SciPy Library, which provides a C++ back-end to complete the heavy lifting and improves the performance.

I used the Selenium browser automation tool to control the Firefox web browser and produce realistic browsing traffic.



(a) Encoding and Sending Mechanism



(b) Receiving and Decoding Mechanism

Figure 8.3: Implementation Diagrams

Chapter 9

Critical Evaluation

After building the proposed scheme, I conducted some experiments to test its performance in a practical situation. Firstly, I will test and explain how adjusting a heuristic parameter affects spaces requirements and ciphertext expansion. Secondly, I will evaluate the scheme for its statistical indistinguishability property to ensure it holds when the boundary tweak is applied.

9.1 Evaluating Pattern Length Effects

As disused in section 7.1, pattern frequency can be used as a good heuristic to describe normal traffic. It also discusses that this can be done at varying levels of granularity. Here I will observe the practical outcomes that occur when the pattern length is varied between four bits (half a byte) and forty-eight bits (six characters).

Pattern Length	Distribution Bin Number			Max Splitting			Extension Factor		
	10000	20000	30000	10000	20000	30000	10000	20000	30000
4	16	16	16	5	5	6	1.7	1.7	1.6
8	81	82	89	14	14	19	2.1	2.1	1.9
16	3310	3906	5184	52	68	51	2.8	2.6	2.8
24	10115	16443	63273	101	88	100	3.6	3.7	3.6
32	16471	28031	97963	93	88	95	4.9	5.0	4.8
40	19464	33393	105573	88	95	99	6.2	6.1	6.0
48	19863	34881	104888	80	69	152	7.6	7.9	6.6

Table 9.1: Effects of changing Pattern length

Table 9.1 shows some trends that occur when the pattern length is altered. The statistics are gathered from analysis of one reference sample of normal URLs.

The expansion factor follows the following equation; let s be the maximum splitting before errors, and pl to be the pattern length.

$$|output| = |input| \times (\log_s 256 \times pl/8)$$

This stems from the conversion of bases from $0x100 = 256$ in a character to s divisions over the pattern of pl bits and therefore $pl/8$ bytes. The equation shows the linear correlation between pattern length and expansion, though the graph shows the rate of inefficiency increases slightly with pattern size. This is due to the how well the bigger pattern distributions split into the equal portions. Initially, the splitting amount increases but after 24 bits (3 characters) it pivots and gradually decreases. This is due to the increasing variance in bin size as the patterns get larger, causing the packing into divisions harder. I will discuss the techniques of packing in the implementation section.

The distribution bin number describes the number of non-empty bins in the histogram describing the distributions. For small pattern, this is very low, since for sub-byte patterns the unstructured definition of `ascii` representations means any character distribution tends to produce all patterns of that length. At

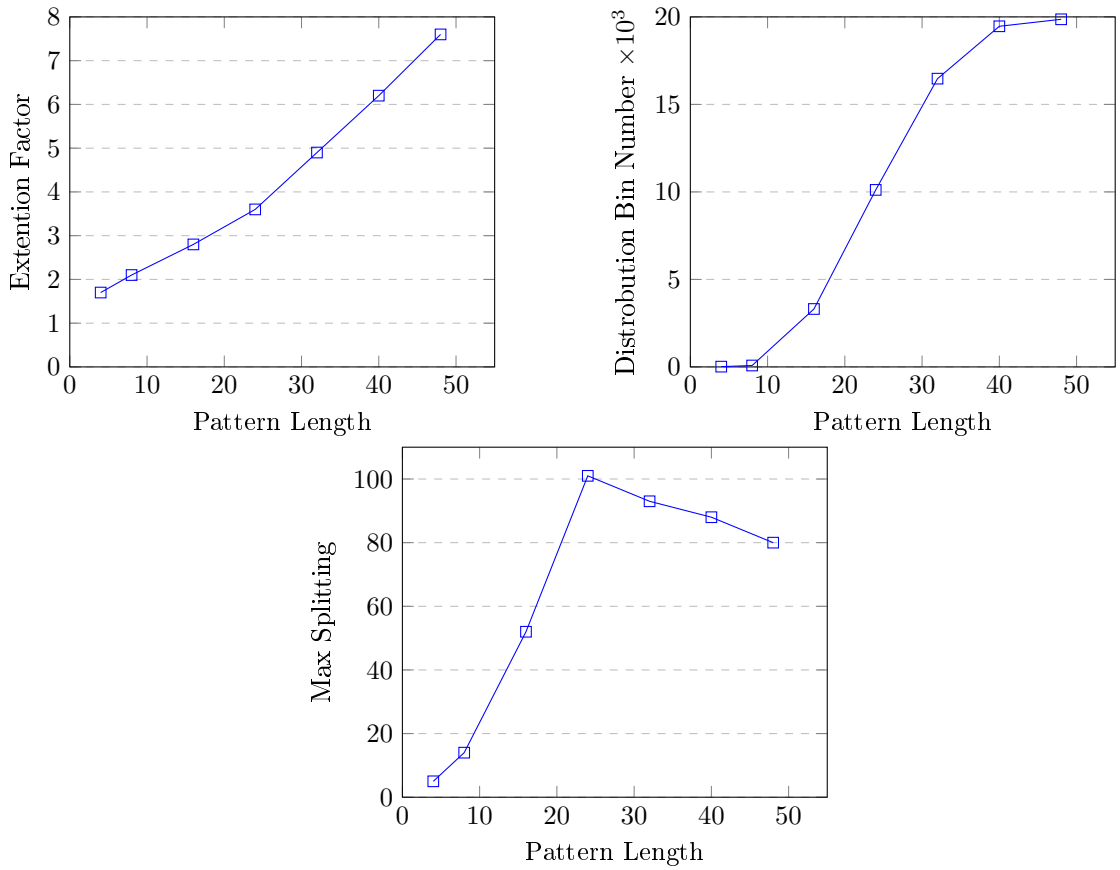


Figure 9.1: Pattern Length statistics on sample size 10000

the byte length, looking at table 9.1, it is clear not all character as created equal, since only a maximum of 89 different byte patterns is seen. In a purely random sample, you would expect to see the number of bins used to increase exponentially, however, the graph shows this is not the case as the rate of increase slows. This could be since as the pattern length increases, the number of patterns in the sample that it has to draw from decreases. Also, the variation between histogram bin sizes widens since smaller patterns used in different permutations produce even spreading throughout the histogram even if the larger patterns the permutations make up are very uneven. To give concrete examples, let us take the examples of length 4 and 40. Length 4 produces 16 bins, of which the maximum proportion is 0.2191, whereas for length 40 with 105573 bins has a maximum proportion of a smaller absolute value 0.0024. However, when we interpret these values as ratios from uniform we get,

$$0.2191/\frac{1}{16} = 0.2191 \cdot 16 = 3.5056$$

and

$$0.00245/\frac{1}{105573} = 0.00245 \cdot 105573 = 258.65385$$

which shows the much larger disparity in the larger pattern size bins. In the case of length 40, the particular value corresponds to the string `['/', 'i', 'm', 'a', 'g']`, which as expected, is semantically meaningful as an extremely common string in today's media-rich web experiences. Compare that to the low-frequency strings such as `['V', 'U', 'S', '5', 'W']`, which are probably only appearing in cryptographic random identifiers that sites sometimes use to pass tokens from one entity to another like in the Oauth2 protocol. For the 4 bit case, the most common result is `0x6`, which also is expected since the `ascii` representation of the letters `[a-o]` are in the range $0x60 \leq x < 0x70$. To sum up, the smaller the patterns lead to more independence and so more even distributions, but this makes them easier to forge as they are closer to uniform.

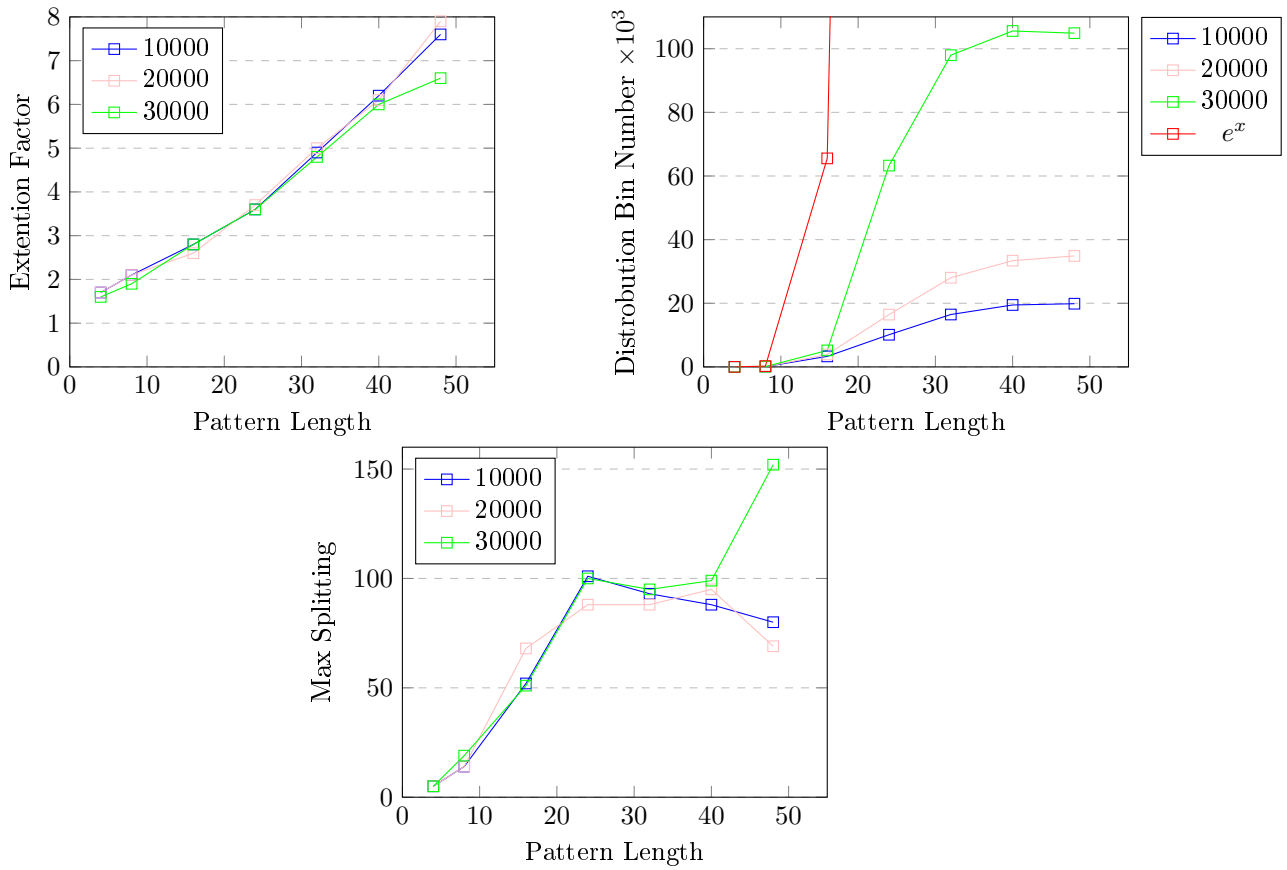


Figure 9.2: Pattern Length statistics on sample sizes $\{10000, 20000, 30000\}$

Larger Samples

To investigate the effect of the reference sample size on these figures, I extracted more network traffic so I could compare 3 different-sized datasets. The figure on histogram-bin-numbers shows how, although each sample size does have diminishing returns, as pattern length increases, the maximum value reached does improve directly with the size of the sample. This is because although at small pattern sizes the samples are exhaustive, on larger patterns although the actual normal distribution may contain up to an exponential number of sequences in a fixed sample we only ever see a small portion of them.

I believe the largest sample produces such a large increase compared to the step from 10000 to 20000 due to the way the web crawler worked. As described in the implementation section, it starts with a small set of sites and ventures outwards by following links on the pages. Because of this, the number of sites increases but also the diversity of these sites increases in language, style and technologies (different web frameworks produce different request patterns).

On larger samples, the possible encoding output expansion is slightly less for larger pattern length than when using a smaller one. This is due to the increase in the possible number of divisions of the distribution down to the larger number of bins giving a more evenly divided distribution.

Summary

In summary, for all sample sizes, shorter patterns produce lower ciphertext expansion overheads, in fact, the relationship is linear. However, we have seen that longer pattern lengths do produce more context-aware characters and so lead to higher quality distributions. Therefore, there is an application specific trade-off that deployers of the scheme must consider. Also, as expected, larger samples produce a higher fidelity distribution. This does increase the space requirement of the scheme, but time performance should not be affected to the same extent due to the sublinear complexity of the binary search.

9.2 Statistical Indistinguishability

Test	Value	Result
Chi-square	0	good
G-test	0	good
Kullback-Leibler	0	good
Jensen-Shannon	0	good
Fisher's exact test	0	good

Chapter 10

Conclusion

A compulsory chapter, of roughly 5 pages

The concluding chapter of a dissertation is often underutilised because it is too often left too close to the deadline: it is important to allocation enough attention. Ideally, the chapter will consist of three parts:

1. (Re)summarise the main contributions and achievements, in essence summing up the content.
2. Clearly state the current project status (e.g., "X is working, Y is not") and evaluate what has been achieved with respect to the initial aims and objectives (e.g., "I completed aim X outlined previously, the evidence for this is within Chapter Y"). There is no problem including aims which were not completed, but it is important to evaluate and/or justify why this is the case.
3. Outline any open problems or future plans. Rather than treat this only as an exercise in what you *could* have done given more time, try to focus on any unexplored options or interesting outcomes (e.g., "my experiment for X gave counter-intuitive results, this could be because Y and would form an interesting area for further study" or "users found feature Z of my software difficult to use, which is obvious in hindsight but not during at design stage; to resolve this, I could clearly apply the technique of Smith [7]").

Bibliography

[1] Alexa. The top 500 sites on the web. https://www.alexa.com/topsites. Date accessed: 20180-04-20.

[2] Internet Archive. Wayback machine online tool. https://archive.org/web/.

[3] Daniel Arp, Fabian Yamaguchi, and Konrad Rieck. Torben: A practical side-channel attack for deanonymizing tor communication. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, pages 597–602, 2015.

[4] David Bamman, Brendan O'Connor, and Noah Smith. Censorship and deletion practices in chinese social media. *First Monday*, 17(3), 2012.

[5] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. *Format-Preserving Encryption*, pages 295–312. Springer Berlin Heidelberg, 2009.

[6] J. Black and P. Rogaway. *Ciphers with arbitrary finite domains*, pages 114–130. Springer, 2002.

[7] M. Brightwell and H. Smith. *Using Datatype-Preserving Encryption to Enhance Data Warehouse Security*, pages 141–149. NIST, 1997.

[8] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 605–616. ACM, 2012.

[9] Richard Clayton, Steven J Murdoch, and Robert NM Watson. Ignoring the great firewall of china. In *International Workshop on Privacy Enhancing Technologies*, pages 20–35. Springer, 2006.

[10] European Commission. Can i ask a company to delete my personal data? https://ec.europa.eu/info/law/law-topic/data-protection/reform/rights-citizens/my-rights/can-i-ask-company-delete-my-personal-data_en. date access: 2018-04-25.

[11] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 61–72. ACM, 2013.

[12] Benjamin Edelman and Jonathan Zittrain. Empirical analysis of internet filtering in china. *Berkman Center for Internet & Society, Harvard Law School, http://cyber. law. harvard. edu/filtering/china*, 2005.

[13] Undisclosed employee at the NSA. Tor stinks presentation. https://edwardsnowden.com/wp-content/uploads/2013/10/tor-stinks-presentation.pdf, 2012. date access: 2018-04-25.

[14] security Financial Services. *Banking – Procedures for message encipherment*. International Organization for Standardization.

[15] M. Goldberg, A. Sipser. *Compression and Ranking*, pages 440–448. ACM Press, 1985.

[16] United Kingdom Government. Uk defamation act 2013. http://www.legislation.gov.uk/ukpga/2013/26. date access: 2018-04-25.

[17] Felix Horger, Tobias Würfl, Vincent Christlein, and Andreas Maier. Deep Learning for Sampling from Arbitrary Probability Distributions, 2018. arXiv preprint.

[18] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The parrot is dead: Observing unobservable network communications. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, pages 65–79. IEEE Computer Society, 2013.

[19] Ari Juels and Thomas Ristenpart. Honey encryption: Security beyond the brute-force bound. In *Advances in Cryptology – EUROCRYPT 2014*, pages 293–310. Springer Berlin Heidelberg, 2014.

[20] Tim Berners Lee. Establish web's principles of openness and privacy. `https://www.theguardian.com/technology/video/2014/mar/12/world-wide-web-tim-berners-lee-principles-openness-privacy-video`. date access: 2018-04-25.

[21] E. Mäkinen. Ranking and unranking left szilard languages. Master's thesis, Department of Computer Science, University of Tampere, 1997.

[22] Rishab Nithyanand, Oleksii Starov, Adva Zair, Phillipa Gill, and Michael Schapira. Measuring and mitigating as-level adversaries against tor. *CoRR*, abs/1505.05173, 2015.

[23] European Court of Human Rights. Fact sheet - hate speech. `https://www.echr.coe.int/Documents/FS_Hate_speech_ENG.pdf`. date access: 2018-04-25.

[24] National Bureau of Standards. Guidelines for implementing and using the nbs data encryption standard. *FIPS PUB 74*, April 1, 1981.

[25] C. Perkins. Ip mobility support for ipv4. Technical report, The Internet Society, 2002. `https://tools.ietf.org/pdf/rfc3344.pdf`.

[26] The TOR Project. obfs4 transport evaluation. `https://trac.torproject.org/projects/tor/wiki/doc/PluggableTransports/Obfs4Evaluation`, 2015. date access: 2018-04-25.

[27] Phillip Rogaway. Nonce-based symmetric encryption. In *Fast Software Encryption*, pages 348–358. Springer Berlin Heidelberg, 2004.

[28] R. Leibler S. Kullback. On information and sufficiency. *The Annals of Mathematical Statistics*, 1951.

[29] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Computer Security – ESORICS 2006*, pages 18–33. Springer Berlin Heidelberg, 2006.

[30] Voltage. Hewlett packard enterprise format-preserving encryption. `https://www.voltage.com/technology/data-encryption/hpe-format-preserving-encryption`. Accessed on 2018-04-20.

[31] C. G. Nevill-Manning I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Artificial Intelligence Research*, 7, 1997. site: `http://www.sequitur.info/`.

[32] Chairman: James Zimmerman. Business climate survey. Technical report, AmCham China, 2016.

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the marking panel is not obliged to read such appendices.