3RD QUARTER 2007, VOLUME 9, NO. 3

IEEE COMMUNICATIONS SURVEYS
The Electronic Magazine of Original Peer-Reviewed Survey Articles

www.comsoc.org/pubs/surveys

A SURVEY OF COVERT CHANNELS AND COUNTERMEASURES IN COMPUTER NETWORK PROTOCOLS

SEBASTIAN ZANDER AND GRENVILLE ARMITAGE, AND PHILIP BRANCH,
SWINBURNE UNIVERSITY OF TECHNOLOGY MELBOURNE, AUSTRALIA

ABSTRACT

Covert channels are used for the secret transfer of information. Encryption only protects communication from being decoded by unauthorised parties, whereas covert channels aim to hide the very existence of the communication. Initially, covert channels were identified as a security threat on monolithic systems i.e. mainframes. More recently focus has shifted towards covert channels in computer network protocols. The huge amount of data and vast number of different protocols in the Internet seems ideal as a high-bandwidth vehicle for covert communication. This article is a survey of the existing techniques for creating covert channels in widely deployed network and application protocols. We also give an overview of common methods for their detection, elimination, and capacity limitation, required to improve security in future computer networks.

Often it is thought that the use of encryption is sufficient to secure communication. However, encryption only prevents unauthorised parties from decoding the communication. In many cases the simple existence of communication or changes in communication patterns, such as an increased message frequency, are enough to raise suspicion and reveal the onset of events. Covert channels aim to hide the very existence of the communication. Typically, covert channels use means of communication not normally intended to be used for communication, making them quite elusive.

Lampson introduced covert channels in 1973 in the context of monolithic systems as a mechanism by which a process at a high security level leaks information to a process at a low security level that would otherwise not have access to it [1]. While a serious threat even for single hosts, the potential for covert channels in computer networks is greatly increased. In computer networks overt channels, such as network protocols, are used as carriers for covert channels [2, 3].

The huge amount of data and vast number of different protocols in the Internet makes it ideal as a high-bandwidth vehicle for covert communications. The capacity of covert channels in computer networks has greatly increased because of new high-speed network technologies, and this trend is likely to continue. Even if only one bit per packet can be covertly transmitted, a large Internet site could lose 26GB of data annually [4].

Covert channels in computer network protocols are similar to techniques for hiding information in audio, visual or textual content (steganography). While steganography requires some form of content as cover, covert channels require some network protocol as carrier.

The ubiquitous presence of a small number of network protocols suitable as carriers (e.g. the Internet Protocol [5]) make covert channels widely available. They are usable even in situations where steganography cannot be applied. For example, the web-based techniques described later encode information sent from client to server as a covert channel, because normally web clients do not include any content in their requests.

Many applications of covert channels are of a malicious or unwanted nature, and therefore pose a serious threat to network security. Furthermore, we think that because of increased measures against *open channels*, such as the free transfer of memory sticks in and out of organisations as described in [6], the use of covert channels in computer networks will increase. Understanding existing covert channel techniques is crucial in developing countermeasures. The

detection, elimination, and capacity limitation of covert channels are challenging but need to be addressed to secure future computer networks.

This article is a survey of existing covert channels in network and application protocols and their countermeasures over the period 1987–2006. There are a number of related research areas we do not cover here for space reasons: covert channels in single hosts [1, 7], steganographic techniques for hiding information in content [8], and subliminal channels in cryptosystems [9].

Another related area is the field of anonymous communications concerned with obfuscating sender/receiver identities and their relationships (who is communicating with whom) [10]. However, this topic does not address full covertness of communication, since observers can still determine that some form of communication is taking place [11].

The rest of the article is organised as follows. First we describe potential application scenarios, define the terminology and explain the basic communication principles of covert channels. Then we present currently known covert channel techniques and discuss possible countermeasures. Finally, we conclude and identify future research.

Application Scenarios

A diverse range of individuals and groups has found reason to utilise covert channels for communication and coordination. Typically this is motivated by the existence of an adversarial relationship between two parties (such as government agencies versus criminal or terrorist organisations, hackers or corporate spies versus a company IT department, or dissenting citizens versus their governments).

Clearly, government agencies, criminals, or terrorist organisations have an interest to keep their communication secret. However, simply using encryption does not prevent adversaries from detecting communication patterns. Often only the evidence that communication takes place is sufficient to detect the onset of activity, discover organisational structures or justify obtaining police warrants.

Once spies or hackers have compromised computer systems they usually ex-filtrate data or instrument the systems for malicious purposes, including communication with installed Trojan horses (malicious programs disguised as or embedded within legitimate software) or tools for launching denial of service attacks. Such activities generate network traffic that — if not covert — would immediately alert system administrators, who then would discover the compromised systems. Exfiltrating sensitive data over covert channels does not even require compromised computers. It is sufficient if the attacker can compromise an input device such as a keyboard [12], or a software package such as a web browser [13].

It should be emphasized that often even *ordinary* employees may want to use covert channels to bypass their company firewalls in order to access Internet resources. Furthermore, recent attempts by some governments to limit the freedom of speech in the Internet have led to proposals for using covert channels to circumvent these measures [14, 15]. In countries that forbid (strong) encryption of data, covert channels can be used to *secure* the information transport (although this is not strong security in the cryptographic sense).

Network administrators can use covert channels to secure network management related communication by hiding it from hackers [16]. Again this is not strong security in the cryptographic sense. Honeypots, which are computer systems set up as trap for hackers, can also use covert channels to export logged data in real-time hidden from the attacker [17].

Computer viruses or worms can use covert channels to spread themselves undetected or for covertly exchanging information necessary for distributed processing (e.g. execute brute-force attacks on cryptosystems [18]).

Moskowitz *et al.* showed that imperfections in techniques for anonymous communication are effectively covert channels usable to thwart anonymisation [19]. Covert channels have been used for breaking anonymisation in specific network-related scenarios: Xu *et al.* described an attack on network traffic trace file anonymisation through covert channels [20] and Bethencourt *et al.* developed a covert-channel-based technique to identify the locations of sensors used for detecting malicious network traffic [21].

Covert channels can also be used for transmitting authentication data. A number of techniques have been developed for allowing authorised external users to access open firewall ports while presenting these ports as closed to all other users. One particular technique, called *port knocking*, uses covert channels for sending the authentication information [22]. Mazurczyk *et al.* proposed using covert channels and steganography to link control information, including authentication data, to the actual data flows [23, 24].

A number of researchers have developed packet traceback techniques using covert channels [25, 26]. Traceback techniques provide downstream nodes with information about the path of incoming packets. This is important in case of denial of service attacks, because it allows filtering the attack traffic at upstream nodes or even isolating the attacker(s).

Terminology and Communication Model

Terminology

Different terms have been used for describing the process of hiding information in network protocols. Whereas many researchers referred to covert channels some also used the terms steganography or information hiding. Partly this has been caused by differences between Lampson's original covert channel definition and a later definition by the US Department of Defense (US DoD). Lampson defined covert channels as "channels, […] not intended for information transfer at all" [1] whereas the US DoD defined covert channels as "[…] any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy" [27]. Another reason is that terminology has evolved over the years: the term "information hiding" simply had not been coined when the first covert channels in computer network protocols were proposed [28].

Throughout this article we use the term *covert channel* when we refer to the hiding of information in network protocols and refer to the information transmitted across the covert channel as *hidden* or *covert information*. Consistently with Petitcolas *et al.* we use the term *steganography* (literally meaning covered writing) when we refer to the hiding of information in content, and the term *information hiding* as a generic term for both [8]. Transmission of information through a system mechanism is an *overt channel* [29].

Traditionally covert channels were classified into storage and timing channels even though there is no fundamental distinction between them [27]:

- *Storage channels* involve the direct/indirect writing of object values by the sender and the direct/indirect reading of the object values by the receiver.
- *Timing channels* involve the sender signaling information by modulating the use of resources (e.g. CPU usage) over time such that the receiver can observe it and decode the information.

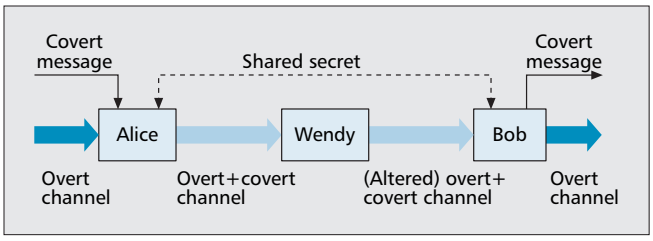

Figure 1. *The prisoner problem – the de-facto model for covert channel communication.*

The Prisoner Problem

The prisoner problem was first posed by Simmons and is the de-facto model for covert channel communication [30]. Two people, Alice and Bob,[1] are thrown into prison and intend to escape. To agree on an escape plan they need to communicate but Wendy the warden monitors all their messages. If Wendy finds any signs of suspicious messages she will place Alice and Bob into solitary confinement — making it impossible for them to escape. Alice and Bob must exchange innocuous messages containing hidden information that (hopefully) Wendy will not notice. Craver describes the different types of wardens [31]:

- A *passive* warden can only spy on the channel but cannot alter any messages.
- An *active* warden is able to slightly modify the messages, but without altering the semantic context.
- A *malicious* warden may alter the messages without impunity, but in reality malicious wardens are rare [31].

Handel *et al.* extended this scenario towards computer networks, where Alice and Bob use two networked computers for communication [32]. They run an innocuous looking overt communication channel between their computers, containing a hidden covert channel. Alice and Bob share a secret, which is useful for determining covert channel encoding parameters and encrypting/authenticating the hidden messages. For practical purposes Alice and Bob may well be the same person, for example a hacker ex-filtrating restricted information. Wendy manages the network and can monitor the passing traffic for covert channels or alter the passing traffic to eliminate or disrupt covert channels. Figure 1 depicts the model (Alice sending to Bob).

In the prisoner model Alice communicates with Bob, but in general covert channels are not restricted to unicast (one-to-one) channels. Alice could also send hidden information to Bob, Carol and Dave at the same time if the channel allows multicast (one-to-many) communication.

Communication Scenarios

There are a number of different scenarios for covert communication depending on whether Alice and Bob are the sender and receiver of the overt channel, or if they act as *middlemen* and manipulate an overt channel between innocent users [33].

If the sender of the covert channel is also the sender of the overt channel, it can manipulate the overt channel as desired (e.g. to maximise the covert channel capacity or its stealth). However, sometimes the covert sender may not be able to create overt channels or may choose not to do so for increased stealth. In this case the sender can act as middleman embedding a covert channel into an existing overt channel. Obviously, then the covert sender has no control of the overt channel, and the maximum capacity of the covert channel depends on the existing overt channel.

The covert receiver can be the receiver of the overt channel, but to increase stealth the receiver can also be a middleman extracting the hidden information from an overt communication destined for an innocent receiver. Then the covert receiver should (if possible) remove the covert channel preventing possible detection by the receiver or any other intermediate nodes.

Being a middleman does not necessarily mean the covert sender/receiver has to be physically separated from the overt sender/receiver. Covert sender and receiver could be located on routers/gateways between the overt sender and receiver, but they could also be on the same physical device located in lower levels of the network protocol stack.

Figure 2 illustrates the possible combinations of covert sender and receiver locations. The actual communication scenario depends on the application of the covert channel. For example, if the covert channel is used to circumvent censorship covert and overt sender/receiver would likely be identical, whereas if it is used by a hacker for ex-filtrating data the covert sender and receiver would likely be middlemen (e.g. the sender could be inside the network protocol stack of the compromised machine and the receiver could be on a router close to the edge of the compromised network).

Covert Channels

In this section we give an overview of existing covert channel techniques in computer network protocols. In contrast to previous work we group channels according to their mechanism and not just based on the layers of the Open Systems Interconnection (OSI) model. We believe our approach is advantageous because it provides a more fine-grained classification and also accommodates the fact that some methods could be applied very similarly on different OSI layers. For example, the address modulation channel described later works on the link layer with MAC addresses, on the network layer with IP addresses and on the transport layer with UDP/TCP port numbers.

Unused Header Bits

Covert channels can be encoded in unused or reserved bits of frame or packet headers. This is particularly problematic if

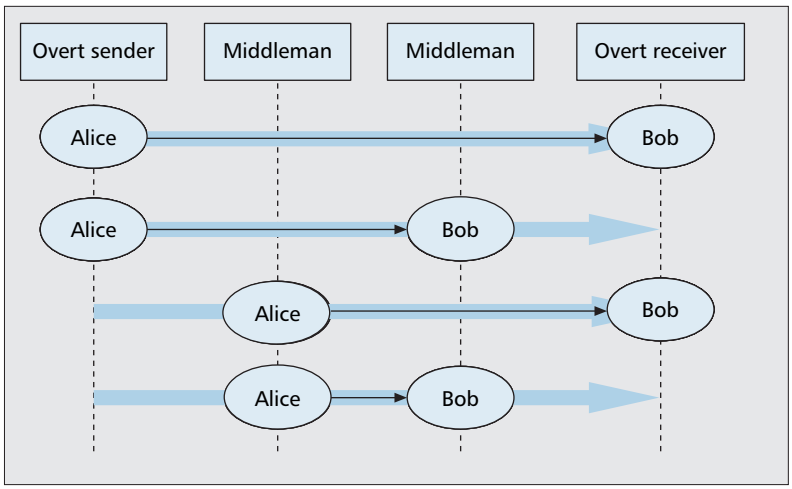

Figure 2. *Possible combinations of different covert sender/receiver locations.*

[1] *Cryptographic protocols are usually illustrated using two participants named alphabetically (Alice, Bob) or with names where the first letter matches their role (Wendy the warden).*

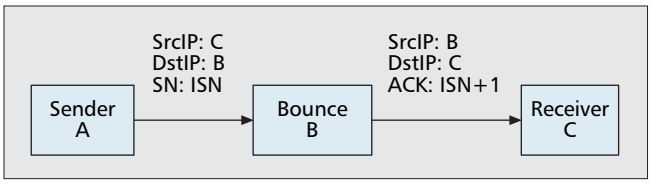

Figure 3. *The TCP Initial Sequence Number (ISN) bounce channel.*

protocol standards do not mandate specific values or receivers do not check for the standard values. Handel *et al.* proposed a covert channel using the unused bits of the IP header's Type of Service (TOS) field or of the TCP header's Flags field [32]. Kundur *et al.* suggested using the IP header's Don't Fragment (DF) bit as a covert channel [34]. The DF bit can be set to arbitrary values if the sender knows the Maximum Transfer Unit (MTU) size of the path to the receiver and only sends packets of less than MTU size. Hintz proposed transmitting covert data in the TCP Urgent Pointer (used to indicate high priority data) that is unused if the URG bit is not set [35].

Fisk *et al.* identified covert channels in TCP Reset segments (TCP segments with the RST flag set abort the connection and usually contain no data) and in the unused code fields of some Internet Control Message Protocol (ICMP) messages [4]. Wolf proposed covert channels in header fields of multiple (now obsolete) protocols such as Token Ring [36]. Lucena *et al.* identified a number of covert channels in various IPv6 header fields such as Traffic Class and Flow Label [37].

Header Extensions and Padding

Many protocols support extension of the standard header. Usually there are some pre-defined header extensions that allow transporting non-mandatory information on demand, but many protocols also allow header extensions to carry data not foreseen in the original specification, extending the capabilities of the protocol.

Graf proposed transmitting covert information in the IPv6 destination options header [38]. This header carries optional information for the packet's destination node. If the option type is set so that the receiver ignores the option, covert information can be directly encoded as option data. Lucena *et al.* identified covert channels in the IPv6 Hop-by-Hop, Routing, Fragment, Authentication and Encapsulating Security Payload extension headers [37]. Trabelsi *et al.* proposed to hide covert data masked as IP addresses in IP Route Record option headers [39].

Covert information can be encoded in frame or packet padding. For example, Ethernet frames must be padded to a minimum length of 60 bytes. If the protocol standard does not enforce specific values for the padding bytes, any data can be used [32, 36]. Padding of the IP and TCP header to 4-byte boundaries (in case header options are present) and padding in IPv6 can also be used to transmit covert data [4, 37].

IP Identification and Fragment Offset

The IP Identification (ID) header field is used for reassembling fragmented IP packets. The only requirement from the IP standard is that each IP ID uniquely identifies an IP packet for a certain time period [5]. The Fragment Offset is used to determine in which sequence the fragments need to be reassembled.

Rowland proposed multiplying each byte of the covert information by 256 and directly using it as the IP ID [15]. Ahsan *et al.* proposed transmitting covert information in the high e ight bits of the IP ID (as the XOR of the data and a secret key) and generate the low eight bits randomly [40].

Cauich *et al.* described how to use this covert channel between middlemen [41]. If an existing packet is not fragmented the covert sender inserts the covert information into the IP ID and Fragment Offset fields and sets one bit from the reserved bits of the flags field. This bit is used to mark packets with covert information so that the covert receiver can distinguish between real final fragments (which have the More Fragments bit set to zero) and packets with hidden information.

Danezis proposed an indirect covert channel using the IP ID field [11]. This channel requires an (unwitting) intermediary host running an operating system with globally incrementing IP ID counter for outgoing packets. Furthermore, the covert sender and receiver must be able to force the intermediary to receive packets and send packets back (e.g. using ping). In each time interval the covert sender sends n packets to the intermediary, where n is the encoded covert information, forcing it to send n packets back. At the start of each time interval the covert receiver forces the intermediary to send one packet. The covert receiver can recover n (the covert information) by computing the IP ID difference of two consecutive packets received from the intermediary.

TCP Initial Sequence Number Field

TCP sequence numbers are used to coordinate which data has been transmitted and received guaranteeing reliable transport. The first sequence number selected by the client is called the Initial Sequence Number (ISN). The ISN must be chosen such that the sequence numbers of new incarnations of a TCP connection do not overlap with the sequence numbers of earlier incarnations of a TCP connection [42].

Rowland proposed multiplying each covert byte with 256^3 and directly using it as the TCP ISN [15]. He also outlined an indirect channel called the bounce channel (Fig. 3). Instead of sending the ISN directly in a TCP SYN packet to the receiver, the sender sends the TCP SYN packet to a bounce host with a spoofed IP source address set to the intended destination. Upon reception of the SYN packet the bounce host sends a SYN/ACK or SYN/RST to the receiver with the acknowledged sequence number equal to the ISN+1. The receiver decrements the ACK number and decodes the hidden information.

Rutkowska developed a covert channel based on TCP ISNs for Linux, where the covert information is encrypted in the ISN fields so that the resulting distribution is uniform random [43]. However, Murdoch *et al.* pointed out that all the previous proposed ISN techniques produce a different distribution than the real operating system implementations [44]. They developed ISN covert channels tailored for Linux and OpenBSD, where the ISN distribution of the covert channel looks like the normal ISN distribution.

Checksum Field

Abad described how the IP header Checksum field could be used for covert communication [45]. The Checksum field is modified to encode the secret information and an IP header extension is added with the content chosen such that the modified checksum is correct again. The same technique could be used for the TCP header checksum. However, since this technique requires adding a header extension the covert information could also be encoded directly into the header extension.

Since the checksum in UDP packets is optional [46], Fisk *et al.* proposed using its presence or absence to signal one bit of covert information per UDP packet [4].

Modulating the IP Time To Live Field

Jones *et al.* proposed a covert channel based on the IP header Time to Live (TTL) field as solution to trace back IP flows

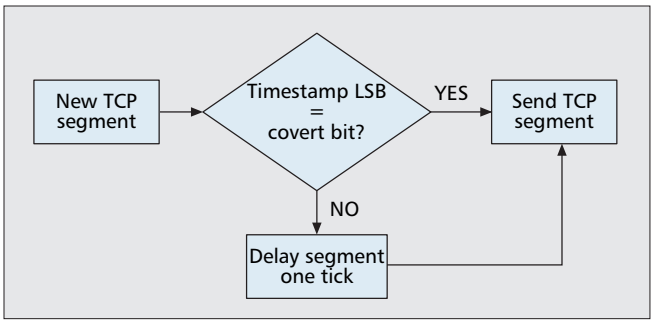

Figure 4. *Modulating the least significant bit of the TCP timestamp field.*

without using the source address field [25]. In their approach, routers modulate the TTL field of packets so that downstream receivers can unambiguously identify their upstream router. The covert channel is used for marking instead of general purpose communication.

Qu *et al.* and Lucena *et al.* proposed techniques for embedding covert information into the TTL [47] and the IPv6 Hop Limit field (IPv6 equivalent of the IP TTL) [37]. Neither scheme takes into account typical initial TTL values chosen by the sender and *normal* TTL variation occurring in networks. Zander *et al.* analysed initial TTL values and normal TTL variation and proposed an improved covert channel encoding in the TTL field that is harder to detect [48].

Modulating Address Fields and Packet Lengths

Any communication protocol uses address fields to identify senders and receivers. The most prominent today are arguably the IP source and destination address part of the IP header. Padlipsky *et al.* and Girling proposed either directly encoding information in the destination address fields or by modulating the order of valid destination addresses in subsequent transmissions [2, 3]. An amount of bits can be communicated based on the number of different addresses a covert sender can use. The initial proposal was targeted to link layer frames but the technique can be used on other layers, for example IP addresses (network layer) or port numbers (transport layer) can be modulated. Covert information can also be transmitted in source addresses, if they can be modulated [37]. This is the case for IP addresses (if spoofing is possible) or port numbers.

Any communication protocol uses length fields to indicate the length of headers, header extensions or messages (frames, packets). Padlipsky *et al.* and Girling proposed to modulate the lengths of link layer frames to transmit covert information [2, 3]. The same technique could be used to modulate the size of IP/UDP/TCP packets as proposed in [37].

Perkins developed a protocol-independent covert channel that encodes the information in the sum of all bits of a message [49]. Covert sender and receiver agree on the maximum possible sum S (all bits set in a message of maximum length) and a division of $[0, S]$ into several intervals. The channel capacity depends on the number of intervals e.g. two bits per packet can be transmitted if there are four intervals. The covert sender encodes covert bits by constructing or reordering messages so that the bit sum is in the desired interval. The covert receiver decodes the information by identifying the interval in which the bit sum of a received message lies.

Modulating Timestamp Fields

Handel *et al.* proposed modulating the timestamp of the IP timestamp header extension to transmit covert data [32]. However, this header extension limits a packet to only 24 hops and is no longer used.

Giffin *et al.* developed a method for covert messaging through TCP timestamp header options, which are widely used to improve TCP performance [50]. Covert information is inserted in the low order bits of the sender timestamps, because these are effectively random for slow TCP connections. Instead of directly modifying timestamps the algorithm slows the TCP stream so that the timestamps on packets are valid when they are sent. The algorithm compares the least significant bit (LSB) of every TCP segment generated by the system with the current covert bit to be sent. If the LSB matches the covert bit the TCP segment is sent immediately otherwise it is delayed for one timestamp tick (Fig. 4).

Packet Rate/Timing

Covert information can be encoded by varying packet rates, which is equivalent to modulating the packet timing (the interpacket times) [2, 3, 51]. The covert sender varies its packet rate between two (binary channel) or multiple packet rates each time interval. The receiver measures the rate in each time interval and decodes the covert information. A binary channel can transmit one bit per time interval, whereas a multi-rate channel can transmit $\log_2 r$ bits per time interval where r is the number of distinct rates. The sender and receiver need a mechanism for synchronisation of the time intervals.

Padlipsky *et al.* outlined a timing channel where the sender either transmits or stays silent in each time interval [2]. This on/off timing channel is a special case of the binary channel where one rate is zero and the other rate is chosen arbitrarily. Girling also proposed covert timing channels at the packet level and suggested mitigating the noise problem by the length of the inter-packet delay (which is inversely proportional to the channel capacity) [3]. Cabuk *et al.* implemented the on/off timing channel [51]. In their scheme the covert data is divided into small fixed-size frames and synchronisation between sender and receiver is achieved through a special start sequence at the beginning of each frame. As the authors note this scheme does not entirely solve the synchronisation problem and they propose several better techniques as future work.

Berk *et al.* introduced a variant of the packet-timing channel that does not require sender-receiver synchronisation because the covert information is encoded directly in the inter-packet delays of consecutive packets [52]. They compared channels with two delays (binary channels) and multiple delays, and developed a mechanism by which the sender can pick the optimal symbol distribution in multi-rate channels given the channel characteristics.

Murdoch developed an indirect covert channel that is a hybrid between the packet rate and timestamp modulation channels [53]. The channel requires an intermediary that receives and sends packets to both covert sender and receiver. The channel exploits the fact that a host's CPU temperature is proportional to the number of packets per time unit it processes and a host's system clock skew depends on the temperature. The covert sender either sends packets to the intermediary or stays silent. The covert receiver estimates the intermediary's clock skew by observing a series of timestamps in packets sent by the intermediary (e.g. the TCP timestamps introduced earlier). Although Murdoch developed the technique specifically for identifying the operator of servers hidden by anonymisation networks he concludes that it could also be used as a noisy channel for general covert communication.

Message Sequence Timing

Wolf mentioned the possibility of constructing covert channels by modulating the use of protocol operations [36]. For exam-

ple, a receiving station can acknowledge each frame separately or wait until two frames have arrived before acknowledging the first. Handel *et al.* proposed a covert channel based on modulating the clear to send/ready to send (CTS/RTS) signals of serial port communication [32]. This technique can be applied to other network protocols utilising CTS/RTS (e.g. Wireless LANs). Handel also proposed using the timing of ICMP source quench messages [54] to create a covert channel. A gateway or receiver of an overt communication can send source quench messages to the sender requesting a decrease of traffic rate (flow control). In practice ICMP source quench messages are no longer used.

Hintz described an indirect timing channel [35]. The sender sends a large number of requests to a server or stays silent in each time interval, equivalent to one bit per time interval. The receiver periodically probes the server and measures the response time to recover the covert information. Eßer *et al.* implemented a web-based timing channel and analysed its capacity [55]. In their scheme a web server sends covert data to a client by delaying a response (one) or responding immediately (zero).

Packet Loss and Packet Sorting

Servetto *et al.* demonstrated that channel erasures (packet loss) intentionally introduced at the sender could be used as a low-rate covert channel [56]. In practice, the technique requires per packet sequence numbers and erasures are realised by skipping sequence numbers (artificially losing packets at the sender). Therefore, the authors also refer to this mechanism as *phantom packets*.

Kundur *et al.* described a covert channel implemented through packet sorting [34]. Because a set of n packets can be arranged in any $n!$ ways a maximum of $\log_2 n!$ bits can be transmitted. This approach requires per packet sequence numbers to determine the original packet order. Instead of actually sorting the packets the method only modifies the sequence numbers, thus keeping payload sent across multiple packets intact. The authors proposed using the sequence number of the IPSec Authentication Header (AH) or Encapsulating Security Payload (ESP) [57], but other sequence numbers could possibly be used as well (e.g. the TCP sequence number).

Galatenko *et al.* proposed sending covert information by reordering packets so that destination addresses in a series of subsequent packets are ordered [58]. The covert sender encodes a one as a sequence of packets with increasing addresses and a zero as a sequence of packets with decreasing addresses. The sequence length used depends on the desired error rate of the channel.

Frame Collisions

Handel *et al.* proposed exploiting the Ethernet Carrier Sense Multiple Access Collision Detection (CSMA/CD) mechanism [32]. If frames collide in CSMA/CD, a jamming signal is issued and the senders back off a random amount of time. The covert sender jams any packets of another user. Then it uses a back-off delay of either zero or the maximum value. Therefore all frames sent will either lead or lag packets sent by the other user, essentially creating a one bit per frame covert channel. The receiver can recover the information by detecting the collisions and analysing the order of the frame arrivals. Bhadra *et al.* proposed a similar jamming channel in the slotted ALOHA protocol [59].

With the transition to switched Ethernet CSMA/CD has disappeared, but the same technique can be used in current Wireless LAN networks as proposed by Szczypiorski *et al.* [60]. Since wireless networks use air connections with variable bit error rate (BER), they provide the opportunity for injecting synthetic *corrupt* frames. All stations that are part of the covert channel communicate via sending some percentage of their frames with intentionally created bad checksums. Other stations discard the *corrupted* frames.

To improve performance of shared medium access, splitting algorithms are used to divide the set of collided senders into smaller subsets and then these subsets retransmit in order. Dogu *et al.* designed a covert channel using the First Come First Serve (FCFS) splitting algorithm [61]. The covert information is conveyed in the number of collisions observed in a collision resolution period. The covert sender controls this number by generating dummy packets and thus causing additional collisions. The covert receiver passively monitors the channel and keeps track of the collision resolution procedure to extract the covert information.

Li and Ephremides' transmission scheme uses the covert sender's splitting decisions (which subsets it joins) as the carrier of covert information [62]. The covert receiver passively tracks the collision resolution procedure. When the covert receiver detects a successful transmission from the covert sender it can retrieve the past splitting decisions, which is the encoded covert information.

Ad-hoc Routing Protocols

Marone presented several covert channels in the Dynamic Source Routing (DSR) protocol used for routing in ad-hoc networks [63]. Covert information can be encoded in header fields present in DSR routing requests, for example the request identification number, hop limit, clock time, or address fields. Covert data can also be piggybacked on regular routing requests in the options header. Another, more sophisticated method presupposes that the covert sender and receiver have a prearranged list of routes, where each route is a symbol. Sending a combination of routes transmits the covert information.

Li *et al.* described a number of covert channels in the Ad hoc On-Demand Distance Vector (AODV) protocol [64]. A covert sender can modulate the delays between successive AODV route requests, and the covert receiver can decode the information from the message timing. Furthermore, covert information can be transmitted by manipulating the source sequence number field in the route requests, by embedding covert information in the destination ID of route requests, or by manipulating the lifetime field in route replies sent by an intermediate node.

Wireless LAN (WLAN)

Szczypiorski *et al.* proposed embedding covert data in the RC4 initialisation vector, which is part of the IEEE 802.11 Wired Equivalent Privacy (WEP) mechanism [60].

Qu *et al.* developed a covert channel based traceback mechanism for WLANs to improve resistance against denial of service attacks [26]. In their approach intermediate access points encode the path of frames in the More Fragments bit of the Frame Control field and the Duration/ID field of 802.11 header fields.

Butti *et al.* proposed sending covert information across 802.11 networks in unsolicited ACK frames or invalid frames (frames with deliberately incorrect checksums) [65]. The covert sender encodes the covert data and a magic number inside the receiver address. The covert receiver decodes the information from frames containing the magic number.

Krätzer *et al.* proposed two covert channels in 802.11 frames [66]. The first channel transmits the covert information embedded in various header fields, such as the Retry bit and More Data bit of the Frame Control field, and the Duration/ID field. The second channel transmits information through the *retransmission* of frames. The covert sender encodes covert bits by duplicating frames of specific connections (frames going from a particular sender to a particular receiver) and the covert receiver decodes the hidden data by detecting the duplications.

Hypertext Transfer Protocol (HTTP)

Bauer proposed using covert channels in web traffic of uninvolved users to enlarge the set of users and improve the degree of anonymity [67]. The information is hidden in JavaScript/HTML and transported through the use of JavaScript redirects. An observer who cannot look into the content transported by HTTP [68] cannot distinguish between harmless web surfers and the covert senders/receivers.

Feamster *et al.* proposed Infranet — a framework to use covert channels in HTTP to circumvent censorship [14]. Web servers participating in Infranet receive covert requests for web pages encoded as a sequence of HTTP requests to harmless web pages and return the content hidden inside harmless images using steganography. Bowyer proposed a very similar mechanism to communicate with Trojans behind firewalls [69]. A Trojan on the compromised system sends HTTP requests to a web server, with the covert data encoded as URL parameters. The web server returns innocent looking web pages with images that contain hidden data (steganography).

Dyatlov *et al.*, Kwecka and Van Horenbeeck proposed various methods for embedding covert channels into HTTP protocol headers [70–72]. These encompass encoding covert data into header field values, the order of header fields, the use of lower or upper case, the presence or non-presence of optional header fields, the use of multiple white spaces, and new non-standard header fields. More recently Castro *et al.* have also developed a method for transmitting covert information through HTTP cookies [73].

Domain Name System (DNS) Protocol

An anonymous author proposed an indirect covert channel over the DNS protocol [74]. The channel exploits negative caching of domain names [75]. Covert sender and covert receiver agree on a series of non-existent domain names. The covert sender recursively queries for all domain names for which it wants to transmit a one and does nothing otherwise. The covert receiver non-recursively queries for all domain names interpreting a cached response as one and an uncached response as zero.

Kaminsky and Gil independently implemented tools to covertly tunnel IP packets over the DNS protocol [76, 77]. Communication takes place between a client and a fake DNS server. The client sends data to the server in DNS requests (hostname lookups) where the actual hostnames are the encoded covert information. The server sends data back to the client contained in the DNS responses.

Other Application Protocols

Mazurczyk *et al.* proposed using covert channels and watermarking to embed control information in voice over IP (VoIP) streams [23, 24]. VoIP data transmission is usually based on the Real-time Transport Protocol (RTP), and control information is separately exchanged over the Real-time Control Protocol (RTCP) [78]. Instead of using a separate RTCP flow the authors proposed embedding the control information into the actual RTP flow. Unused bits in the IP/UDP/RTP headers signal the type of parameters, whereas the parameter values are embedded as watermark in the voice data.

Lucena *et al.* developed covert channels for the Secure Shell (SSH) protocol [33]. The first technique hides information in the Message Authentication Code (MAC) header present in each packet. The sender either completely replaces the MAC with encrypted covert data, or uses a short MAC padded with encrypted covert information to make it resemble a long MAC. This mechanism works only if the covert sender/receiver is also the overt sender/receiver. The second method also works if covert sender and receiver are middlemen. The covert sender intercepts the SSH traffic and adds an additional fixed-size encrypted message at the beginning of the already encrypted payload. A 32-bit magic number at the start marks the presence of the covert data. The covert receiver decodes the covert information and removes it, restoring the original packet.

Zou *et al.* proposed two mechanisms for embedding covert channels into the File Transfer Protocol (FTP) [79]. The first mechanism encodes covert bits directly into the FTP commands defined [80]. The second mechanism transmits covert data through varying the number of FTP NOOP commands send during idle periods — the number of NOOPs sent is equal to the integer value of the covert data.

Payload Tunneling

Payload tunnels are covert channels that tunnel one protocol (usually the IP protocol) in the payload of another protocol. The main purpose of these channels is circumventing firewalls that limit outgoing traffic to few allowed application protocols (e.g. HTTP). Therefore, most of these channels do not aim for stealth but rather for maximising the throughput. A variety of tools exist for tunneling over application protocols that are often not blocked such as ICMP or HTTP [81].

One of the first approaches for tunneling protocols over ICMP was Loki, which tunnelled data in the payload of ICMP echo messages [82]. Today many IP over ICMP tunnel solutions exist, for example Stødle implemented another IP over ICMP tunneling tool [83]. Zelenchuk implemented an indirect IP over ICMP tunnel [84]. The covert sender sends echo request packets to a bounce host with spoofed source address (set to the address of the covert receiver) and the covert data encoded in the payload. The bounce host then sends echo replies to the covert receiver with the same payload as in the requests.

Another popular method is to tunnel protocols over HTTP. Padgett developed a tool that tunnels SSH over HTTP proxies [85]. Dyatlov and LeBoutillier implemented tools for tunneling UDP or TCP over HTTP [86, 87]. Lundström implemented a tool that can establish a bi-directional tunnel over the exchange of emails [88].

Covert Channel Countermeasures

In this section we provide an overview of the available countermeasures. In the folloing sections we describe in detail countermeasures against the covert channels outlined.

Identification

Before any action can be taken against a covert channel, first it needs to be identified. The identification of a covert chan-

nel can be ad-hoc or based on a formal method. A number of formal methods were developed for identifying covert channels in specifications or implementations of single host systems: information flow analysis [89, 90], non-interference analysis [91], Shared Resource Matrix (SRM) method [92, 93] and the Covert Flow Tree (CFT) method [94]. These techniques can be applied to identify covert channels during the design phase, or in an already deployed system. Gligor provides a good introduction to the different methods (except CFT) [7]. There are only a few works on formal techniques for identifying covert channels in computer network protocols.

Donaldson *et al.* discussed how analysis techniques, SRM in particular, could be applied to network protocol covert channels [95]. They proposed analysing network covert channels by separately inspecting host-to-host channels on the lower network layers and intra-host channels between processes on a single host (which can be different layers in the network stack).

Hélouët *et al.* proposed performing covert channel analysis for distributed systems at the requirement level, when design decisions can still be made to eliminate or limit covert channels [96]. Their approach is based on a representation of requirements by scenarios. Covert channels detected during the design phase are likely to be present in any implementation, and thus are not implementation-specific.

Aldini and Bernardo proposed a methodology for combining covert channel identification and performance evaluation [97]. The advantage of this integrated approach is that it provides insights into how to trade off quality of service with covert channel capacity. The authors applied their methodology to the PUMP model (explained later), obtaining the relation of covert channel capacity and number of connection requests served per time unit.

Countermeasures

Once a covert channel has been identified the generally available countermeasures are [98]:

- Eliminate the channel
- Limit the bandwidth of the channel
- Audit the channel
- Document the channel

According to Jeng *et al.* there are two causes of covert channels: design oversights and weaknesses inherent in the system design [98]. While covert channels caused by oversights may be corrected once discovered, those intrinsic to the system can never be removed without redesigning the system. Therefore, ideally covert channels should be identified and removed during the design phase.

If a covert channel was not removed in the design phase the next best option is to eliminate its possible use, because even very low capacity channels could be successfully exploited. However, the removal of all covert channels leads to very inefficient systems, as covert channels can often only be completely removed by replacing automated procedures with manual procedures [99]. Furthermore, covert channels based on the modulation of visible message parameters are inherent in distributed systems, such as computer networks. Therefore, we and many other researchers believe covert channels cannot all be completely eliminated [98, 100].

If a channel cannot be eliminated its capacity should be reduced. What is an acceptable capacity depends on the amount of information leakage that is critical. For example, if the channel is so small that classified information cannot be leaked before it is outdated, then the channel capacity is tolerable. Limiting the channel capacity is often problematic, because it means slowing down system mechanisms or introducing noise, which both limit the performance of the system. Note that this approach assumes that there are means to determine the capacity of the channel.

Any covert channels that cannot be removed should be audited. Auditing acts as deterrence to possible users of the covert channel and requires its reliable detection. Covert channels with capacities too low to be significant, or which cannot be audited should at least be documented (e.g. in the protocol specification), so that everybody is aware of their existence and potential threat.

Eliminating Covert Channels

In this section we describe approaches to eliminate many of the covert channels described earlier.

Host Security

Securing hosts cannot remove covert network channels, but it can prevent their exploitation in some application scenarios. If hosts were secured from being hacked, the installation of Trojans, and the modification of software or the network stack would be impossible, thus hackers could not exploit covert channels. However, relying on host security could be dangerous and it would be better to eliminate covert channels in the first place where possible. Furthermore, this approach does not solve the covert channel problem in other application scenarios (e.g. censorship circumvention).

Network Security

One approach to counter covert channels is to block protocols/ports that are susceptible. For example, ICMP is blocked by many firewalls these days rendering approaches such as Loki [82] ineffective. Obviously, in the Internet some protocols cannot be blocked because they are vital (e.g. IP, TCP, DNS), or because their services are too important (e.g. email, Web). However, in a closed network protocols prone to covert channels could be blocked, or replaced by versions with fewer or limited covert channels.

The leakage of classified information from a high security system to a low security system (the classic covert channel) can be prevented by a network design where only hosts on the same security level are allowed to communicate. Such an approach may be practical for highly secure networks, but not for diverse large open networks such as the Internet.

Bouncing covert channels as described by Rowland [15] only work if IP address spoofing is possible. Besides solving a number of other security issues preventing IP spoofing (e.g. by ingress/egress filtering) closes such channels. Furthermore, securing networks against wiretapping, and securing routers against compromise prevents some covert channel scenarios in which covert senders or receivers act as middlemen [2].

Traffic Normalization

Many of the channels described earlier can be eliminated by normalising protocol headers, padding and extensions as described by Malan *et al.* [101], Handley *et al.* [102] and Fisk *et al.* [4] in general, or more specifically for the IPv6 protocol by Lucena *et al.* [37] and ICMP tunneling by Singh [103]. Traffic normalisation can be performed by end hosts or by network devices such as firewalls or proxies.

Unused or reserved bits and padding can be dealt with easily by setting them to zero and unknown header extensions can be removed. Some covert channels exploit the fact that

certain header fields are not always used (and their use is indicated by other header fields). This fact can be used for normalisation as well. For example, set the IP ID to zero if the DF bit is set, set the Urgent Pointer to zero if the URG bit is not set, and set the Fragment Offset to zero if the DF bit is set. Furthermore, it should be ensured that checksums are always used and correct.

A number of other header fields can be rewritten under certain assumptions. For example, enable the DF bit and set IP ID and Fragment Offset to zero if the packet is below the MTU size (assuming the normaliser knows the MTU), rewrite the IP ID (assuming the normaliser can manipulate all fragments), rewrite the TCP ISN, source IP address and source port (assuming the normaliser can keep a mapping between original and new values and rewrite packets going in the opposite direction accordingly). Time-to-Live and TCP timestamp can also be rewritten (assuming the normaliser is located at or very close to the source) or the low order bits can be randomised (similar to fuzzy-time described in [104]).

The same concepts can be used for eliminating covert channels in application protocols. Schear *et al.* proposed eliminating covert channels in HTTP responses by enforcing protocol-compliant behaviour and restricting usable response headers to a fixed set in a particular order, and by verifying response header fields against the corresponding object metadata and client request [105].

Limiting Covert Channel Capacity

In this section we describe techniques that can be used to limit the capacity of some channels described earlier. A prerequisite of determining the efficiency of capacity limitation is that the capacity of the covert channel can be estimated.

Capacity Estimation

In the absence of noise the covert channel capacity can be estimated based on the size of the object values (storage channels) or the amount of information encodable in the resources (timing channels) and the speed with which the objects/resources can be modulated. For some channels it is easy estimating the capacity in terms of bits per packet or bits per message sequence. For example, the channels proposed in [15] have a capacity of one byte per packet. However, capacity in bits per second depends on the amount of the overt traffic between covert sender and receiver or on the amount of suitable overt traffic available in the network (if the sender is a middleman).

Some covert channels have inherent noise, for example the packet rate channels described in [2, 51, 52] suffer from noise introduced by sender timing inaccuracies, network switches and routers, cross traffic, or jammers (active wardens). More general approaches analysing the capacity of noisy timing channels are based on Shannon [106].

Millen estimated the capacity of covert timing channels with noise and/or memory [107], while Moskowitz analysed the capacity of discrete, noiseless, and memoryless timing channels [108]. Venkatraman extended Moskowitz work towards a mode-based system [109]. A mode-based system switches between modes in certain time intervals and only allows a certain number of transitions in each mode, effectively limiting the covert channel capacity. Berk *et al.* studied the capacity of binary and multi-symbol inter-packet delay channels [52]. Gray developed an upper bound for the capacity of timing channels when Wei-Mings' fuzzy time [104] is used [110]. Bhadra *et al.* derived the capacity of the frame collision channel for slotted ALOHA [59].

Limit Address and Packet Length Modulation

To counter the address modulation channel described earlier previous research has suggested limiting the number of possible addresses [2, 3, 95]. If the number of different usable addresses is small, the covert channel capacity is very small. Limiting the number of addresses effectively means limiting the allowed host-to-host connections. This may be possible in closed networks, but not in open networks such as the Internet. The sender address should always be fixed (preventing IP spoofing), but the number of destination addresses can hardly be limited to a small number. Similarly, usable source and destination ports can hardly be restricted.

Instead of limiting the interactions between hosts, sending dummy packets between random hosts inserts noise into the traffic patterns. Indirect routing achieves the same effect with lower overhead [111].

Padding all packets to a common size eliminates the packet length modulation channel introduced earlier [2], but this adds significant overhead and decreases efficiency, especially for small packets. Anonymiser networks use a constant packet size to prevent traffic analysis [111], but it seems unrealistic that such approaches would ever be deployed on a large scale. To increase efficiency Girling proposed allowing a certain number of possible packet sizes; the number of available packet sizes should be small enough to limit the covert channel capacity appropriately [3].

However, it seems unlikely that a sender's ability to modulate packet size could be limited in current IP networks (at least the UDP packet size can be modulated within the limits of the MTU).

Limit Packet Rate/Timing Channels

Multiple solutions have been proposed to limit the capacity of the channels described earlier: either random noise is introduced to mask the covert channel or the overt channel is forced using fixed packet/message rates and dummy packets or messages are inserted when useful information is not sent [98].

Wei-Ming's *et al.* fuzzy-time proposal makes all clocks in the system noisy [104]. A sender cannot exactly time outgoing packets and a receiver cannot accurately measure the timing, which reduces the capacity of the covert channel.

Link padding forces a packet flow to adhere to a specific traffic pattern (e.g. packet rate) by delaying packets and injecting dummy packets if necessary (for preventing on/off channels) and should eliminate packet rate/timing channels [112]. However, Graham *et al.* showed that even if link padding is used information about the payload traffic rate is still leaked because of the inability of the padding gateway to completely isolate the processing of outgoing packets from the interrupt processing necessary to handle incoming packets [113]. These imperfections of link padding can still be used as a covert channel.

Because padding links to a single packet rate creates significant overhead, Girling proposed that senders could emit a small number of different packet rates [3]. This increases efficiency and limits covert channels to acceptable capacities.

Message sequence timing channels can be eliminated by buffering and delaying connection attempts, service requests etc. Spurious data can be inserted into the network against wiretapping receivers, but this does not help against end-host receivers. Schear *et al.* proposed delaying HTTP responses to limit the capacity of timing channels in HTTP [105].

Giles *et al.* [114] studied the problem of limiting the capacity of covert timing channels in the framework of a game between the covert sender-receiver pair and a jammer. The

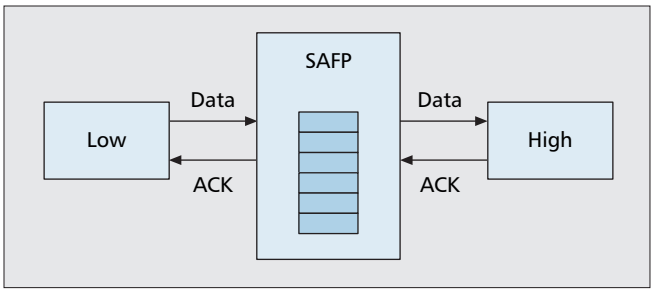

Figure 5. *The Store and Forward Protocol (SAFP) gateway — a simple approach for limiting the covert timing channel in the flow of ACKs from high to low.*

jammer attempts to re-time the packets from the covert sender. The mutual information between the input and output processes is the objective of the game: the jammer wishes to reduce this, while the covert sender-receiver pair would like it to be high. The authors proved the value for certain games and provided coding schemes for sender-receiver pair and jammer.

Again, it seems unlikely that these countermeasures could be deployed in current IP networks and therefore packet rate/timing channels are likely to stay.

Split Connections

One of the simplest most common security policies is the Bell-LaPadula model [115]. It can be summarised as *no read up and no write down* where *up* is an entity with a higher security level and *down* is an entity with a lower security level. A problem arises when a low entity wants to reliably send data to a high entity. Reliable communication requires the high entity sends back Acknowledgments (ACKs) for the data received. The timing of the ACKs can be manipulated to transmit hidden information from high to low. A number of methods have been proposed for minimising the capacity of this covert timing channel [116].

In the Store And Forward Protocol (SAFP) a gateway sits between the low-security sender and high-security receiver (Fig. 5). When the gateway receives a packet from low it stores it into a buffer and sends an ACK to low. Then it transmits the packet to high and waits for an ACK. If the ACK is received the gateway removes the packet from the buffer. However, when the buffer is full the gateway must wait for high to acknowledge a received packet until another packet from low can be acknowledged and stored in the buffer; the time it takes the gateway to send an ACK to low is directly related to the time of receiving an ACK from high. Since high can vary the rate of its ACKs it can ensure the buffer is always filled and still exploit the covert channel.

The PUMP model introduced by Kang and Moskowitz substantially reduces the covert channel capacity of the SAFP [117, 118]. The PUMP uses an historic average of high's ACK-rate as the rate of sending ACKs to low (Fig. 6). For every packet from high received by the trusted high process a moving average of high's ACK rate is updated. When the trusted low process receives a message from low it inserts the message into the buffer, and then sends an ACK to low after a delay. The delay is a random variable chosen from an exponential distribution with the mean equal to the current average of high's ACKs rate. Although the PUMP does not completely eliminate the covert channel it does significantly decreases its capacity by *decoupling* ACKs sent to low from high's ACK timing and by adding random noise.

Ogurtsov *et al.* proposed the quantised pump, which is an easier to analyse version of the PUMP [116]. It has a provable upper bound on the covert channel capacity and provides the same performance as the original PUMP. Lanotte *et al.* defined a probabilistic model for the PUMP that allows computing the probability of security violations depending on various parameters (e.g. buffer size) [119].

Auditing Covert Channels

Auditing of covert channels requires reliable detection by a passive warden. Many of the covert channel techniques described earlier only provide *security through obscurity* and can be detected easily. All proposed detection methods are based on the detection of non-standard or abnormal behaviour (anomaly detection). The assumption is that the warden knows the normal behaviour of protocols and hosts and can detect abnormal behaviour caused by covert channels.

While it is possible to detect many of the proposed covert channels, every channel that looks identical to normal use of the protocol will be harder to detect. For example, Murdoch's TCP ISN channel has a value distribution matching the distribution of real operating systems [44], and Lucena's SSH MAC channel has statistical characteristics identical to real MACs [33]. The only way a passive warden could reliably detect these covert channels is to somehow detect the embedding process at the covert sender.

Header Field Channels

Most protocol standards mandate that unused or reserved bits and padding must be filled with specific values (e.g. zeros). Even if this is not the case the behaviour of actual implementations can be viewed as de-facto standards [44]. All covert channels based on non-standard use of the protocol can be detected easily. Furthermore, some proposed covert channels are obsolete because previously unused bits are now used (for example, some unused bits in the IP header are now widely used for explicit congestion notification), or defined messages or extension headers are de-facto not used anymore and their appearance would be suspicious (for example ICMP based flow control and IP timestamp header extension).

Some covert channels described previously exploit the fact that header fields can have *arbitrary* values within the requirements of the standard. However, if the fields are naively used and the resulting value distributions are different from the distributions generated by real operating systems, the covert channels are easy to detect (as demonstrated for IP ID channels in [44]). The usual approach is either to train a classifier on the normal and abnormal behaviour, or to train a classifier on the normal behaviour and detect anomalies. The behaviour

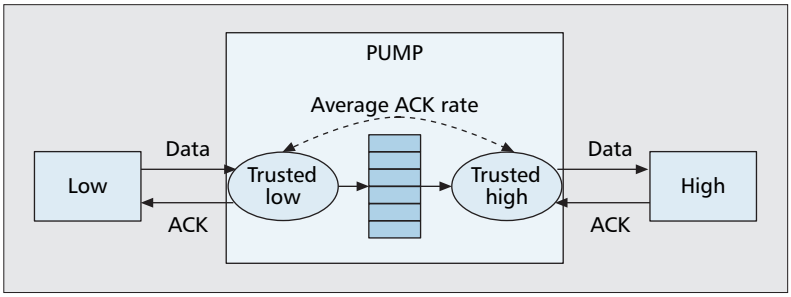

Figure 6. *The PUMP significantly reduces covert channel capacity of the SAFP because it* decouples *high's ACKs from ACKs sent to low.*

is analysed from a set of traffic flows, where each flow is described by a number of characteristics (features).

Sohn *et al.* demonstrated that covert channels with a simple encoding in the IP ID or TCP ISN field (as proposed by [15]) could be discovered with high accuracy using Support Vector Machines (SVMs) [120]. The authors evaluated different feature sets and achieved classification accuracies of up to 99 percent. Tumoian *et al.* evaluated the accuracy of a neural network to detect Rutkowska's [43] TCP ISN covert channel [121, 122]. The neural network was trained to predict successive ISNs for different operating systems. Then ISNs used by hosts are monitored and compared to the prediction model. An actual ISN sequence not matching any prediction model indicates a covert channel. The authors find that for more than 100 consecutive ISNs observed detection accuracy reaches 99 percent.

Application protocol covert channels can be detected in a similar way (as discussed for HTTP in [71, 105]).

Timestamp Channel

Hintz proposed a detection method for the TCP timestamp channel described earlier [35]. In low-speed networks a randomness test can be applied to the LSB of the timestamps. Too much randomness could reveal the covert channel. In high-speed networks the segment rate is usually larger than the timestamp update rate, because the TCP timestamp clock tick is only between 1Hz and 1kHz [123]. A warden can detect the channel by computing the ratio of different timestamps used and total number of possible timestamps (depending on the duration of the connection). For a normal connection the ratio should be very close to 1 (at least one segment sent at every clock tick), but for the covert channel it should be ≤ 0.5 (if the LSB of the timestamp is not equal to the covert bit to be sent at least one clock tick has to be skipped).

Packet Rate/Timing Channels

Venkatraman *et al.* proposed auditing the change of traffic rates over time to detect packet rate channels [109]. If the traffic rate of one host changes by more than a certain threshold this would indicate a covert channel. The authors proposed setting the threshold to the standard deviation of the rate change observed in the past for a large set of hosts.

Cabuk *et al.* proposed a detection method for on/off packet timing channels [51]. They defined a metric that measures whether the cumulative inter-arrival time distribution of a traffic flow has only a small number of high jumps (indicating a covert channel) or is more evenly spread (indicating a normal flow). In an empirical evaluation based on multiple traffic traces Cabuk *et al.* showed that their metric can effectively detect covert channels even if the sender changes the timing interval or there is random noise [51].

Berk *et al.* proposed methods for detecting inter-packet delay channels using two delays (binary channels) or multiple delays (multi-symbol channels) [52]. For binary channels the authors developed a simple detection method based on statistical analysis of the packet inter-arrival times. For covert channels the inter-arrival time histogram has two distinct spikes, and the mean inter-arrival time is between the spikes and has a very low packet count. For normal flows the interarrival time histogram will have a large (if not the largest) spike at the mean inter-arrival time.

For multi-symbol channels Berk *et al.* argued that a skilled covert sender would pick a delay distribution (symbol distribution) that maximizes the capacity. The passive warden can also estimate the optimal distribution, compare it to the distribution of the traffic flow under observation using a similarity test, and detect the presence of the covert channel if both distributions are similar. A fundamental problem with this approach is that a covert sender would probably not choose to maximise capacity if this would compromise the covert channel. Another problem is that the warden would have to build the channel matrix for each suspect traffic flow or have a very large number of pre-built channel matrices based on different times of day, hop counts etc. which seems impractical.

Payload Tunneling

Sohn *et al.* used the same SVM-based approach as described earlier to evaluate the accuracy of detecting covert channels embedded in ICMP echo packets (as done by Loki [82]) and achieved classification accuracies of up to 99 percent when training a classifier on normal packets from Windows, Solaris and Linux and abnormal packets generated by Loki [124].

Pack *et al.* proposed detecting HTTP tunnels by using behaviour profiles of traffic flows [125]. Behaviour profiles are based on a number of metrics such as the average packet size, ratio of small and large packets, change of packet size patterns, total number of packets sent/received, and connection duration. If the behaviour of a flow under observation deviates from the normal HTTP behaviour profile it is likely to be an HTTP tunnel. Borders *et al.* developed a tool for detecting covert channels over outbound HTTP tunnels based on a similar approach [126]. The tool analyses HTTP traffic over a training period, and is then able to detect abnormal HTTP flows using metrics such as request size, request regularity, time between requests, time of the day, and outbound bandwidth usage.

Other Channels

Depending on the normal network conditions high packet loss or packet reordering or a high number of frame collisions could indicate potential covert channels. The SSH middleman covert channel could possibly be detected because it changes the packet length distribution of normal SSH connections. Analysis of address and packet length variation compared to normal variation could reveal these covert channels.

However, it is difficult to detect covert channels if there is a lot of variation in the normal behaviour of the protocol. For example, Krätzer's frame duplication channel is potentially hard to detect because normal frame retransmission rates vary significantly [66].

Conclusions and Future Work

We have given an overview of covert channels in computer network protocols. We have introduced the idea and communication model of covert channels and explained the different application scenarios in which they can be used, many posing serious security threats. We have then described the existing covert channel techniques and the countermeasures used to detect, eliminate, or limit the capacity of covert channels.

Many existing covert channels in network protocols follow the *security through obscurity* approach and in principle their detection or elimination is straightforward. However, while researchers have developed a number of countermeasures, real world experience shows that industry products still lack methods to deal with covert channels. Also some proposed countermeasures could significantly reduce overt channel performance and therefore their applicability in real high-speed networks is questionable. Furthermore, there are many possi-

bilities of creating covert channels in computer network protocols and the complete elimination of all these channels seems almost impossible.

There are a number of directions for further research. Currently, a comprehensive taxonomy for classifying the different covert channels and countermeasures is missing. Additionally, there is a lack of work on capacity estimation in the presence of channel errors, formal methods for identifying covert channels during protocol design, and more holistic approaches for covert channel elimination/detection and their integration into existing network security management methods. Finally, it seems likely that the arms race of developing new covert channels with improved stealth and capacity and developing more effective detection and elimination techniques will continue.

Acknowledgments

We thank Nigel Williams and the anonymous reviewers for their valuable comments, which greatly helped improve the article.

References

[1] B. Lampson, "A Note on the Confinement Problem," *Commun. ACM*, vol. 16, no. 10, Oct. 1973, pp. 613–15.
[2] M. A. Padlipsky, D. W. Snow, and P. A. Karger, "Limitations of End-to-End Encryption in Secure Computer Networks," Tech. Rep. ESD-TR-78-158, Mitre Corporation, August 1978. http://stinet.dtic.mil/cgi-bin/GetTRDoc?AD=A059221&Location=U2&doc=GetTRDoc.pdf.
[3] C. G. Girling, "Covert Channels in LAN's," IEEE Trans. Software Engineering, vol. SE-13, no. 2, Feb. 1987, pp. 292–96.
[4] G. Fisk *et al.*, "Eliminating Steganography in Internet Traffic with Active Wardens," *Proc. 5th Int'l. Wksp. Information Hiding*, Oct. 2002.
[5] J. Postel, "Internet Protocol," RFC 0791, IETF, Sept. 1981. http://www.ietf.org/rfc/rfc0791.txt
[6] A. Giani, V. H. Berk, and G. V. Cybenko, "Data Exfiltration and Covert Channels," *Proc. SPIE Sensors, and Command, Control, Commun., and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, Apr. 2006.
[7] V. Gligor, "A Guide to Understanding Covert Channel Analysis of Trusted Systems," Tech. Rep. NCSC-TG-030, National Computer Security Center, Nov. 1993, http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-030.html.
[8] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information Hiding — A Survey," *Proc. IEEE*, special issue on Protection of Multimedia Content, vol. 87, no. 7, July 1999, pp. 1062–78.
[9] G. J. Simmons, "The History of Subliminal Channels," *IEEE JSAC*, vol. 16, no. 4, May 1998, pp. 452–62.
[10] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Secondgeneration Onion Router," *Proc. 13th USENIX Security Symp.*, Aug. 2004.
[11] G. Danezis, "Covert Communications Despite Traffic Data Retention," tech. rep., ESAT, University of Leuven, Jan. 2005, http://homes.esat.kuleuven.be/~gdanezis/cover.pdf
[12] G. Shah, A. Molina, and M. Blaze, "Keyboards and Covert Channels," *Proc. USENIX Security Symp.*, Aug. 2006.
[13] N. Vachharajani *et al.*, "RIFLE: An Architectural Framework for User-Centric Information-Flow Security," *Proc. 37th IEEE/ACM Int'l. Symp. Microarchitecture*, Dec. 2004, pp. 243–54.
[14] N. Feamster *et al.*, "Infranet: Circumventing Web Censorship and Surveillance," *Proc. 11th USENIX Security Symp.*, Aug. 2002.
[15] C. H. Rowland, "Covert Channels in the TCP/IP Protocol Suite," *First Monday*, Peer Reviewed Journal on the Internet, July 1997.
[16] D. V. Forte *et al.*, "SecSyslog: An Approach to Secure Logging Based on Covert Channels," *Proc. First Int'l. Wksp. Systematic Approaches to Digital Forensic Engineering*, Nov. 2005, pp. 248–63.
[17] The Honeynet Project, "Know Your Enemy: Sebek — A Kernel Based Data Capture Tool," tech. rep., 2003, http://www.honeynet.org/papers/sebek.pdf
[18] S. R. White, "Covert Distributed Processing with Computer Viruses," *Proc. 9th Annual Int'l. Cryptology Conf. Advances in Cryptology*, 1989, pp. 616–19.
[19] I. Moskowitz, R. Newman, and P. Syverson, "Quasi-Anonymous Channels," *Proc. Communication, Network, and Information Security (CNIS)*, Dec. 2003.
[20] J. Xu *et al.*, "Prefixpreserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme," *Proc. 10th IEEE Int'l. Conf. Network Protocols*, Nov. 2002.
[21] J. Bethencourt, J. Franklin, M. Vernon, "Mapping Internet Sensors with Probe Response Attacks," *Proc. USENIX Security Symp.*, Aug. 2005.
[22] R. deGraaf, J. Aycock, and M. Jacobson Jr., "Improved Port Knocking with Strong Authentication," *Proc. 21st Annual Computer Security Applications Conf.*, Dec. 2005.
[23] W. Mazurczyk and Z. Kotulski, "New Security and Control Protocol for VoIP Based on Steganography and Digital Watermarking," tech. rep., Institute of Fundamental Technological Research, Polish Academy of Sciences, June 2005, http://arxiv.org/ftp/cs/papers/0602/0602042.pdf
[24] W. Mazurczyk and Z. Kotulski, "New VoIP Traffic Security Scheme with Digital Watermarking," *Proc. Int'l. Conf. Computer Safety, Reliability, and Security (SafeComp)*, Sept. 2006, pp. 170–81.
[25] E. Jones, O. Le Moigne, and J.-M. Robert, "IP Traceback Solutions Based on Time to Live Covert Channel," *Proc. 12th IEEE Int'l. Conf. Networks (ICON)*, Nov. 2004, pp. 451–57.
[26] H. Qu, Q. Cheng, and E. Yaprak, "Using Covert Channel to Resist DoS Attacks in WLAN," *Proc. Int'l. Conf. Wireless Networks*, June 2005, pp. 38–44.
[27] National Computer Security Center, US DoD, "Trusted Computer System Evaluation Criteria," Tech. Rep. DOD 5200.28-STD, National Computer Security Center, Dec. 1985, http://csrc.nist.gov/publications/history/dod85.pdf
[28] J. Millen, "20 Years of Covert Channel Modeling and Analysis," *Proc. IEEE Symp. Security and Privacy*, May 1999, pp. 113–14.
[29] R. A. Kemmerer, "A Practical Approach to Identifying Storage and Timing Channels," *Proc. IEEE Symp. Security and Privacy*, Apr. 1982.
[30] G. J. Simmons, "The Prisoners' Problem and the Subliminal Channel," Proc. Advances in Cryptology (CRYPTO), 1983, pp. 51–67.
[31] S. Craver, "On Public-Key Steganography in the Presence of an Active Warden," *Proc. 2nd Int'l. Wksp. Information Hiding*, Apr. 1998, pp. 355–68.
[32] T. Handel and M. Sandford, "Hiding Data in the OSI Network Model," *Proc. 1st Int'l. Wksp. Information Hiding*, 1996 pp. 23–38.
[33] N. Lucena *et al.*, "Syntax and Semantics-Preserving Application-Layer Protocol Steganography," *Proc. 6th Information Hiding Wksp.*, May 2004.
[34] D. Kundur and K. Ahsan, "Practical Internet Steganography: Data Hiding in IP," *Proc. Texas Wksp. Security of Information Systems*, Apr. 2003.
[35] A. Hintz, "Covert Channels in TCP and IP Headers," 2003, http://www.defcon.org/images/defcon-10/dc-10-presentations/dc10-hintz-covert.ppt
[36] M. Wolf, "Covert Channels in LAN Protocols," *Proc. Wksp. Local Area Network Security (LANSEC)*, 1989, pp. 91–101.
[37] N. B. Lucena, G. Lewandowski, and S. J. Chapin, "Covert Channels in IPv6," *Proc. Privacy Enhancing Technologies (PET)*, May 2005, pp. 147–66.
[38] T. Graf, "Messaging over IPv6 Destination Options," tech. rep., Swiss Unix User Group, 2003, http://grayworld.net/papers/messip6.txt
[39] Z. Trabelsi *et al.*, "Traceroute Based IP Channel for Sending Hidden Short Messages," *Proc. Advances in Information and Computer Security (IWSEC)*, Oct. 2006, pp. 421–36.

[40] K. Ahsan and D. Kundur, "Practical Data Hiding in TCP/IP," *Proc. ACM Wksp. Multimedia Security*, Dec. 2002.
[41] E. Cauich, R. Gómez Cárdenas, and R. Watanabe, "Data Hiding in Identification and Offset IP Fields," *Proc. 5th Int'l. School and Symp. Advanced Distributed Systems (ISSADS)*, Jan. 2005, pp. 118–25.
[42] J. Postel, "Transmission Control Protocol," RFC 0793, IETF, Sept. 1981, http://www.ietf.org/rfc/rfc0793.txt
[43] J. Rutkowska, "The Implementation of Passive Covert Channels in the Linux Kernel," *Proc. Chaos Communication Congress*, Dec. 2004.
[44] S. J. Murdoch and S. Lewis, "Embedding Covert Channels into TCP/IP," *Proc. 7th Information Hiding Wksp.*, June 2005.
[45] C. Abad, "IP Checksum Covert Channels and Selected Hash Collision," tech. rep., UCLA, 2001. http://http://gray-world.net/papers/ipccc.pdf
[46] J. Postel, "User Datagram Protocol," RFC 0768, IETF, Aug. 1980. http://www.ietf.org/rfc/rfc0768.txt
[47] H. Qu, P. Su, and D. Feng, "A Typical Noisy Covert Channel in the IP Protocol," *Proc. 38th Annual Int'l. Carnahan Conf. Security Technology*, Oct. 2004, pp. 189–92.
[48] S. Zander, G. Armitage, and P. Branch, "Covert Channels in the IP Time To Live Field," *Proc. Australian Telecommunication Networks and Applications Conf. (ATNAC)*, Dec. 2006.
[49] M. C. Perkins, "Hiding out in Plaintext: Covert Messaging with Bitwise Summations," Master's thesis, Iowa State University, 2005.
[50] J. Giffin *et al.*, "Covert Messaging Through TCP Timestamps," *Proc. Privacy Enhancing Technologies Workshop (PET)*, Apr. 2002, pp. 194–208.
[51] S. Cabuk, C. E. Brodley, and C. Shields, "IP Covert Timing Channels: Design and Detection," *Proc. 11th ACM Conf. Computer and Communications Security (CCS)*, Oct. 25–29 2004, pp. 178–87.
[52] V. Berk, A. Giani, and G. Cybenko, "Detection of Covert Channel Encoding in Network Packet Delays," Tech. Rep. TR2005-536, Department of Computer Science, Dartmouth College, Nov. 2005, http://www.ists.dartmouth.edu/library/149.pdf
[53] S. J. Murdoch, "Hot or Not: Revealing Hidden Services by Their Clock Skew," *Proc. 13th ACM Conf. Computer and Communications Security (CCS)*, Nov. 2006, pp. 27–36.
[54] J. Postel, "Internet Control Message Protocol," RFC 0792, IETF, Sept. 1981, http://www.ietf.org/rfc/rfc0792.txt
[55] H.-G. Eßer and F. C. Freiling, "Kapazitätsmessung eines verdeckten Zeitkanals über HTTP," Tech. Rep. TR-2005-10, Universität Mannheim, 2005, http://bibserv7.bib.uni-mannheim.de/madoc/volltexte/2005/1136/pdf/tr_2005_10.pdf (in german).
[56] S. D. Servetto and M. Vetterli, "Communication Using Phantoms: Covert Channels in the Internet," *Proc. IEEE Int'l. Symp. Information Theory (ISIT)*, June 2001.
[57] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," RFC 2401, IETF, Nov. 1998. http://www.ietf.org/rfc/rfc2401.txt
[58] A. Galatenko *et al.*, "Statistical Covert Channels Through PROXY Server," *Proc. 3d Int'l. Wksp. Mathematical Methods, Models, and Architectures for Computer Network Security*, Sept. 2005, pp. 424–29.
[59] S. Bhadra, S. Shakkottai, and S. Vishwanath, "Communication Through Jamming Over a Slotted ALOHA Channel," *Proc. 42nd Allerton Conf. Commun., Control, and Computing*, Oct. 2004.
[60] K. Szczypiorski, "HICCUPS: Hidden Communication System for Corrupted Networks," *Proc. 10th Int'l. Multi- Conf. Advanced Computer Systems*, Oct. 2003, pp. 31–40.
[61] T. M. Dogu and A. Ephremides, "Covert Information Transmission through the Use of Standard Collision Resolution Algorithms," *Proc. 3rd Int'l. Wksp. Information Hiding (IH)*, Sept. 1999, pp. 419–33.
[62] S. Li, A. Ephremides, "A Covert Channel in MAC Protocols based on Splitting Algorithms," *Proc. Wireless Commun. and Networking Conf. (WCNC)*, Mar. 2005, pp. 1168–73.
[63] M. Marone, "Adaptation and Performance of Covert Channels in Dynamic Source Routing," tech. rep., Computer Science Department, Yale University, Dec. 2003, http://zoo.cs.yale.edu/classes/cs490/03-04a/michael.marone/paper.pdf
[64] S. Li and A. Ephremides, "A Network Layer Covert Channel in Adhoc Wireless Networks," *Proc. 1st IEEE Conf. Sensor and Ad Hoc Commun. and Networks (SECON)*, Oct. 2004, pp. 88–96.
[65] L. Butti and F. Veysset, "Wi-Fi Advanced Stealth," Proc. Black Hat US, Aug. 2006, http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Veyssett.pdf
[66] C. Krätzer *et al.*, "WLAN Steganography: a First Practical Review," *Proc. 8th ACM Multimedia and Security Wksp.*, Sept. 2006.
[67] M. Bauer, "New Covert Channels in HTTP: Adding Unwitting Web Browsers to Anonymity Sets," *Proc. Wksp. Privacy Electronic Society*, Oct. 2003, pp. 72–78.
[68] R. Fielding *et al.*, "Hypertext Transfer Protocol — HTTP/1.1," RFC 2616, IETF, June 1999. http://www.ietf.org/rfc/rfc2616.txt
[69] L. Bowyer, "Firewall Bypass via Protocol Steganography," Sept. 2002, http://www.networkpenetration.com/protocol_steg.html
[70] A. Dyatlov and S. Castro, "Exploitation of Data Streams Authorized by a Network Access Control System for Arbitrary Data Transfers: Tunneling and Covert Channels over the HTTP Protocol," tech. rep., Gray-World, June 2003, http://gray-world.net/projects/papers/covert_paper.txt
[71] Z. Kwecka, "Application Layer Covert Channel Analysis and Detection," tech. rep., Napier University Edinburgh, 2006. http://www.buchananweb.co.uk/zk.pdf
[72] M. Van Horenbeeck, "Deception on the Network: Thinking Differently About Covert Channels," *Proc. 7th Australian Info. Warfare and Security Conf.*, Dec. 2006.
[73] S. Castro and Gray World Team, "Cooking Channels," *hakin9 Magazine* (www.hakin9.org), May 2006, pp. 50–57.
[74] Anonymous, "DNS Covert Channels and Bouncing Techniques," 2005, http://archives.neohapsis.com/archives/fulldisclosure/2005-07/att-0472/p63_dns_worm_covert_channel.txt
[75] M. Andrews, "Negative Caching of DNS Queries (DNS NCACHE)," RFC 2308, IETF, Mar. 1998, http://www.ietf.org/rfc/rfc2308.txt
[76] D. Kaminsky, "IP-over-DNS using Ozyman," 2004, http://www.doxpara.com/
[77] T. M. Gil, "IP-over-DNS using NSTX," 2005, http://thomer.com/howtos/nstx/
[78] A.-V. T. W. Group *et al.*, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, IETF, Jan. 1996, http://www.ietf.org/rfc/rfc1889.txt
[79] X. Zou *et al.*, "The Research on Information Hiding Based on Command Sequence of FTP Protocol," *Proc. 9th Int'l. Conf. Knowledge-Based Intelligent Info. and Engineering Systems*, Sept. 2005, pp. 1079–85.
[80] J. Postel and J. Reynolds, "File Transfer Protocol," RFC 0959, IETF, Oct. 1985. http://www.ietf.org/rfc/rfc0959.txt
[81] M. Smeets and M. Koot, "Research Report: Covert Channels," Master's thesis, University of Amsterdam, Feb. 2006.
[82] daemon9, "LOKI2: The Implementation," *Phrack Magazine*, vol. 7, no. 51, Sept. 1997.
[83] D. Stødle, "ptunnel — Ping Tunnel," 2005, http://www.cs.uit.no/daniels/PingTunnel
[84] I. Zelenchuk, "Skeeve — ICMP Bounce Tunnel," 2004, http://www.gray-world.net/poc_skeeve.shtml
[85] P. Padgett, "Corkscrew," 2001, http://www.agroman.net/corkscrew/
[86] A. Dyatlov, "Firepass — Is a Tunneling Tool," 2003, http://gray-world.net/pr_firepass.shtml
[87] P. LeBoutillier, "HTTunnel," 2005, http://sourceforge.net/projects/httunnel/
[88] M. Lundström, "MailTunnel," http://gray-world.net/tools/mailtunnel-0.2.tar.gz
[89] D. Denning, "A Lattice Model of Secure Information Flow," *Communications ACM*, vol. 19, no. 5, May 1976, pp. 236–43.
[90] J. Millen, "Information Flow Analysis of Formal Specifications," *Proc. IEEE Symp. Security and Privacy*, Apr. 1981, pp. 3–8.
[91] J. A. Goguen and J. Meseguer, "Security Policies and Security Models," *Proc. IEEE Symp. Security and Privacy*, Apr. 1982, pp. 11–20.
[92] R. A. Kemmerer, "Shared Resource Matrix Methodology: an

Approach to Identifying Storage and Timing Channels," *ACM Transactions on Computer Systems (TOCS)*, vol. 1, no. 3, Aug. 1983, pp. 256–77.
[93] R. A. Kemmerer, "A Practical Approach to Identifying Storage and Timing Channels: Twenty Years Later," *Proc. Annual Computer Security Applications Conference (ACSAC)*, Dec. 2002, pp. 109–18.
[94] R. Kemmerer and P. Porras, "Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels," *IEEE Trans. Software Eng.*, vol. SE-17, no. 11, Nov. 1991, pp. 1166–85.
[95] A. L. Donaldson, J. McHugh, and K. A. Nyberg, "Covert Channels in Trusted LANs," *Proc. 11th NBS/NCSC National Computer Security Conf.*, Oct. 1988, pp. 226–32.
[96] L. Hélouët, C. Jard, and M. Zeitoun, "Covert Channels Detection in Protocols Using Scenarios," *Proc. Wksp. Security Protocols Verification (SPV)*, Apr. 2003.
[97] A. Aldini and M. Bernardo, "An Integrated View of Security Analysis and Performance Evaluation: Trading QoS with Covert Channel Bandwidth," *Proc. 23rd Int'l. Conf. Computer Safety, Reliability and Security (SAFECOMP)*, Sept. 2004, pp. 283–96.
[98] A. B. Jeng and M. D. Abrams, "On Network Covert Channel Analysis," *Proc. 3rd Aerospace Computer Security Conf.*, Dec. 1987.
[99] N. E. Proctor and P. G. Neumann, "Architectural Implications of Covert Channels," *Proc. 15th National Computer Security Conf.*, Oct. 1992, pp. 28–43.
[100] I. S. Moskowitz and M. H. Kang, "Covert Channels — Here to Stay?," *Proc. 9th Annual Conf. Computer Assurance*, 1994, pp. 235–44.
[101] G. R. Malan *et al.*, "Transport and Application Protocol Scrubbing," *Proc. IEEE Conf. Computer Communications (INFOCOM)*, Mar. 2000, pp. 1381–90.
[102] M. Handley, C. Kreibich, and V. Paxson, "Network Intrusion Detection: Evasion, Traffic Normalization," *Proc. 10th USENIX Security Symp.*, Aug. 2001.
[103] A. Singh *et al.*, "Malicious ICMP Tunneling: Defense against the Vulnerability," *Proc. 8th Australasian Conf. Information Security and Privacy (ACISP)*, July 2003, pp. 226–35.
[104] H. Wei-Ming, "Reducing Timing Channels with Fuzzy Time," *Proc. IEEE Computer Society Symp. Research in Security and Privacy*, May 1991, pp. 8–20.
[105] N. Schear *et al.*, "Glavlit: Preventing Exfiltration at Wire Speed," *Proc. 5th Wksp. Hot Topics in Networks (HotNets)*, Nov. 2006.
[106] C. E. Shannon, "A Mathematical Theory of Communications," *Bell Systems Tech. J.*, vol. 27, no. 3, July 1948.
[107] J. K. Millen, "Covert Channel Capacity," *Proc. IEEE Symp. Research in Security and Privacy*, May 1987, pp. 60–66.
[108] I. S. Moskowitz and A. R. Miller, "Simple Timing Channels," *Proc. IEEE Symp. Research in Security and Privacy*, 1994, pp. 56–64.
[109] B. R. Venkatraman and R. E. Newman-Wolfe, "Capacity Estimation and Auditability of Network Covert Channels," *Proc. IEEE Symp. Security and Privacy*, May 1995, pp. 186–98.
[110] J. W. Gray III, "Countermeasures and Tradeoffs for a Class of Covert Timing Channels," Tech. Rep. HKUST-CS94-18, Hong Kong University of Science and Technology, 2000, http://repository.ust.hk/dspace/bitstream/1783.1/25/1/tr94-18.pdf
[111] R. E. Newman-Wolfe, B. R. Venkatraman, "High Level Prevention of Traffic Analysis," *Proc. 7th Annual Computer Security Applications Conf.*, Dec. 1991, pp. 102–09.
[112] B. R. Venkatraman and R. E. Newman-Wolfe, "Transmission Schedules To Prevent Traffic Analysis," *Proc. 9th Annual Computer Security and Applications Conf.*, Dec. 1993, pp. 108–15.
[113] B. Graham *et al.*, "Using Covert Channels to Evaluate the Effectiveness of Flow Confidentiality Measures," *Proc. 11th Int'l. Conf. Parallel and Distributed Systems*, July 2005, pp. 57–63.
[114] J. Giles and B. Hajek, "The Jamming Game for Packet Timing Channels," *Proc. IEEE Int'l. Symp. Information Theory (ISIT)*, June 2000.
[115] D. Bell and L. LaPadula, "Secure Computer Systems: Mathematical Foundation," Tech. Rep. ESD-TR-73-278, Mitre Corp, 1973.
[116] N. Ogurtsov *et al.*, "Experimental Results of Covert Channel Limitation in One-Way Communication Systems," *Proc. Symp. Network and Distributed System Security (SNDSS)*, Feb. 1997.
[117] M. H. Kang and I. S. Moskowitz, "A Pump for Rapid, Reliable, Secure Communication," *Proc. ACM Conf. Computer and Communications Security (CCS)*, 1993, pp. 119–29.
[118] M. H. Kang, I. S. Moskowitz, and D. C. Lee, "A Network Version of the Pump," *Proc. IEEE Symp. Security and Privacy*, May 1995, pp. 144–54.
[119] R. Lanotte *et al.*, "Automatic Covert Channel Analysis of a Multilevel Secure Component," *Proc. 6th Int'l. Conf. Information and Commun. Security (ICICS)*, Oct. 2004, pp. 249–61.
[120] T. Sohn, J. Seo, and J. Moon, "A Study on the Covert Channel Detection of TCP/IP Header Using Support Vector Machine," *Proc. 5th Int'l. Conf. Info. and Commun. Security*, Oct. 2003, pp. 313–24.
[121] E. Tumoian and M. Anikeev, "Detecting NUSHU Covert Channels Using Neural Networks," tech. rep., Taganrog State University of Radio Engineering, 2005, http://grayworld.net/papers/neural_networks_vs_NUSHU.pdf
[122] E. Tumoian and M. Anikeev, "Network Based Detection of Passive Covert Channels in TCP/IP," *Proc. 1st IEEE LCN Wksp. Network Security*, Nov. 2005, pp. 802–09.
[123] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," RFC 1323, IETF, May 1992, http://www.ietf.org/rfc/rfc1323.txt
[124] T. Sohn *et al.*, "Covert Channel Detection in the ICMP Payload Using Support Vector Machine," *Proc. 18th Int'l. Symp. Computer and Information Sciences (ISCIS)*, Nov. 2003, pp. 828–35.
[125] D. Pack *et al.*, "Detecting HTTP Tunneling Activities," *Proc. 3rd Annual Information Assurance Wksp.*, June 2002.
[126] K. Borders and A. Prakash, "Web Tap: Detecting Covert Web Traffic," *Proc. 11th ACM Conf. Computer and Communications Security (CCS)*, Oct. 2004, pp. 110–20.

BIOGRAPHIES

SEBASTIAN ZANDER [S'06] (szander@swin.edu.au) received his Dipl.-Ing. in Technical Informatics from the Technical University of Berlin, Germany in 1999. Since 2006 he is a Research Student at the Centre for Advanced Internet Architectures at Swinburne University of Technology and his research topic is covert channels in computer network protocols. Previously he has worked as a Scientist at Fraunhofer FOKUS in Berlin, Germany and as a Research Fellow at Swinburne University.

GRENVILLE ARMITAGE [M'03] (garmitage@swin.edu.au) earned a B.Eng (Elec)(Hons) in 1988 and a PhD in electronic engineering in 1994, both from the University of Melbourne, Australia. Since 2002 he has been Associate Professor of Telecommunications Engineering and Director of the Centre for Advanced Internet Architectures at Swinburne University of Technology, Melbourne, Australia. He authored "Quality of Service In IP Networks: Foundations for a Multi-Service Internet" (Macmillan Technical Publishing, April 2000) and co-authored "Networking and Online Games — Understanding and Engineering Multiplayer Internet Games" (John Wiley & Sons, UK, April 2006). Associate Professor Armitage is also a member of ACM and ACM SIGCOMM.

PHILIP BRANCH [M'95] (pbranch@swin.edu.au) received a Ph.D. in Engineering from Monash University, Victoria, Australia in 2000. Since 2003 he has been a Senior Lecturer in Telecommunications Engineering at Swinburne University of Technology, conducting research within the Centre for Advanced Internet Architectures. His research interests are in game traffic, network security and lawful interception. He is a co-author of "Networking and Online Games — Understanding and Engineering Multiplayer Internet Games" (John Wiley & Sons, UK, April 2006).