The Silence of the LANs: Efficient Leakage Resilience for IPsec VPNs

Ahmad-Reza Sadeghi[1,2,3], Steffen Schulz[1,2,4], and Vijay Varadharajan[4]

[1] System Security Lab, Technische Universität Darmstadt
[2] System Security Lab, Ruhr-University Bochum
[3] Fraunhofer SIT, Darmstadt
[4] Information and Network Security Research Lab, Macquarie University

Abstract. Virtual Private Networks (VPNs) are increasingly used to build logically isolated networks. However, existing VPN designs and deployments neglect the problem of traffic analysis and covert channels. Hence, there are many ways to infer information from VPN traffic without decrypting it. Many proposals were made to mitigate network covert channels, but previous works remained largely theoretical or resulted in prohibitively high padding overhead and performance penalties.

In this work, we (1) analyse the impact of covert channels in IPsec, (2) present several improved and novel approaches for covert channel mitigation in IPsec, (3) propose and implement a system for dynamic performance trade-offs, and (4) implement our design in the Linux IPsec stack and evaluate its performance for different types of traffic and mitigation policies. At only 24% overhead, our prototype enforces tight information-theoretic bounds on all information leakage.

Keywords: IPsec, VPNs, covert channels, performance trade-offs.

1 Introduction

Virtual Private Networks (VPNs) are popular means for enterprises and organizations to securely connect their network sites over the Internet. Their security is implemented and enforced by VPN gateways that tunnel the transferred data in secure channels, thus logically connecting the remote sites in an isolated network. Abstracted this way, VPNs are increasingly used in scenarios that secure channels were not designed for: to logically isolate networks, providing "networks as a service" in virtualized environments like Clouds, Trusted Virtual Domains, or the Future Internet [1–3]. What is not considered in these scenarios is the long known problem of covert channels.

Covert channels violate the system security policy by using channels "*not intended for information transfer at all*" [4, 5]. While there is a large body of research on covert channels, few works have considered the practical implementation and performance impact of comprehensive covert channel mitigation in modern networks. We believe such work is important for a number of reasons, especially regarding VPNs and network virtualization:

S. Foresti, M. Yung, and F. Martinelli (Eds.): ESORICS 2012, LNCS 7459, pp. 253–270, 2012.
© Springer-Verlag Berlin Heidelberg 2012

(1) Insider Threat: In contrast to end-to-end secure channels, where the endpoints are implicitly trusted, VPNs are also used for logical network isolation and perimeter security enforcement. In this context, the members of a VPN are often not fully trusted, but instead the trust is reduced to central policy enforcement points, the VPN gateways, which should prevent undesired information flows. However, malicious insiders in the LAN may leak information through the VPN gateways using covert channels, thus circumventing the security policy. Examples of such insiders can be actual humans or stealth malware, engaging in industrial espionage, leaking realtime financial transaction data, or disclosing large amounts of data from physically secured institutions (e.g., to Wikileaks).

(2) Traffic Analysis: By analysing traffic patterns and meta-data, it is also possible to infer information about transferred data without assuming a malicious insider [6, 7]. Such "passive" Man-in-the-Middle (MITM) scenarios are becoming more prevalent with network virtualization, allowing co-located, supposedly isolated systems to analyse each other [8]. To mitigate such attacks, a common approach is to consider the maximum possible information leakage by a colluding malicious insider. In limiting this maximum information leakage, covert channel analysis and mitigation thus also affects traffic analysis [9].

(3) Combination with Detection: Although application-layer firewalls and intrusion detection systems are widely deployed, carefully designed covert channels remain hard to detect [10, 11]. In these systems, the adversary chooses a weaker signal and mimics the patterns of regular channel usage. Covert channel mitigation can be useful here to induce noise, forcing the adversary to use a stronger signal and thus facilitate detection. We expect the combination of covert channel mitigation and detection to significantly reduce the performance penalty of covert channel mitigation by allowing less intrusive pattern enforcement.

Contributions. This paper provides for the first time an explicit analysis of covert channels in IPSec based VPNs and a comprehensive set of techniques and mechanisms to mitigate them. We identify and categorize the different types of covert channels and determine their capacity. We develop a framework for mitigation of these covert channels and describe mechanisms and techniques for high-performance covert channel mitigation. In particular, we propose an algorithm for on-demand adjustment of traffic pattern enforcement that increases peak network performance while also reducing overhead during reduced usage. We present a practical instantiation of this framework for the Linux IPSec stack and analyse its performance for different kinds of traffic. In contrast to previous works, which achieve throughput rates in the range of modem speed [9, 12] and taunt the performance impact of proposed mitigation mechanisms [13], our prototype achieves 169 Mbit/s in a 200 Mbit/s VPN link at only 24% overhead.

Outline. After defining the problem of VPN covert channels in Section 2, we discuss efficient covert channel mitigation and performance trade-offs in Section 3. An implementation for the Linux IPsec stack is presented and evaluated in Section 4. We discuss related work in Section 5 and conclude in Section 6.

2 Problem Setting and Adversary Model

In the following we define the problem of covert channels in VPNs. Note that our definition differs from previous, less explicit considerations, which consider communication between legitimate VPN participants and are better described as steganographic channels [14–16]. Although we limit ourselves to VPNs in state-of-the-art IPsec configuration [17], most of our results can be generalized.

2.1 System Model and Terminology

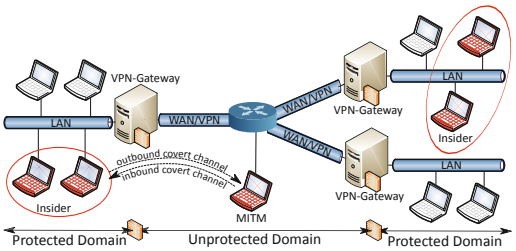
As illustrated in Figure 1(a), we consider a VPN comprised of two or more Local Area Networks (LANs) that are inter-connected over an insecure Wide Area Network (WAN). In our scenario, the security goal of the VPN is not only to provide a secure channel (confidentiality, authenticity, integrity) but also to *confine* communication of LAN hosts to the VPN, i.e., to isolate the *protected* from the *unprotected* domain. VPNs are increasingly used for such logical isolation, to create secure virtualized or overlay networks, or simply enforce perimeter security in large companies [1–3]. This de-facto security goal of isolating the protected from the unprotected domain, and its efficient implementation, is the main focus of this work.

For this purpose, we distinguish *legitimate* channels that transfer and protect user data according to the VPN security policy from *covert* channels that can be used to circumvent this policy. Covert channels exist because the legitimate channel acts as a shared resource between the protected and unprotected domain, exhibiting certain *characteristics* that can be manipulated and measured by different parties. We denote channels from the protected to unprotected domain and vice versa as *outbound* and *inbound* covert channels, respectively.

We measure the security of our system using the Shannon capacity of the covert channels, i.e., the information theoretic limit on the amount of information that can be transferred through them [6]. The covert channel capacity is given in bits per legitimate channel packet (bpp) or, where applicable, in bits per second (bps). The capacity of each covert channel *type* is denoted as C^{type}. The capacities are classified as maximum (m) vs. remaining (r) covert channel rate for inbound (in) vs. outbound (out) covert channels. For example, the maximum capacity of the outbound covert channel based on packet size is denoted as $C^{\text{PktSize}}_{m,out}$, or as $C^{\text{PktSize}}_{r,out}$ after countermeasures have been applied. The remaining *aggregated* inbound and outbound covert channel rates are denoted as $\hat{C}_{r,in}$ and $\hat{C}_{r,out}$, respectively.

2.2 Adversary Model

The adversary controls one or more compromised hosts in the LAN sites as well as an active MITM in the WAN. We refer to the LAN hosts controlled by the adversary as (malicious) *insiders*, regardless of whether they are controlled by actual humans or stealth malware. The adversary's goal is to establish a



(a) A VPN with three LAN sites. The adversary aims to exchange information between the MITM and malicious insiders using covert channels.

Class	Type	Capacity C_m in bpp	
		Outbound	Inbound
storage	ECN	2	1
	DS	6	6
	Flags	1	-
timing/ channel-logic	PktSize	8.4	-
	IPD	≥ 1	≥ 1
	PktOrd	-	> 6.58
	PktDrop	-	1
	PMTUD	-	0.13
amplify	DestIP	$\log_2(N)$	-

(b) Inbound and outbound covert channels capacities for an IPsec VPN with $N + 1$ endpoints.

Fig. 1. Problem scenario: A complex VPN with multiple identified covert channels

communication channel between the MITM and one or more possibly colluding malicious insiders, as illustrated in Figure 1(a). This would allow the adversary to send instructions to the insiders or to leak information from the protected to the unprotected domain, breaching the perimeter security of the VPN. For this purpose, we assume a state-of-the-art IPsec configuration with authenticated encryption using ESP in tunnel mode [17], and the cryptographic primitives and keys of the VPN are securely enforced by the VPN gateways. However, the legitimate VPN traffic can be manipulated by malicious parties in the protected and unprotected domains to exchange information that "survives" these packet transformation enforced by the VPN gateways.

Unfortunately, no systematic approach is known for identifying network covert channels apart from exhaustive search, and the categorization as storage or timing channels can be ambiguous [5]. We used a comprehensive analysis on the IPsec specification and related work on covert channels in network protocols (cf. Section 5), as well as source code analysis and testing[1] to identify potential covert channels in IPsec VPNs. IP-Tunneling and authenticated encryption by the IPsec gateways greatly simplified this problem, as none of the protocol headers that the MITM can read or modify (i.e., the outer IP and Encapsulated Security Payload (ESP) header) are directly available to the LAN hosts.

In total, we have identified only eight covert channels. As shown in Figure 1(b), the available covert channels comprise three storage-based channels based on fields in the outer IP header (ECN, DS, Flags) and five timing-based covert channels that manipulate Inter-Packet Delay (IPD), packet order (PktOrd), WAN capacity (PktDrop), and Path MTU Discovery (PMTUD). The remaining characteristic of the respective destination LAN of a packet (DestIP) does not constitute a covert channel in its own right but can act as amplification of other covert channels. A detailed discussion of the covert channels we identified in IPsec VPNs is available in the full version [18].

[1] Specifically, we examined the IPsec implementations of the current Linux 2.6.32 to 2.6.38 and OpenBSD 4.7 to 4.8 releases.

We emphasize that some of these channels are implementation dependant, e.g., the treatment of ECN header flags or PMTUD at the VPN gateway, while others (IPD, PktSize, PktOrd) are generic problems faced by all packet-oriented channels. While we are confident to have identified all covert channels, we cannot account for all possible implementations and interpretations of IPsec. Hence in this paper we limit our considerations to the identified attack vectors.

3 Covert Channel-Resilient IPsec

In this section we present the design of a high-performance covert channel-resilient IPsec, i.e., a system with low, known covert channel capacity and high throughput. We present novel or improved techniques for efficient covert channel mitigation in Section 3.1. Section 3.2 considers the performance of different mitigation strategies, introducing on-demand performance trade-offs. Finally, we derive the remaining aggregated inbound and outbound covert channel capacities of the system in Section 3.3.

3.1 Covert Channel Mitigation

In the following we present and improve efficient mitigation mechanisms for each of the covert channels identified in Section 2.2.

Packet Size (PktSize). The packet size characteristic is usually addressed by padding packets to maximum size or assuming them to be of constant size [6]. However, as the product throughput = pkt_size · pkt_rate is constant for a given link, enforcement of small packet sizes can reduce the load per packet significantly, allowing higher packet rates and more simultaneous connections.

It was previously proposed to allow multiple alternate packet sizes [19], but then the ratio between packets of different sizes creates another covert channel. Mode Security [20] was proposed to manage the switching between different enforcement modes and audit such a remaining covert channel. However, real network traffic is often mixed, i.e., packet streams using different packet sizes are often transmitted at the same time. Moreover, the enforcement of small packet sizes is problematic for IP protocols: With Path MTU Discovery (PMTUD), the connection endpoints quickly detect and adapt to the maximum allowed packet size of an IP route, but only slowly recover to a larger MTU using a conservative trial-and-error approach. This active adaption also makes it harder for the VPN gateways to estimate the actual demand for larger packets.

We address these problems by combining packet padding with transparent fragmentation and multiplexing, mechanisms that were previously only considered for traffic obfuscation [21]. Packet fragmentation *within* IPsec allows us to efficiently and transparently enforce various packet sizes at the gateway without influencing the channel's Path MTU (PMTU). This is different from regular IP fragmentation before or after IPsec processing, which results in visible fragments either on the LAN or WAN sides that could again be used as covert channels.

On the other hand, packet multiplexing can be used to reduce packet padding overhead, and in general to reduce the IPsec encapsulation overhead (ESP, IP).

When working with mixed traffic, the sender gateway first fragments large packets and then attempts to multiplex small packets or fragments into the padding area of previously processed packets that are still in the packet buffer. At the receiving gateway, packets are first de-multiplexed and then defragmented.

Inter-Packet Delay (IPD). The covert channel based on IPDs and its mitigation were subject of several previous works (e.g., [6, 10, 22–24]). In theory, it is easily eliminated by enforcing a fixed IPD at the VPN gateway, inserting dummy packets when no real packets are available [24]. However, due to the very high packet rates in modern networks, even short periods of non-optimal enforcement of IPDs (and thus packet rate) at the VPN gateway quickly result in packet loss due to packet buffer overflows or network congestion. This is particularly critical for Internet protocols, where packet loss triggers congestion avoidance, degrading overall throughput independently of the packet rate enforced by the VPN gateways. The effect can be partly mitigated with large packet buffers; however, large buffers can also create high packet delays, degrading network responsiveness [26]. Also, the optimal enforced packet rate can be very large in modern networks, creating a high computational overhead for the time-synchronous packet processing. For example, to saturate a 100 Mbit/s link with 200 byte packets, an average IPD of $\frac{500\ byte}{100 \cdot 10^6\ byte/s} = 2\mu s$ should be enforced. Finally, one must consider inaccuracies in the timing enforcement that appear at high system loads [23, 25]: Since high activity on the LAN interface can influence the system load of the gateway, a LAN host may induce inaccuracies in the IPD enforcement of the gateway that can again be measured by the Man-in-the-Middle (MITM), yielding $C_r^{\mathrm{IPD}'} = 0.16$ bps [9].

We have implemented the traffic reshaping inside the Linux kernel, using the modern High-Precision Event Timer (HPET) infrastructure for packet scheduling with nanosecond resolution. This substantially reduces the overhead of context switching and buffering, allowing an IPDs in the range of microseconds rather than several milliseconds (e.g., [9, 12]) and noticeably improves throughput and responsiveness. To maintain good system performance at even higher packet rates we use packet bursts, i.e., we translate very low IPDs into bursts of multiple packets at correspondingly larger delays. For optimal packet buffering our system adjusts the buffer size depending on the currently enforced IPD. This prevents long delays at low rates while allowing generous buffering at high rates.

To address the problem of timing inaccuracies, we use the high resolution of the HPET timers to monitor and actively *compensate* for timing inaccuracies in randomized IPD enforcement. Specifically, we exploit the fact that determining timing inaccuracies during randomized IPD enforcement is harder for the remote MITM than for the local system. The adversary always requires significantly more measurements to first detect the variance of the random IPD enforcement and then the inaccuracy in the enforced variance [23], while the VPN gateway itself can directly compare the intended versus actual packet sending time. Hence, the gateway can approximate the current inaccuracy faster,

requiring less measurement samples. Given this knowledge of unintended change in IPD variance, we let the VPN gateways compensate for the enforcement inaccuracy by dynamically compensating the variance of the IPD enforcement. This prevents the adversary from ever measuring the actual inaccuracy, eliminating the timing channel ($C_r^{\mathrm{IPD}} = 0$). However, further evaluation with specialized network hardware is needed to confirm (the non-existance of) this effect.

Packet Order (PktOrd). Sequence numbers in protocol headers have been used before to create a covert or steganographic channel based on packet reordering [16, 27]. However, in contrast to previous works we can eliminate this channel in the VPN scenario using the IPsec anti-replay window and secure sequence numbers in Encapsulated Security Payload (ESP).

IPsec implementations maintain a bitmap of the last r seen and unseen sequence numbers so that replay attacks within the window size can be detected and older packets discarded. To eliminate communication through packet reordering, we propose to implement this window as a packet buffer, where new packets are inserted *sorted* by their ESP sequence number and leave the buffer as the window advances. As a result, all packets forwarded from the VPN gateway into the LAN are ordered and the covert channel is eliminated: $C_{r,in}^{\mathrm{PktOrd}} = 0$.

Unfortunately, the approach is problematic for low packet rates, since the window may advance slowly and individual packets are not forwarded fast enough. We solve this issue by establishing a certain maximum IPD (e.g., 50ms) at the sender and assure that at least r dummy packets are sent by a gateway before a connection is stopped. These constraints are necessary in any case to assure network responsiveness and hide short periods of inactivity.

Packet Drops (PktDrop). In general, it appears impossible to eliminate covert channels based on packet dropping in the WAN. Mitigation with error correction codes is expensive and easily defeated by dropping even more packets. Instead, we propose to mitigate the channel by injecting noise, by *increasing* packet loss proportionally to the actual packet loss.

Specifically, the gateways maintain a buffer p of size d. At the sender gateway, packets are buffered in p and their order is randomized before encapsulation. At the receiver gateway, the packets are again collected in p and the number of dropped packets i is determined based on their ESP sequence number. If $i > 0$, the gateway drops another j packets from the current buffer, such that $i+j = 2^x$, where $1 < x \leq \log_2(d)$, and forwards the remaining packets after randomizing their order once again. As a result, the MITM can choose the overall number of packets to be dropped but cannot select which packets to drop, resulting in a symbol space of $\log_2(d)+1$ packets per window d. The remaining covert channel capacity is then $C_{r,in}^{\mathrm{PktDrop}} = \frac{1}{d} \cdot \log_2(\log_2(d) + 1)$ bpp.

Similar to the above packet re-ordering mitigation, the inbound packet buffer at the receiving gateway is problematic for very low traffic rates and requires similar restrictions to assure a steady stream of (dummy) packets. The implementation can be simplified at the cost of a slightly higher covert channel rate

by removing the randomization buffer at the sending gateway and re-using the anti-replay window for dropping the additional j packets.

Path MTU Discovery (PMTUD). To our knowledge, no previous work considered the possibility of covert channels based on PMTUD, in particular with respect to VPNs. Since PMTUD is critical for good network performance, we do not disable it but instead mitigate the channel by enforcing limits on the rate and values that are propagated by the VPN gateways into the LAN.

In particular, we limit the possible PMTU values by maintaining a list of common PMTU values and only propagate the respective next lower PMTU to the LAN. Such common PMTUs values can be established on site or can be derived from previously proposed performance optimizations for PMTUD [28]. The rate limitation of PMTU propagation is problematic in general, as a lack of MTU adaption will lead to packet loss. However, in our case the current PMTU is always known to the trusted VPN gateways, which can then use the transparent fragmentation feature from PktSize enforcement to translate between LAN and WAN packet sizes. Considering the 10 most common PMTUs and an average interval of, e.g., 2 minutes [28] between propagation of PMTU changes, our measures reduce the covert channel rate to less than $C_{r,in}^{\mathrm{PMTUD}} = 0.02$ bps.

Storage-Based Channels (ECN, DS, Flags). The storage-based covert channels exploiting the Explicit Congestion Notification (ECN), Differentiated Services (DS) and IPv4 Flags handling of IP/IPsec are easily eliminated by re-setting the respective fields of the outer IP header at encapsulation and ignoring them during decapsulation. Normalizing the IPv4 Flags field is unproblematic as en-route fragmentation is deprecated in IP. However, eliminating the ECN and DS covert channels disables these performance optimizations in the WAN.

3.2 Mitigation Policies and Performance

In this section, we discuss different covert channel mitigation policies that can be enforced using the techniques described in Section 3.1. We start by discussing the problems of previously proposed Fully Padded Channel and Mode Security approaches, and then propose a new system for on-demand, dynamic adaption of the enforced channel characteristics. We focus on the IPD and PktSize enforcement mechanisms, since they have by far the highest performance impact.

Fully Padded Channel. When applied without any performance trade-offs, the mitigation mechanisms described in Section 3.1 result in a *fully padded channel*: The WAN packet stream is constantly padded to the maximum desired throughput rate and packet size. However, this mitigation policy has several disadvantages: (1) The system must compromise between high throughput and responsiveness, likely opting to enforce maximum packet sizes to reduce fragmentation overhead; (2) the maximum (desired) network load is constantly enforced in both directions, reducing overall performance due to network congestion; (3) TCP/IP congestion avoidance algorithms do not work, since any rate

throttling is compensated by additional channel padding. In case of temporary reductions in WAN capacity, this leads to repeated packet loss and throttling, until the network is not usable anymore. Hence, the fully padded channel policy is unfit for practical use, except in private/dedicated physical infrastructures.

Mode Security. Mode Security is a generic scheme for trading covert channel-resilience against system performance. This is done by organizing system operation in a set of alternative *operation modes* that can be switched at a certain rate [20]. The current operation mode is then selected such that performance penalty and/or overhead produced by the covert channel mitigation is minimized. Since the operation mode is typically adapted depending on the actually required usage, the adaption itself may be exploited as a covert channel. In this case, the covert channel capacity can be given as $C_{out}^{\text{ModeSec}} = R \cdot \log_2(M)$, where M is the number of operation modes and R is the maximum rate at which the operation mode can be changed (transition rate).

Mode Security was used to estimate the theoretic network overhead and covert channel capacity [6]. However, this assumes an algorithm that can determine the optimal operation mode to switch to. To the best of our knowledge, no practical implementation and evaluation of this mechanism exists; in particular, no strategies have been proposed to automatically determine and apply the optimal operation mode in the face of often unpredictable traffic, with exponential rate increases and congestion avoidance algorithms. In fact, our attempts to directly apply Mode Security to on-demand covert channel mitigation resulted in poor performance, with TCP throughput benchmarks becoming stuck at very low packet rates or completely losing the connection.

On-Demand Mode Security Management. An algorithm for on-demand adaptation in network covert channel mitigation must accommodate multiple conflicting constraints. It must quickly react to changes in channel usage to elude congestion avoidance algorithms, yet the amount of possible mode changes should be minimal. Moreover, the employed packet queue should buffer packet bursts at various average packet rates, yet react quickly when the current average rate is overused by dropping individual packets. We address these conflicts using the following regulation mechanisms:

Token Bucket Filter. We generalize the transition rate R of the Mode Security paradigm to a token bucket filter [29]. Tokens are generated at a fixed rate R and each mode transition consumes a token from the token bucket. This allows us to "save up" unused mode transitions in form of tokens and consume them on demand, at temporarily higher rates than R. The amount of cached tokens is limited by the token bucket size and the *average transition rate* $\bar{R}$ is bound by the rate R at which new tokens are generated. Thus, the token bucket filter allows us to immediately react to changes in network usage, before connection throttling kicks in or network delays become noticeable. Further, the token bucket status may influence and optimize decisions on the operation mode to be enforced.

```
1  while true do
2  |  (r_LAN, r_frag, r_mplex) ← get-stats()
3  |  r_opt ← r_LAN + r_frag − r_mplex
4  |  r_avg ← 0.1 · r_opt + 0.9 · r_avg
5  |  case r_opt > 0.9 · r_now
6  |  |  r_amp ← (r_max − r_opt)/t_num
7  |  |  r_new ← r_opt + ½ r_quant + r_amp
8  |  case r_now > 1.1 · r_avg ∧ t_num > t_dec
9  |  |  r_new ← r_avg
10 |  r_new ← quantatize(r_new, r_quant)
11 |  sleep(ival)
```

(a) Simplified pseudo-code for dynamic packet rate adjustment in steps of r_{quant}.

Type	**Max. Capacity** C_m **in bpp**		**Rem. Capacity** C_r **in bps**	
	Outbound	Inbound	Outbound	Inbound
ECN	2	1	0	0
DS	6	6	0	0
Flags	1	-	0	-
PktSize	8.4	-	0	-
IPD	≥ 1	≥ 1	0	0
PktOrd	-	> 6.58	-	0
PktDrop	-	1	-	≤ 5
PMTUD	-	0.13	-	0.02
DestIP	$\log_2(N)$	-	$\log_2(N)$	-
Overall	> 18.4	> 15.64	0	≤ 5.02

(b) Maximum and remaining covert channel capacities for VPNs with $N + 1$ endpoints.

Fig. 2. Design of high-performance covert channel mitigation

Aggressive Increase. Network throughput is scaled mainly based on its packet rate r, with typically exponential rate increase until the first network bottleneck is detected. While the optimum WAN packet rate r_{opt} is easily calculated based on the currently observed LAN rate r_{LAN}, fragmented and multiplexed packets (r_{frag}, r_{mplex}), the derivation of the next enforced packet rate r_{new} is more involved, as shown in Figure 2(a).

To adequately consider exponential rate increases without requiring too frequent changes to r_{now}, our rate increase phase is designed to constantly *overestimate* the current optimal packet rate r_{opt}, by increasing r_{now} as soon as it is approached by r_{opt} (cf. Figure 2(a), Line 5). Combined with buffering and short monitoring intervals $ival \approx 200$ms, this approach successfully eludes congestion avoidance algorithms and prevents undesired throughput throttling. However, the overestimation should also not be too large, as it directly affects the padding overhead and can also reduce the inbound traffic rate due to the imposed network load. Moreover, all stored tokens may be used up before a reasonably high packet rate $r_{\text{now}} \approx r_{\text{max}}$ is reached, resulting in bad performance until new tokens are generated. Hence we also include an amplification mechanism that increases the rate r_{opt} in larger steps r_{amp}, depending on the currently available amount of tokens r_{num} (Line 6f.). This prevents the system from becoming "stuck" at low packet rates, at the cost of potentially high padding overhead in cases where such amplification was not required.

Conservative Slowdown. When putting the WAN channel in a state of decreased performance, we must take care that sufficient transition tokens are available to adequately adapt to a possible subsequent usage increase as outlined above. In contrast to the aggressive rate increase policy, any reduction in the enforced traffic rate is therefore delayed until a certain amount of tokens t_{dec} have been collected in the token bucket. Moreover, to reduce the impact of short-term fluctuations in the packet rate, the rate is only reduced based on the longer-time average traffic rate r_{avg}, as shown in Figure 2(a) Line 8f. Overall, the described

approach saves up tokens in the "slowdown" phases while aggressively spending them in the "increase" phase, creating an equilibrium around t_{dec} and r_{avg}.

Dynamic Queue Size with RED. When dynamically adjusting the overall throughput of the WAN channel, we must also adjust the size of the packet queue accordingly. At small rates, a lot of packets may build up in a large queue, leading to large delays and timeouts. Similarly, a small queue is not effective at supporting a channel with high packet rates. Hence, we dynamically adapt the queue size based on the desired maximum buffering delay and the currently enforced packet rate. Eventually, the WAN channel or its enforcement policy may also reach a point where further rate increases are not possible. In this case, the endpoints should be notified of the current throughput limit as quickly as possible, without dropping several packets at once due to full buffers. We achieve this by deploying Random Early Detection (RED) [30] as the packet queue's dropping policy, so that packets are randomly dropped with increasing queue usage.

We implemented several variations of this approach and evaluated the effect of different parameters on the short-term and long-term usage adaption. The achieved performance and adaptation behavior is presented in Section 4.3.

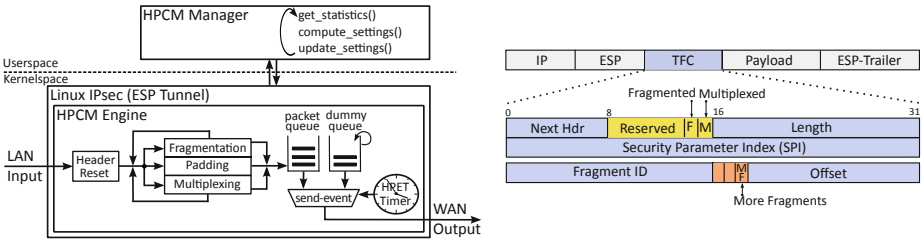
3.3 Remaining Covert Channel Capacity

In the following we summarize the identified covert channels and derive the aggregated remaining covert channel capacity of our covert channel-resilient VPN.

Unfortunately, it is not possible to give all the covert channel rates in a closed form and with comparable units. Several covert channels also depend on additional parameters like network PMTU or minimum WAN packet rate. To provide a reasonable overview of the overall effectiveness of the covert channel mitigation, we have used the capacity estimations derived in the examples of Section 3.1, assuming a state-of-the-art IPsec VPN configuration (cf. Section 2).

Figure 2(b) lists the individual covert channel capacities for the unmitigated (C_m) and mitigated (C_r) case. Considering that today's networks easily transmit several thousand packets per second, i.e., 1 bpp $\gg$ 1 bps, our system results in significant improvements over standard IPsec. In fact, all outbound covert channels are completely eliminated, except for the DestIP channel. However, as explained in Section 2.2, the DestIP characteristic does not by itself constitute a covert channel but can only be used to amplify other channels. Hence, the overall remaining covert channel capacity is given by $\hat{C}_{r,out} = C_{out}^{\text{ModeSec}} \cdot C_{out}^{\text{DestIP}}$.

For the less critical inbound covert channels (e.g., control channels for stealth malware), only the channels based on PMTUD and PktDrop remain. The PktDrop covert channel has the highest impact with $C_{r,in}^{PktDrop} \leq 5$ bps and is easy to exploit. Since the PMTUD channel could be exploited at the same time, their capacities must be added up: $\hat{C}_{r,in} = C_{r,in}^{\text{PktDrop}} + C_{r,in}^{\text{PMTUD}} = 5.02$ bps.



(a) Architecture of our Linux prototype. (b) TFC encapsulation protocol.

Fig. 3. Implementation architecture and encapsulation protocol

4 Practical Covert Channel Mitigation with Linux

In this section we describe the instantiation of our system based on the Linux IPsec stack and analyse the achieved network performance and behavior.

In our prototype implementation and evaluation we only consider the mitigation of *outbound* covert channels, since information leakage from the protected to the unprotected domain is usually considered more critical (e.g., consider Bell-LaPadula [31]). Moreover, from our discussions in Section 3 it is clear that outbound covert channel mitigation is more efficient, as it requires less buffering and processing but is more effective in reducing the covert channel capacity.

4.1 Architecture and Implementation Details

We have implemented a High-Performance Covert Channel Mitigation (HPCM) system inside the IPsec stack of the Linux kernel. The architecture and encapsulation protocol are based on the Traffic Flow Confidentiality (TFC) project, a system for probabilistic traffic flow obfuscation and re-routing in IPsec [21]. We revised and extended TFC to support High-Precision Event Timers (HPETs), fragmentation, multiplexing and dummy packet generation that is indistinguishable from real traffic payloads, elimination of storage-based covert channels in the encapsulation headers and, most importantly, a interface for packet processing statistics and flexible policy enforcement in userspace. The resulting architecture is illustrated in Figure 3(a). In kernelspace, the HPCM Engine processes packets as part of the IPsec subsystem, rewriting problematic header fields and enforcing the currently desired size and IPD constraints as described in Section 3.1. In userspace, the HPCM Manager collects processing statistics from the enforcement engine and combines them with the observed inbound LAN traffic to determine the optimal enforcement parameters, as presented in Section 3.2.

For flexible packet padding and rerouting, TFC deploys its own encapsulation protocol with explicit signalling flags and length header [21]. To also support transparent fragmentation and multiplexing within our system, we extend the TFC protocol with an optional 32 bit fragmentation extension header as illustrated in Figure 3(b). The employed header format is compatible with the IPv4

Table 1. Throughput and transaction rate for regular and modified IPsec VPN

Benchmarks	**No Padding**			**Fully Padded**		**On-Demand**
	IP	ESP	TFC	1422	800	$\bar{R} = 10^{-1}$s
LAN Throughput (Mbit/s)	570	201	175	58	75	169
TCP Transaction Rate (Hz)	1756	1462	1364	611	740	532
LAN/WAN Overhead (%)	0	10	13	(73)	(61)	≈24
Relative Throughput (%)	283	100	87	28	37	84

header format, allowing us to reuse the existing IP defragmentation framework of Linux for defragmenting TFC payloads. Also note that the additional Security Parameter Index (SPI) field is only required due to restrictions of the Linux IPsec framework, and could be removed to reduce the TFC protocol overhead.

4.2 Testbed and Raw Performance

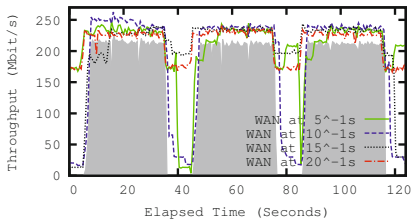
In this section we describe the performance achieved by our prototype in terms of network throughout, transaction rate (i.e., roundtrip time) and protocol overhead. Our testbed corresponds to the VPN scenario in Figure 1(a), except that we use only two LAN sites with one physical host per LAN. The Man-in-the-Middle (MITM) is implemented as an Ethernet bridge between the two VPN gateways, allowing reliable observation of all transmitted packets. For our evaluation, the MITM is completely passive and only used to provide independent performance measurements of the WAN. All hosts are 3.2 Ghz Intel Core i5-650 machines, equipped with two Intel PCIe GBit network cards and 4GB system memory. All network links are established at full-duplex GBit/s speed.

We have used the Netperf[2] benchmarks `TCP_STREAM` and `TCP_RR` to measure the maximum TCP throughput and transaction rate between the LAN sites. By comparing LAN and WAN throughput, we can determine the protocol overhead of the covert channel mitigation, including dummy packets and packet padding.

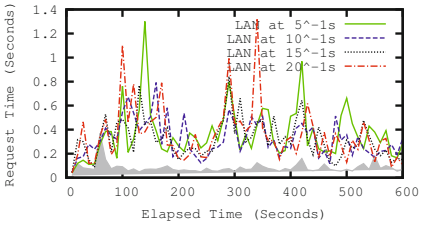
We list the overall performance results in Table 1. The first two columns show the testbed performance for raw IP (plain-text) transmission and IPsec ESP tunneling. With 570 Mbit/s, the raw transmission does not reach the expected GBit throughput, likely due to deficient hardware or drivers. As the LAN hosts and the MITM measure the same IP payloads, there is no LAN/WAN overhead. With 201 Mbit/s, the throughput of a standard IPsec ESP tunnel is already notably slower due to 10% protocol overhead but mainly computational constraints of the VPN gateways. As our covert channel mitigation is an extension of this ESP tunnel configuration, we normalize the relative throughput to 100%.

For reference and confirmation of the expected implementation overhead of our prototype, we next evaluated the raw performance of our HPCM Engine compared to the standard IPsec ESP tunnel. The third column "TFC" of Table 1 lists the achieved network performance when tunneling TFC inside ESP with with all covert channel mitigation techniques *disabled*. The overall LAN/WAN overhead of 13% (or 3% when compared with the ESP tunnel) is the result of the 8 to 12 byte TFC protocol encapsulation plus some computational overhead.

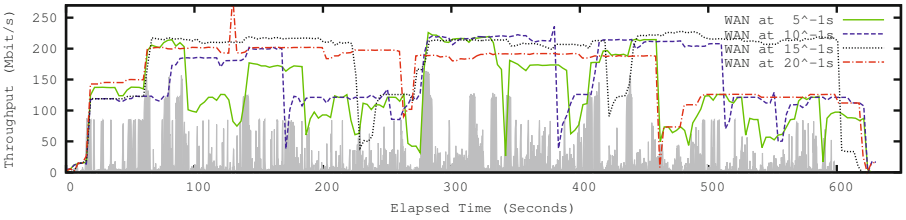
[2] `http://www.netperf.org`



(a) WAN adaption to repeated TCP load.



(b) HTTP request delay in mixed traffic.



(c) WAN adaption to pseudo-random web traffic and downloads.

Fig. 4. Behavior of mode adaptation for different token generation rates R. For reference, the grey filled graph shows LAN performance without time/size padding.

4.3 Covert Channel Mitigation Performance

We now describe the behavior and performance of different mitigation policies.

The fourth and fifth column of Table 1 show the performance of a "fully padded channel", enforcing packet sizes of 1422 and 800 bytes at the maximum possible packet rate. For this purpose, we first measured the maximum bi-directional throughput of the VPN channel (201 Mbit/s per direction) and then selected the desired packet rate (inverse IPD) such that the bidirectional channel capacity is almost[3] saturated. We then again measured the maximum (uni-directional) throughput and roundtrip time. As shown in Table 1, the fully padded channel configuration achieves rather poor performance in both configurations, reaching only 37% and 28% of the ESP tunnel throughput. Observe that the enforcement of 800 byte packet size achieves higher transaction rate as well as higher throughput. We believe this is due to the overhead of padding TCP acknowledgements to maximum packet size.

We have also implemented and tested an instantiation of our on-demand mode security management scheme presented in Section 3.2. As shown in the last column of Table 1, the employed mode adaption heuristics reach almost the same maximum throughput as the raw TFC encapsulation without time/size padding (169 Mbit/s vs. 175 Mbit/s). The LAN/WAN overhead is slightly higher (24% vs. 13%) and the transaction rate rather low. The high throughput despite relatively high overhead is explained by the mode adaption behavior: As shown in Figure 4(a), the WAN channel adapts to the maximum possible

[3] As explained in Section 3.2, it is critical that the link is not fully saturated since congestion leads to packet loss and congestion avoidance does not work.

throughput, but suffers overhead in the rate increase and especially rate decrease phases. As desired by our design in Section 3.2, the main impact of reduced token regeneration rates $\bar{R} \leq 15^{-1}s$ in Figure 4(a) is the increased overhead in the intervals *between* TCP loads, when the rate is not decreased to save tokens.

Finally, we have investigated the ability of our on-demand mode security management to adapt to random, highly heterogeneous traffic patterns one would expect from a VPN with many users. We used Tsung, a traffic load testing tool[4], to record several HTTP sessions in our network, partly also including larger ($\approx$ 60 MB) HTTP downloads. We then configured one of our testbed LANs to act as Internet gateway for the other LAN and used Tsung to replay the recorded HTTP sessions in a pseudo-random fashion with 60 to 80 simultaneous users. Figure 4(c) shows how the WAN traffic enforcement for four different token regeneration rates $\bar{R}$ dynamically adapts to the LAN usage (grey filled). For $\bar{R} \leq 15^{-1}s$, only the larger peaks in LAN usage influence the WAN traffic enforcement, reducing information leakage at the cost of padding overhead. As shown in Figure 4(b), the mean duration of responding to individual HTTP requests is kept within reasonable limits. However, in contrast to unpadded traffic (grey filled) the accumulated request delays become noticeable to the user.

In the presented configuration, our mode adaption algorithm switches packet sizes in steps of 100 bytes and packet rates in steps of 1000 packets per second. Considering the maximum WAN packet rate of about 250.000 packets/s, we can derive $C_{r,out}^{\text{ModeSec}} = \bar{R} \cdot \log_2(\frac{1500}{100} \cdot \frac{250000}{1000}) = \bar{R} \cdot 11.87$ and an overall outbound covert channel capacity of, e.g., $\hat{C}_{r,out} = 0.6$ bps for $\bar{R} = 20^{-1}s$ and $N = 1$.

5 Related Work

Several works consider the problem of covert channels and covert channel mitigation in Internet protocols [32, 33], yet we know of no works that specially discuss the problem of covert channels in IPsec. The covert channels we identify in IPsec are generally known, but we found no previous discussion of the PMTUD channel. Additionally, the PktSize [19], PktSort [11, 27] and DestIP [19] characteristics have different impact in IPsec, and the discussion of storage-based covert channels in the IPsec specification [14] proved to be inaccurate.

Although the IPD-based covert channel is generally well-known [6, 10, 19, 22, 34], the problem of inaccuracies in timing enforcement during increased system load remained unsolved [9, 23]. We consider this complication in our design in Section 3.1 and present a compensation mechanism that detects and compensates unintended timing inaccuracies. Also, while most works simply assume that packets are of constant size [24] or padded to the maximum desired size [19, 33], our adoption of multiplexing and fragmentation enables flexible packet size enforcement. The combination of different mitigation techniques makes our implementation the first prototype for comprehensive covert channel mitigation.

[4] `http://tsung.erlang-projects.org`

Regarding performance trade-offs, Mode Security was proposed as a general approach to adapt to resource usage by switching between different operation modes [20]. A similar approach called Traffic Stereotyping was proposed for networks [19]. To our knowledge, there is only one system that uses Mode Security to optimize covert channel mitigation, which aims to provide sender anonymity based on dynamic re-routing and IPD enforcement [6,24]. They assume a trusted network stack on each network endpoint and a periodic global negotiation to achieve an equalized traffic matrix [24]. A performance analysis was done based on statistics collected from a medium-sized network [12]; however, no actual performance measurements of their system have been provided and the problem of determining the optimal enforcement mode was left unsolved. Alternatively, NetCamo [34] requires its endpoints to explicitly request their delay and throughput demands beforehand. We extend on these works by proposing a practical algorithm to determine the optimal operation mode on-demand. As we do not aim for sender-anonymity, we do not require mix-networks and various attacks on mixes do not apply to our approach (e.g., [35,36]).

In contrast to probabilistic traffic obfuscation schemes such as HTTPOS [37] or Traffic Morphing [38], our framework enforces an information-theoretic boundary for the maximum information leakage. As argued in Section 1, covert channel detection schemes such as [39,40] are complementary to our work and should be used where mitigation is costly, e.g., for the PktSort and PktDrop characteristics.

While we know of no practical performance measurements for comprehensive covert channel elimination, an overhead of 45%-56% was reported solely for obfuscating the packet size in website traffic [7,38].

6 Conclusion and Future Work

We have motivated the problem of covert channels in Virtual Private Networks (VPNs) and presented the design, implementation, and performance of a covert channel-resilient VPN. We identified several covert channels and presented new countermeasures. We have investigated the problem of on-demand adaption of operation modes and presented an implementation for comprehensive, high-performance covert channel mitigation in the Linux IPsec stack. Our evaluation shows that on-demand rate adaption is feasible and practical even for highly unpredictive traffic. In more predictable throughput benchmarks, our system achieves remarkable 169 Mbit/s in a 201 Mbit/s VPN connection (84%).

As part of our future work, we will consider the effectiveness of alternative trade-off and normalization strategies. Furthermore, we aim to investigate the impact of inaccuracies in IPD enforcement.

Acknowledgments. We thank Amir Herzberg, Haya Shulmann and Thomas Schneider for their insightful comments and review of earlier versions of this work.

References

1. Cohesive Flexible Technologies: VPN-Cubed (2012), `http://cohesiveft.com`
2. Catuogno, L., Dmitrienko, A., Eriksson, K., Kuhlmann, D., Ramunno, G., Sadeghi, A.-R., Schulz, S., Schunter, M., Winandy, M., Zhan, J.: Trusted Virtual Domains – Design, Implementation and Lessons Learned. In: Chen, L., Yung, M. (eds.) INTRUST 2009. LNCS, vol. 6163, pp. 156–179. Springer, Heidelberg (2010)
3. Carapinha, J., Feil, P., Weissmann, P., Thorsteinsson, S.E., Etemoğlu, Ç., Ingthórsson, Ó., Çiftçi, S., Melo, M.: Network Virtualization - Opportunities and Challenges for Operators. In: Berre, A.J., Gómez-Pérez, A., Tutschku, K., Fensel, D. (eds.) FIS 2010. LNCS, vol. 6369, pp. 138–147. Springer, Heidelberg (2010)
4. Lampson, B.W.: A note on the confinement problem. Communications of the ACM 16(10) (1973)
5. National Computer Security Center: A Guide to Understanding Covert Channel Analysis of Trusted System (1993)
6. Venkatraman, B.R., Newman-Wolfe, R.E.: Capacity estimation and auditability of network covert channels. In: Research in Security and Privacy (S&P), Oakland, CA. IEEE (1995)
7. Liberatore, M., Levine, B.N.: Inferring the source of encrypted HTTP connections. In: Computer and Communications Security (CCS). ACM (2006)
8. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Computer and Communications Security (CCS). ACM (2009)
9. Graham, B., Zhu, Y., Fu, X., Bettati, R.: Using covert channels to evaluate the effectiveness of flow confidentiality measures. In: Parallel and Distributed Systems (ICPADS). IEEE (2005)
10. Liu, Y., Ghosal, D., Armknecht, F., Sadeghi, A.-R., Schulz, S., Katzenbeisser, S.: Hide and Seek in Time — Robust Covert Timing Channels. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 120–135. Springer, Heidelberg (2009)
11. Murdoch, S.J., Lewis, S.: Embedding Covert Channels into TCP/IP. In: Barni, M., Herrera-Joancomartí, J., Katzenbeisser, S., Pérez-González, F. (eds.) IH 2005. LNCS, vol. 3727, pp. 247–261. Springer, Heidelberg (2005)
12. Venkatraman, B.R., Newman-Wolfe, R.E.: Performance analysis of a method for high level prevention of traffic analysis using measurements from a campus network. In: Computer Security Applications Conference (ACSAC). IEEE (1994)
13. Millen, J.: 20 years of covert channel modeling and analysis. In: Research in Security and Privacy (S&P), Oakland, CA. IEEE (1999)
14. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301 (2005)
15. Ahsan, K.: Covert channel analysis and data hiding in TCP/IP. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto (2002)
16. Kundur, D., Ahsan, K.: Practical internet steganography: Data hiding in IP. In: Texas Workshop on Security of Information Systems (2003)
17. Degabriele, J.P., Paterson, K.G.: On the (in)security of IPsec in MAC-then-encrypt configurations. In: Computer and Communications Security (CCS). ACM (2010)
18. Sadeghi, A.R., Schulz, S., Varadharajan, V.: The silence of the LANs: Efficient leakage resilience for IPsec VPNs (full version). Technical report (2012)
19. Girling, C.G.: Covert channels in LAN's. IEEE Transactions on Software Engineering 13(2) (1987)
20. Browne, R.: Mode security: An infrastructure for covert channel suppression. In: Research in Security and Privacy (S&P), Oakland, CA. IEEE (1994)

21. Kiraly, C., Teofili, S., Lo Cigno, R., Nardelli, M., Delzeri, E.: Traffic Flow Confidentiality in IPsec: Protocol and Implementation. In: Fischer-Hübner, S., Duquenoy, P., Zuccato, A., Martucci, L. (eds.) The Future of Identity in the Information Society. IFIP, vol. 262, pp. 311–324. Springer, Boston (2008)
22. Moskowitz, I.S., Miller, A.R.: Simple timing channels. In: Research in Security and Privacy (S&P), Oakland, CA. IEEE (1994)
23. Fu, X.: On Traffic Analysis Attacks and Countermeasures. PhD thesis, Texas A&M University (2005)
24. Venkatraman, B.R., Newman-Wolfe, R.E.: Transmission schedules to prevent traffic analysis. In: Computer Security Applications Conference (ACSAC). IEEE (1994)
25. Fu, X., Graham, B., Bettati, R., Zhao, W.: On effectiveness of link padding for statistical traffic analysis attacks. In: International Conference on Distributed Computing Systems (ICDCS). IEEE, Washington, DC (2003)
26. Gettys, J.: Bufferbloat: Dark buffers in the Internet. IEEE Internet Computing 15(3) (2011)
27. El-Atawy, A., Al-Shaer, E.: Building covert channels over the packet reordering phenomenon. In: International Conference on Computer Communications (INFOCOM). IEEE (2009)
28. Mogul, J., Deering, S.: Path MTU discovery. RFC 1191 (1990)
29. Zhao, W., Olshefski, D., Schulzrinne, H.: Internet quality of service: An overview. Technical report, Columbia University (2000)
30. Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., Zhang, L.: Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309 (1998)
31. Bell, D.E.: Looking back on the Bell-LaPadula model. In: Computer Security Applications Conference (ACSAC). IEEE (2005)
32. Llamas, D., Allison, C., Miller, A.: Covert channels in internet protocols: A survey (2006)
33. Zander, S., Armitage, G., Branch, P.: A survey of covert channels and countermeasures in computer network protocols. Comm. Surveys & Tutorials 9(3) (2007)
34. Guan, Y., Fu, X., Xuan, D., Shenoy, P.U., Bettati, R., Zhao, W.: NetCamo: Camouflaging network traffic for QoS-guaranteed mission critical applications. Trans. on Systems, Man, and Cybernetics - Systems and Humans 31(4) (2001)
35. Shmatikov, V., Wang, M.H.: Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 18–33. Springer, Heidelberg (2006)
36. Abraham, T., Wright, M.: Selective cross correlation in passive timing analysis attacks against Low-Latency mixes. In: Global Communications Conference (GLOBECOM). IEEE (2010)
37. Luo, X., Zhou, P., Chan, E.W.W., Lee, W., Chang, R.K.C., Perdisci, R.: HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows. In: Network and Distributed Systems Security (NDSS). Internet Society (2011)
38. Wright, C.V., Coull, S.E., Monrose, F.: Traffic morphing: An efficient defense against statistical traffic analysis. In: Network and Distributed Systems Security (NDSS). Internet Society (2009)
39. Berk, V., Giani, A., Cybenko, G.: Detection of covert channel encoding in network packet delays. Technical Report TR536, Dartmouth College (2005)
40. Gilbert, P.A., Bhattacharya, P.: An approach towards anomaly based detection and profiling covert TCP/IP channels. In: Information, Communications and Signal Processing (ICICS). IEEE (2009)