Covert Communications Despite Traffic Data Retention

George Danezis

Microsoft Research,
Cambridge, UK
gdane@microsoft.com

Abstract. We show that Alice and Bob can communicate covertly and anonymously, despite Eve having access to the traffic data of most machines on the Internet. Our protocols take advantage of small amounts of shared state that exist in many TCP/IP stacks, and use them to construct a covert channel. Techniques inspired from Direct Sequence Spread Spectrum (DSSS) are used to make sure that the communication is covert and resistant to noise. We implement a prototype based on ICMP Echo (ping) to illustrate the practicality of our approach and discuss how a more complex protocol would modulate information through the use of TCP features to make communication detection very difficult. The feasibility of covert communications despite stringent traffic data retention, has far reaching policy consequences.

1 Introduction

This work contributes to the understanding of covert communications on deployed networks such as the Internet. We show that if any shared state can be accessed and influenced by two parties they can use it to communicate indirectly, making it hard for observers to correlate senders and receivers of messages. We also present a very common feature of the IP protocol [28, 27], based on the IPID packet field, that can be used to implement such covert communications. As a result our scheme does not require a dedicated infrastructure (as mix networks do), but uses any of the large number of deployed machines to relay messages.

We further show that the 'noise' produced by other, innocuous users, can be used to enhance covertness – given the observer does not know the shared key it becomes difficult to assess whether there is a communication at all. To achieve this we are inspired by techniques close to DSSS, that allow for low power signals to be hidden and uncovered from high noise environment. Finally we note that our scheme allows for covert communication despite, even stringent, data retention. This is partly due to the low level mechanisms we rely on (raw IP packets) and the very low signal power that would require prolonged, very costly, observation to allow the identification of a communication.

We first introduce in Section 3 the requirements of a cover communication systems, and discuss why established technologies only partially satisfy them. In Section 4 we present the basic TCP/IP mechanisms on which we shall build two

systems: a basic one based on ICMP Echo requests (Section 4.2) and a second, more covert one, based on TCP circuits (Section 4.3). We discuss extensions and open issues in Section 5 and present our conclusions in Section 6.

2 Background and Related Work

Covert and jamming resistant communications are a well studied discipline in the field of military and civilian radio communications. *Low probability of intercept and position fix* techniques like frequency hopping and Direct Sequence Spread Spectrum (DSSS) have been developed to force an adversary to spend a lot of power to jam a signal, as well as to hide altogether the existence of a communication from those that do not know a shared key [5]. Such technologies have been deployed in military tactical radios, but have also become part of civilian communications with frequency hopping being used in GSM phones, and CDMA (a variant of DSSS that uses orthogonal codes) being used in mobile communications and high-speed modems.

Yet relatively little attention has been directly payed to the *covertness* of communication in the context of the Internet. The field of anonymous communications, as started by David Chaum's [13] proposal for mixes and mix networks, attempts to provide unlinkability of senders and receiver. These anonymity properties fall short of full covertness, in that an observer is in a position to determine that some form of communication is taking place. Jamming resistance is also difficult to achieve, since the anonymous communication infrastructure in deployed systems [14, 23, 15], can easily be targeted and rendered inoperable by a powerful adversary. A peer-to-peer approach [18, 29] to providing anonymity may change this, but so far no such system was found to provide strong anonymity properties.

Steganography [6], the embedding of ciphertext into innocuous data, also provides some form of covertness. An adversary observing a communication cannot determine its content with certainty, and messages can be transferred under the cover of 'normal' traffic. Yet steganography does not hide the acts of communication themselves, or the communicating parties. Therefore traffic analysis techniques that map social structures [30, 21] to extract information would still be able to uncover information. Such techniques often ignore content and are unlikely (in the absence of cover traffic – which would bring us back to anonymous communications) to be affected by steganographic techniques.

Despite the little attention payed to covertness properties, traceability of communications has become a policy hot topic. National legislatures, often after terrorist incidents, have imposed 'traffic data retention' requirements on the telecommunications and Internet service provider industries [12, 20, 24], forcing them to log call, information access and location data (not content). At a European level EU Directive 2002/58/EC [3] (Directive on Privacy and Electronic Communications) and its December 2005 amendment [4] respectively allowing and making retention mandatory, replaced Dir. 1995/46/EC [1] (Data Protection Directive) and Dir. 97/66/EC [2] (Telecommunications Privacy Directive)

that prohibited such practices. The granularity of the retained data is variable, and the directives and laws often refer to communications in an abstract manner to allow for technology independence. As a rule of thumb for this work we shall assume that everything that is routinely logged in deployed systems shall be available for inspection. This requirement is much more stringent than the most draconian data retention schemes proposed, that usually only require logging high (application) level communication events and user identification events (when the user is authenticating to an ISP). Relaxing the attacker models would make covert communication more efficient, yet the principles to achieve a secure scheme would be the same as presented in this paper.

There exist other, simpler, approaches to circumvent traffic data retention and achieve covert communications in practice. The simplest approach would be to use one of the many open relays documented in the SORBS list, for anti-spam purposes. These include SMTP (email) and SOCKS (any TCP stream) relays that would allow two parties to get in contact and talk. Another more ambitious solution would be to establish a bot-net, composed of many compromised machines, and deploy a parallel communication infrastructure that does not log anything. These solutions rely on the assumption that the relays are not observed by the adversary, which is most probably true. The solutions we propose on the other hand allow covert communication even when under some forms of surveillance. In this sense our techniques take advantage of the fundamental limits of traceability versus covertness, and raise significantly the cost of surveillance.

3 Covert Communication Requirements

Alice and Bob would like to communicate without Eve, the adversary, being able to observe them. They share a symmetric key K, unknown to Eve, and can use established cryptography techniques to protect the secrecy and integrity of exchanged messages. In addition to this they would like the mere act of communication to be unobservable to Eve: Eve should not learn that Alice or Bob are communicating with each other, or engaging in an act of covert communication.

Hiding the fact that Alice and Bob are communicating with each other could be achieved using anonymous communication protocols [13, 23, 14, 15]. Yet these protocols (like encryption itself) are very easy to detect, therefore jeopardising covertness. They use standard handshakes, fixed message sizes and formats, a more or less fixed and public infrastructure. As a result, it is easy for Eve to determine that Alice and Bob (along with many others) are taking part in an anonymous communication protocol – which in many cases would give rise to suspicion. Due to their dependence on mixing infrastructure such systems may also be prone to legal compulsion (to log or reveal keys), targeted denial of service attacks or blocking.

The straight forward composition of steganography and anonymous communications comes also short of providing both anonymity and covertness. A message, that possibly contains steganographic embedded information, that is

transported anonymously is already very suspicious, and a clear indication that the sender and the receiver (although not linked) are taking part in some covert communication. On the other hand a mere steganographic message might provide covertness of content, in that the true message is not revealed to Eve, but also provides a clear link between Alice and Bob.

We therefore propose that covert communication mechanisms should have certain characteristics.

Definition: *A covert communication system has to make use of unintended features of commonly used protocols, in a way that does not arise suspicion, in order to unobservably relay messages between two users.*

The use of common communication protocols is essential in not arousing suspicion, since any deviation from the norm may indicate an act of covert communication. The challenge is to find generic enough features of common protocols that allows messages to be relayed through third party machines. Any direct communication between Alice and Bob would create a link between them, that may in the eyes of Eve contain a covert channel or steganographicaly embedded information. On the other hand the use of an intended communication channel provided by a third party can be subject to logging and interception. As a result the only option for implementing covert communications is to use unintended features that allow relaying of messages. Furthermore these features should be exploitable without giving rise to suspicion to an observer (which again would jeopardize covertness).

Given all these requirements it is surprising that such features, not only exist in deployed communication protocols, but they are abundant.

The security of any covert communication scheme is dependent on the observation capabilities of the adversary. We wish to mostly consider an adversary that observes the world through retained traffic data. Furthermore, we would ideally want to provide security against a global passive observer, that has access to any information transiting on the network. We present a spectrum of systems, protecting Alice and Bob from an Eve with increasing surveillance capabilities. As we expect the more we bound and reduce Eve's capabilities the more efficient our systems can be, while still remaining covert.

There are also inherent advantages to finding and exploiting low level network mechanisms to provide covert communications. First low level mechanisms are likely to be used in a variety of ways, depending on the protocols that are stacked on them. This adds variance to the network behavior that would allow communications to be more effectively hidden. Secondly, low level mechanisms are also more abundant – more machines run vanilla TCP/IP than a particular version of a web-service. This allows for more choice when it comes to finding a relay, which in turn increases the cost of an adversary that has to observe all potential hosts for communication. Finally low level protocols produce high granularity traffic data, the storage of which is orders of magnitude more costly than storing high level network events – compare the cost of storing web access logs versus the cost of storing the header of every single IP packet traversing a network.

In the next sections we concentrate on a particular feature of many Internet Protocol (IP) implementations, namely sequential IPID values, that is low level and exhibits all the necessary characteristics to facilitate covert communications.

4 A Covert Communications System

Our key contribution is to show that there is a ubiquitous feature of deployed IP networks that allows for covert communication. The Internet is a collection of networks that 'talk' the same Internet Protocol (IP) [27] to exchange packets containing information. Each packet starts with a header that contains routing information, but also a special identification IPID field. The IPID field is 16 bits long, and is used to detect duplicate packets and perform fragmentation and reassembly of IP packets in the network. The creator of the IP packet sets its identification field to "a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the Internet system." [27]

Many deployed operating systems and TCP/IP stacks use a simple counter to set the value of the IPID field on outgoing packets. This feature has been used in the past to perform security sensitive monitoring in a manner of ways. Steven Bellovin uses the serial nature of the IPID field to monitor the number of different machines behind a Network Address Translation (NAT) gateway [10]. The IPID can be determined either by a global or a 'per-host' counter. The availability of some machines with global counter makes possible a techniques known as 'idle scan' or 'dump scan' [8], that determines which TCP [28] ports a machine is listening to, without sending any direct traffic to it. This technique is implemented in the Nmap [19] network scanner. Applications of serial IPID fields to remote monitoring and traffic analysis have also been proposed [7, 9, 25].

We are going to use the serial nature of IPID fields of many Internet connected computers in order to allow for covert communications. We explain how to implement covert communications using an intermediary that uses a global IPID counter.

Alice wants to talk to Bob, with whom she shares a key K, over an intermediary called Charlie. Charlie implements an IP stack that selects IPID values using a global counter. Note that if Alice an Bob can force Charlie to emit packets, and if they are able to observe any packet from charlie they will be able to communicate. More concretely, Alice will at each time $2t_i$ force Charlie to emit n packets, while Bob will observe a packet from Charlie at times $2t_i + 1$ to retrieve n. The number of packets n is the information that has been transferred between Alice and Bob. By repeating this process Alice can transmit to Bob arbitrary messages.

The first question that arises is: how can Alice and Bob force Charlie to emit packets, and receive packets from him. We shall present two ways in which this is possible based on ICMP Echo [26] and TCP [28], in subsections 4.2 and 4.3.

A second worry is that Charlie will also be generating traffic with third parties, incrementing the IPID counter, and adding noise to the observation of Bob. We note that this is a great opportunity for cover traffic: if Alice and Bob were the only parties that Bob would be receiving and sending information to, they may be linked easily by an observer. On the other hand if Charlie is engaging in multiple conversation, including with Alice and Bob, it is difficult for even a direct observer to establish who may be communicating with whom. Furthermore we shall make it difficult for other clients to establish that there is any signal in the IPID data, by using the shared key K to allow Alice and Bob to communicate over that noisy channel.

4.1 Transmission over a noisy IPID counter

Assume that Alice and Bob want to communicate the binary symbols $n_0 = 0$ or $n_1 = 1$, over the channel. They use their secret key K in order to produce two psedo-random traffic patterns v_0 and v_1 of length l corresponding to each symbols n_0 and n_1 respectively:

$$v_{0i} = H(0, i, K), \forall i \in [0, l-1] \tag{1}$$

$$v_{1i} = H(1, i, K), \forall i \in [0, l-1] \tag{2}$$

We assume that H is a good hash function that takes bit strings and produces uniform values in the interval $[0, 2\mu]$. As a result each symbol is mapped into a traffic pattern, which is a sequence of l values in the interval $[0, 2\mu]$[1]. Alice sends in each round the number of packets specified in the sequence of the symbol she wishes to emit one value at each time period time. For example to transmit the string '0110', the sequence v_0, v_1, v_1, v_0 should transmitted, which would take $4 \cdot l$ time periods.

Bob observes packets from Charlie with IPID increments, from one time period to the next, of u_i for $i \in [0, l-1]$. How does Bob determine the symbol sent by Alice? Based on the knowledge of K, Bob can construct a filter to determine if the traffic pattern v_0 or v_1 is embedded in the noise. To differentiate between the two symbols Bob calculates the values r_0 and r_1, for each candidate symbol:

$$r_j = \sum_{i \in [0, l-1]} v_{ji} u_i, \quad j \in \{0, 1\} \tag{3}$$

The difference between the value of r associated with the correct symbol, versus the value of r associated with other symbols grows linearly with the length of l. It can be shown (full derivation in Appendix A) that, if the selection

[1] We will see that the hash function H should produce values indistinguishable from any distribution D, that is a good model of 'typical' traffic (given the information known to the adversary). The uniform distribution, used as an example here, is not such a distribution, and is therefore insecure against an adversary that logs ICMP Echo packets and events. Such events are considered too low level to be subject to retention at the moment.

of traffic levels v follows a probability distribution D (in our example the uniform distribution $D = U(0, 2\mu)$), this difference is:

$$\mathbb{E}(\Delta r) = \mathbb{E}\left(r_{\text{correct}} - r_{\text{incorrect}}\right) = l \cdot \mathbb{V}(D) \tag{4}$$

The function $\mathbb{V}$ denotes the variance of the distribution D.

It is therefore clear that, if the key K is known, Bob can reconstruct the appropriate traffic patterns v to extract the correct symbols from the IPID in the long run, despite any noise. Furthermore by increasing the length l of the traffic pattern we can afford to keep the additional traffic injected by Alice low and make it difficult for an observer to detect that any communication is taking place.

Our results hold for any distribution D, and therefore we are also free to use a traffic distribution that looks realistic i.e. that mimics the characteristics of some type of innocuous traffic. In fact the covertness our this scheme depends on the adversary's ability to distinguish between the distribution D used and 'normal' traffic, not containing any covert information.

4.2 An ICMP Echo realization

We have established that if Alice can force Charlie to emit any packets, and Bob can receive any packets from Charlie, Alice and Bob can communicate through Charlie using information encoded in the IPID field. The simplest way for Alice and Bob of achieving this is using the ICMP Echo [26] protocol, often referred to as 'ping', that must be implemented by a compliant TCP/IP stack (although some firewalls block it). ICMP Echo allows a host to send a packet to a destination address, which in turn echos it back to the original sender. Alice can therefore send 'ping' messages to force Charlie to increment his counter since responding increases the counter by one. Bob can use the same facility to receive messages from Charlie and determine the state of his IPID field.

This simple minded approach provides surprisingly good results, yet has some security shortcomings as we shall see. Figures 1, 2 and 3 illustrate a single run of our prototype in a low noise environment. For this experiment we used 30 second long traffic patterns of length 30 (which indicates a time interval of at which Bob must observe the counter of one second) from a uniform distribution $U(0, 100)$, to transmit one symbol out of an 8 bit alphabet.

We first collect the data sent by Alice (figure 1). This data is likely to contain some low frequency noise, that can be filtered out, since it is not likely to contain any useful information. To eliminate its effects we calculate the predictors r using a randomly generated traffic pattern, and use this as the baseline for detection (this is equivalent to subtracting from r_{correct} a random $r_{\text{incorrect}}$ providing us the result we expected). The values of $r_{\text{incorrect}}$ for all times are shown in figure 2. Note some patterns emerging, that are due to the traffic patterns not being orthogonal. These might represent a security problem since they leak the message content and their regularity would leak the existence of a message. We shall discuss how to avoid them in the discussion section.

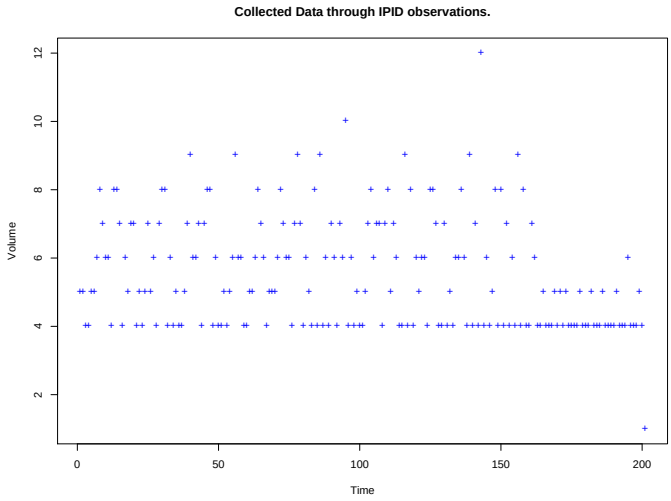

Fig. 1. Raw data as collected using the ICMP Echo method.

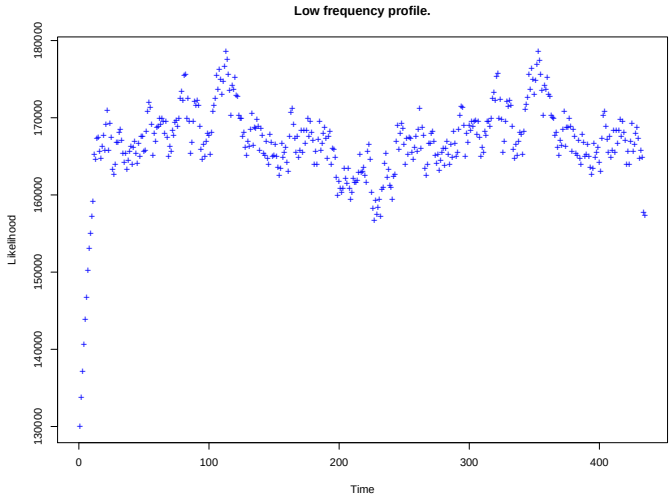

Fig. 2. The low frequency profile of the collected data.

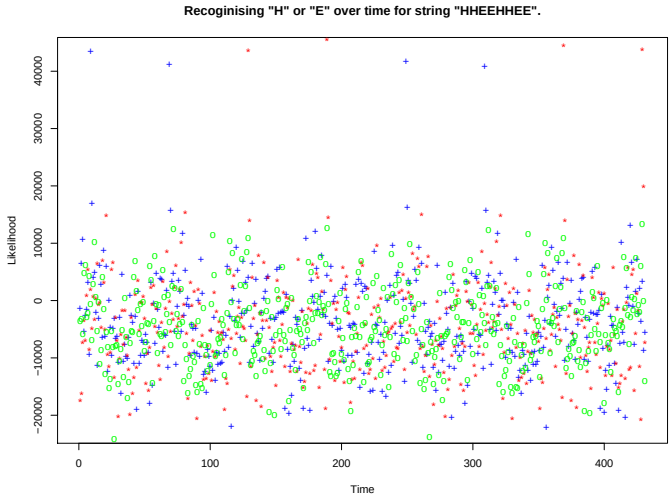

Fig. 3. Recognition of two different symbols.

Finally we calculate values of r for three candidate symbols in figure 3. The value of r for 'H', 'E' and 'A' is denoted by '+', '*' and 'o' respectively. A spike detection filter is also applied. The transmitted string can easily be extracted by choosing the symbol with the highest peak at a 30 second interval. Furthermore we can see that there is little danger of losing synchronization, as long as the difference between a correct and an incorrect symbol is large enough.

The key drawback of the ICMP Echo based technique is that large volumes of ICMP traffic from legitimate users is not common. Such traffic is often the precursor of an attack, and indicative of hostile intentions. As a result standard intrusion detection systems, such as SNORT [32] log information about high rate of ping packets. To keep under the radar of such detection systems we would need to limit ourselves to the transmission of a very low volume of ping packets in time. As a result the variance of the distribution D would be lower, and the rate at which we could transmit and correct for noise would be greatly reduced.

As a proof-of-concept ICMP Echo shows we can engineer covert communications using deployed mechanisms. Yet triggering intrusion detection systems, let alone provoking logging, is not compatible with our requirements for covertness, and the low rates that Alice would have to suffer to evade detection force us to look for a different solution.

4.3 A TCP based realization

The Transmission Control Protocol (TCP) [28], provides multiplexed, reliable and bidirectional stream communication between two Internet hosts. A session is established between two hosts using a 3 way handshake, and then further

data can be exchanged in both directions between the hosts. TCP also provides facilities for rate and congestion control, that we shall make use to provide covert communications.

Two key concepts in TCP congestion control are *acknowledgments* and *windows*. Each TCP packet contains a serial number, and an acknowledgment number. The acknowledgment number is set by the sender to be the serial number of the last TCP packet received, which is part of a continuous sequence from the beginning of the transmission. Conceptually this means that all previous packets, with smaller sequence number, have already been received. Packets that are not acknowledged are re-sent at intervals according to some set algorithms [11, 17] (with exponential increase of the delay and linear reduction, to slow down when there is congestion).

Each host also provides a hint about the amount of data it can hold in its buffers at any time, which is called the window size, also included in each TCP packet sent. The window size indicates the maximum number of unacknowledged bytes that can be sent to that host. Using this mechanism the receiver has control over the rate at which data is reaching him or her.

Alice and Bob, that want to communicate covertly, can use the congestion control features of TCP to modulate a global IPID counter. To do this Alice establishes a TCP session with a third party, Charlie (that implements an IP stack with serial IPID values), and so does Bob. An HTTP (web) request would be perfectly adequate. During the setup of the TCP connection they both negotiate a suitably small maximal payload size (using the Maximum Segment Size option in TCP) to ensure that even if small amounts of data are transmitted many IP packets are generated. Alice can control the rate at which the intermediary's IPID counter is increased by modulating the window size, and by only acknowledging packets when more packet transmission is desirable. As a result Alice can lead Charlie to transmit a set number of packets pet unit time, and increase the IPID field by the amount dictated by the traffic pattern of the codeword she wishes to transmit. Bob on his side keeps the windows very small, and only acknowledges a packet at a time, forcing Charlie to only send one packet per unit time. This allows Bob to read Charlie's IPID counter contained in the TCP packet, without adding too much noise, and recovering the codeword embedded by Alice.

It is important to note that, even genuine, TCP traffic has quite a large variance, and as a result the information encoded by Alice can be extracted by Bob, despite shorter keywords and higher levels of traffic, without compromising covertness. The degree to which the TCP traffic characteristics have to perfectly match a typical TCP connection depends on the level of surveillance expected. In case each and every packet is logged, it would be important to stick to the degrees of freedom provided by standard TCP congestion control algorithms that regulate traffic. This should make cover traffic indistinguishable from 'normal' traffic, but would reduce the bandwidth of the channel – the only parameters of the traffic distribution that Alice could control are the random back-offs, simulated congestion in links, full buffers, etc. On the other hand if we only

expect the connection establishment to be logged, and maybe even the content of the stream, but not the packets themselves, Alice can modulate at will all the window, acknowledgment and Maximum Segment Size parameters to maximize the bandwidth of the channel.

5 Evaluation and Discussion

So far we have provided an overall framework within which Alice and Bob can communicate covertly if they can modulate and read a shared counter. Yet, as for most real-world security systems, the devil is in the details, and a lot of details have to be carefully considered before such systems can be considered secure.

5.1 Auto-correlation and synchronization

The first problem with our simple-minded traffic pattern design is illustrated in figure 2, where an adversary can observe a traffic pattern forming (the different parts of the message look the same). The reason for this is that we use the same traffic pattern to transmit the same symbol. As a result an adversary auto-correlating he traffic volume should be able to extract the full traffic code book, and recover (or at least detect) signal transmitted. The solution to this is to never use the same traffic pattern again. To do this we can include in the generation of the traffic pattern the time, or sequence number of the symbol (denoted t), and include this in the random generation of the traffic pattern for each symbol:

$$v_{0it} = H(0, i, t, K), \forall i \in [0, l-1] \tag{5}$$

$$v_{1it} = H(1, i, t, K), \forall i \in [0, l-1] \tag{6}$$

This means that 0s and 1s will be represented with different traffic patterns according to the time, or their position in the ciphertext.

The new approach for generating traffic patterns to encode symbols is secure, but imposes an additional requirement on Alice and Bob to have some way of synchronizing their clocks or their transmission. Off-the-shelf technology, like GPS, can make this easier, and even cruder Network Time (NTP) based protocols should be able to provide an appropriate time resolution to synchronize the traffic pattern code books. The design of self-synchronizing yet secure codes would be an interesting potential avenue of research, which is beyond the scope of this work.

5.2 Identification of Intermediate Hosts and Incentives

Alice and Bob need to find an intermediate host that implements its IPID using a global counter to be able to use our techniques. During our experiments we scanned our local sub-net (a /24 section of the global address space), and discovered 50 machines responding to ICMP Echo requests. Out of those about 30

used a counter to determine the IPID values of IP packet. About 5 of these used a global counter shared amongst all destinations, the others using only a per-destination counter. An estimate of one machine in ten exhibiting this feature gives hope that finding an appropriate host should not be too difficult.

The simplest approach would be for Alice and Bob to determine an appropriate host ahead of time, and use that for communication. This may not be possible, and they may need to determine a host 'on-the-fly'. A simple-minded approach would be for Alice and Bob to seed a random number generator using their shared key K and test random Internet hosts until they find the first one that exhibits the right characteristics. The number of hosts that they will have to try follows a geometric distribution, and if one in ten hosts is appropriate, then we expect about ten hosts to be tested before finding a good one.

Sadly the simple-minded approach described above is not very covert. In case the adversary controls even a small fraction of the Internet she will be able to observe two parties attempting to connect to the controlled hosts simultaneously. The probability this happens repeatedly becomes quickly very small (the probability of Alice and Bob both accessing l random hosts by chance becomes $\mathcal{O}(2^{-32\cdot(l-1)})$), and after even two observations the adversary can determine that Alice and Bob are trying to find a good relay to talk to each other. This is far from being merely a theoretical threat: large organizations control class A IP address spaces (including MIT and IBM) and large portions of unused address space is connected to Honey Nets [31] to detect automatic scanners – these real world entities and projects would most definitely detect Alice and Bob.

Strategies to avoid detection while identifying appropriate intermediaries would have to masquerade, once again, as legitimate traffic patterns. This might include a random query to a search engine for a relatively common term, and then using the shared key to select candidate hosts from the retrieved results. Alice and Bob selecting hosts using a random (but popular) walk over web-sites may also decrease the likelihood of suspicion or interception.

It is worth noting that unless a host in controlled by the adversary it has very few incentives to stop providing a service as an intermediary. No security properties of the intermediary host are affected at all by our scheme. Alice and Bob communicating, particularly under low noise conditions, is only imposing a very small burden (a few packets a second) – hardly noticeable for current networking infrastructures. Logging such activity in comparison would be much more expensive than bearing the cost of the transmission, and changing operating system or applying a patch that changes the IPID behavior would not be worth the inconvenience. As a result we do not expect this behavior to change any time soon.

5.3 Reducing Noise and Adaptive Codes

It is clear from our constructions that both Alice and Bob can affect Charlie's IPID counter, and they can both observe it. This can prove invaluable for Alice as she can determine the amount of noise present on Charlie and adjust the 'traffic strength' she uses to encode its symbols accordingly. This would involve

applying a set multiplicative factor to all the traffic patterns she induces so that they are still detectable despite the noise.

Since she is receiving feedback, to the same degree as Bob, she can also assess whether the pattern induced are easily detectable and vary their lengths accordingly. This approach favors covertness, since the traffic strength induced can be used by an adversary to detect the covert communication.

More efficient coding techniques may be developed to take into account all the information that Alice and Bob are aware off, that will be undoubtedly more efficient than our simple minded scheme. These are beyond the scope of this work. At the same time our scheme has the advantage that it allows for very simple interactions, where Alice induces the increase of Charlie's counter, and Bob only observes it, to be turned into a full covert communication medium.

5.4 One-Sided Covertness and Firewall Piercing

There is a body of literature concerned with censorship resistance [16, 22], and in particular communication across a filtering firewall, that has a particular type of covertness requirement. In this setting only one partner needs to remain hidden, the one inside the firewall, and has to acquire a small amount of information to communicate with the outside world. This information is usually a 'fresh' address for an anonymizing proxy through which further unfiltered communication is possible. This can be compared to a 'bootstrapping' problem for censorship resistant technologies.

We note that our approach would be extremely effective in providing such information through the firewall. Bob, who is inside the firewall, chooses hosts outside in a pseudo-random way, according to some pre-determined key, until an appropriate host is found to allow for covert communication. Then Alice sends a small message (about 32 bits) that is the fresh address of a proxy, that is not yet on the blacklist of the firewall. Bob retrieves the fresh address and can communicate further through the proxy.

In this scenario we can optimize considerably our algorithms without fear of compromise, since both Alice and Charlie are on the trusted side of the firewall, and not subject to surveillance. The advantage that the covert communication protocol offers to Alice is the ability to modulate the network address that Bob has to access, so that the firewall cannot block the initial communication.

5.5 High level events and counters

For most of this work we have concentrated on low level events, since they are unlikely to be the subject of logging and traffic data retention. Yet our techniques maintain some covertness despite observation and logging (as long as the traffic distribution that carries the covert message is indistinguishable from genuine traffic). We can therefore consider using high level protocols to communicate covertly.

The first approach is to use high level events to increment the IPID counter, instead of low level ICMP Echo packets or TCP features. In this case Alice and

Bob find a suitable Web Server, with a global counter determining the IPID, and simply perform a set of web requests, according to a common distribution sampled using a pseudo-random number generator seeded with their shared key. This will result in the IPID counter increasing, and (in the long run) information flowing from Alice to Bob.

A second possibility is to ignore all together low level counters such as IPIDs and only use high level counters such as counters measuring the number of accesses to particular web pages, that many web-sites incorporate. It is clear that Alice can influence the counter (by performing requests) and Bob can simply read it, and as a result covert communication is possible. Shared counters can also be found in abundance in on-line multi-player games. All the same algorithms for transmission and error correction would also apply to these cases.

6 Conclusions

We have shown that covert communications, that allow Alice and Bob to communicate indirectly and covertly are possible despite widespread traffic data, or even content retention. The bit rates we achieved easily with our prototypes are of the order of 16 bits a second, but can be effortlessly increased using more symbols of the same length. We expect a mature covert communication system to be able to carry a few hundred characters in a few seconds, an amount comparable to contemporary text messaging on mobile phones.

The covertness properties we provide are based on a key assumption: that Alice and Bob are able to generate traffic out of a distribution that looks realistic to the adversary. Very much like steganography and steganalysis relies on very good models of what images 'look like', it is likely that the field of covert communications on the Internet will have to spend more time studying traffic models, and finding efficient ways to tell apart real and synthetic traffic. Such models exist, in the network measurements literature, but have not been designed or used for such security purposes yet.

The model of the world of that adversary is crucially linked to the amount and kind of traffic data retained – the less data, the more uncertainty the adversary will have about the true distribution of the traffic, and higher rate covert communications are possible. If all data transiting in the network are available, then the inherent uncertainty of network traffic behavior can still be used to achieve low rate covert communications. Widening the traffic data to be retained would, of course, considerably raise the cost of the retention scheme.

Finally we can only hope that this study informs the debate about traffic data retention, as to its effectiveness in tracing determined adversaries that wish to communicate covertly. Many simple 'hacks' are possible to evade proposed retention, yet we have demonstrated that there are fundamental limits to the ability to trace, and well grounded ways to evade it. Widening the net of retention to detect those would require logging at the IP level, with limited success, which would make the policy even more expensive, for even lower returns in terms of intelligence product.

Acknowledgments

Many thanks to Nick Feamster for suggesting having a look at the IPID mechanisms in IP. Klaus Kursawe suggested using shared state in on-line games for covert communications. Richard Clayton has provided valuable early feedback.

References

1. Data protection directive (1995/46/EC). Official Journal of the European Communities, 1995.
2. The data protection telecommunications directive (1997/66/EC). Official Journal of the European Communities, 1997.
3. Directive on privacy and electronic communications (2002/58/EC). Official Journal of the European Communities, July 12 2002.
4. Final data retention directive (COM(2005) 438 Final). Commission of the European Communities, 2005.
5. R. Anderson. *Security engineering*. Wiley, 2001.
6. R. J. Anderson. Stretching the limits of steganography. In R. J. Anderson, editor, *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 39–48. Springer, 1996.
7. S. "antirez" Sanfilippo. about the ip header id. Personal communication, `http://www.kyuzz.org/antirez/papers/ipid.html`, December 14 1998.
8. S. "antirez" Sanfilippo. Dumbscan. Personal communication, `http://www.kyuzz.org/antirez/papers/dumbscan.html`, December 18 1998.
9. S. "antirez" Sanfilippo. How to learn firewalling relations using the ip id increment. Personal communication, `http://www.kyuzz.org/antirez/papers/moreipid.html`, November 1999.
10. S. M. Bellovin. A technique for counting natted hosts. In *Internet Measurement Workshop*, pages 267–272. ACM, 2002.
11. R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), Oct. 1989. Updated by RFC 1349.
12. E. P. I. Center. Data retention. On-line, `http://www.epic.org/privacy/intl/data_retention.html`, January 2006.
13. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
14. G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, 11-14 May 2003.
15. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
16. N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.
17. S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582 (Experimental), Apr. 1999. Obsoleted by RFC 3782.
18. M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In V. Atluri, editor, *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 193–206, Washington, DC, November 2002. ACM.

19. Fyodor. Nmap – free security scanner for network exploitation and security audit. http://www.insecure.org/nmap/.
20. P. International. Data retention. On-line, http://www.privacyinternational.org/index.shtml?cmd[342][]=c-1-Data+Retention&als[theme]=Data%20Retention&conds[1][category........]=Data%20Retention&als[_parent_]=Communication%20and%20electronic%20surveillance, January 2006.
21. P. Klerks. The network paradigm applied to criminal organisations. In *Connections 24(3)*, 2001.
22. S. Köpsell and U. Hilling. How to achieve blocking resistance for existing systems enabling anonymous web surfing. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004.
23. U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol version 2. Technical report, Network Working Group, May 25 2004. Internet-Draft.
24. S. Observatory. The surveillance of telecommunications in the eu. On-line, http://www.statewatch.org/eu-data-retention.htm, January 2006.
25. V. Paxson. About the ip header id. Personal communication, http://www.kyuzz.org/antirez/papers/ipid.html, December 15 1998.
26. J. Postel. Internet Control Message Protocol. RFC 792 (Standard), Sept. 1981. Updated by RFC 950.
27. J. Postel. Internet Protocol. RFC 791 (Standard), Sept. 1981. Updated by RFC 1349.
28. J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFC 3168.
29. M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.
30. M. K. Sparrow. The application of network analysis to criminal intelligence: An assessment of the prospects. In *Social Networks (13)*, 1991.
31. S. Sudaharan, S. Dhammalapati, S. Rai, and D. Wijesekera. Honeynet clusters as an early warning system for production networks. In E. Fernández-Medina, J. C. Hernández, and L. J. García, editors, *WOSIS*, pages 77–83. INSTICC Press, 2005.
32. S. team. Snort. http://www.snort.org/.

A Derivations

Here we prove that we can efficiently extract the symbols transmitted over a noisy IPID counter (equation 4). We have defined the values r as:

$$r_j = \sum_{i \in [0,l-1]} v_{ji} u_i, \quad j \in 0, 1 \tag{7}$$

The variable v denotes Bob's guess about the traffic pattern used by Alice, and the variable u the traffic pattern observed by Bob. We assume that Bob's observation contains some noise at each time $i \in [0, l-1]$ denoted n_i^*. So we can rewrite $u_i = s_i + n_i^*$, where s_i is the hidden traffic pattern actually used by Alice, and n_i^* is the noise.

We want to calculate how well r differentiates between correct symbols, and incorrect symbols. Therefore we define $r_{\text{correct}} = r_j$ if $v_j = s_i$ (meaning the equality of the traffic patterns), and $r_{\text{incorrect}} = r_j$ if $v_j \neq s_i$ (meaning the Independence of symbols at each time i).

We now calculate the expected value for r_{correct} (we use extensivelly the linearity of a distribution's expectation $\mathbb{E}$).

$$\mathbb{E}(r|n_i^*, \forall i.v_j i = s_i) = \mathbb{E}(\sum_{i \in [0,l-1]} v_{ji} u_i | n_i^*, \forall i.v_j i = s_i) \tag{8}$$

$$= \mathbb{E}(\sum_{i \in [0,l-1]} v_{ji}(s_i + n_i^*) | n_i^*, \forall i.v_j i = s_i) \tag{9}$$

$$= \sum_{i \in [0,l-1]} \mathbb{E}(v_{ji}(s_i + n_i^*) | n_i^*, \forall i.v_j i = s_i) \tag{10}$$

$$= l(\mathbb{E}(v_{ji} s_i | n_i^*, \forall i.v_j i = s_i) + \mathbb{E}(v_{ji} n_i^* | n_i^*, \forall i.v_j i = s_i)) \tag{11}$$

$$= l(\mathbb{E}(v_{ji}^2) + \mathbb{E}(v_{ji}) n_i^*) \tag{12}$$

Similarly for $r_{\text{incorrect}}$:

$$\mathbb{E}(r|n_i^*) = \sum_{i \in [0,l-1]} \mathbb{E}(v_{ji}(s_i + n_i^*) | n_i^*)) \tag{13}$$

$$= l(\mathbb{E}(v_{ji} s_i | n_i^*, \forall i.v_j i = s_i) + \mathbb{E}(v_{ji} n_i^* | n_i^*, \forall i.v_j i = s_i)) \tag{14}$$

$$= l(\mathbb{E}(v_{ji})^2 + \mathbb{E}(v_{ji}) n_i^*) \tag{15}$$

From the above two expressions we can show equation 4:

$$\mathbb{E}(\Delta r) = \mathbb{E}\left(r_{\text{correct}} - r_{\text{incorrect}}\right) \tag{16}$$

$$= l(\mathbb{E}(v_{ji}^2) + \mathbb{E}(v_{ji})\mathbb{E}(n_i^*)) - l(\mathbb{E}(v_{ji})^2 + \mathbb{E}(v_{ji})\mathbb{E}(n_i^*)) \tag{17}$$

$$= l(\mathbb{E}(v_{ji}^2) - \mathbb{E}(v_{ji})^2) \tag{18}$$

$$= l\mathbb{V}(v_{ji}) \tag{19}$$