Results of VPN Fingerprinting Study

Anna Shubina, Mike Toothaker

September 26, 2013

1 Summary of Main Results

1. Use of internally buffered protocols and tools (such as `ssh` or `scp`) for data transfers in an encrypted VPN causes noticeable differences in reactions of the protocol stack implementation to induced, periodic packet loss. In particular, the cipher used by the VPN can be fingerprinted by means of measuring and clustering average packet rates. By contrast, in the experiments where we used simple non-encrypting tools such as `netcat` to transfer files over a UDP-based VPN, we observed no easily fingerprintable pattern in the packet rates.

   In a series of experiments, we used `ssh`, `scp`, and `netcat` to transfer files through *OpenVPN* and *Linux IPsec* implementations with a number of ciphers[1] on both low-latency Gigabit networks and typical Internet paths, while dropping each n-th packet of a connection; in our experiments, $n = 5$ produced results most representative of the described effects. In particular, in case of `ssh`/`scp` transfers, the 100-packet transfer time distributions clustered into distinct configurations of 2–4 distinct clusters of different weights for different ciphers, forming clearly visible fingerprints. In case of `netcat` transfers, the transfer times for the 6 ciphers formed visually indistinguishable – and, most likely, statistically indistinguishable – distributions (with differences easily destroyed by normal network noise). We discuss these fingerprintable differences in 1.1.

   Conducting data transfers inside an encrypted VPN through a buffered encrypted protocol is a fairly standard and generally recommended practice; moreover, non-encrypting file transfer and communication tools such as `ftp` and `telnet` and strongly discouraged or disabled on most systems. However, our findings suggest that "double encryption" that arises from the use of encrypting tools makes the enclosing VPN and VPN's cipher fingerprintable. At the same time, the use of non-encrypting tools such as `netcat` removes the opportunity for such fingerprinting. Given our findings, it may be worth studying the security tradeoffs of the prevailing practice of "double encryption" when using VPNs.

[1] For OpenVPN, AES256-CBC, Blowfish-CBC, DES-CBC, RC2-CBC; for IPsec, AES-CTR, Blowfish-CBC, CAST-128-CBC, DES-CBC, Rijndael-CBC, Twofish-CBC

Furthermore, this cipher information disclosure suggests that the design of buffering and networking layers in cryptographic VPN software must be re-examined, to eliminate the fingerprintable interactions between these layers in the stacks that produce the observed fingerprints.

For example, we note that our experimental comparisons between OpenVPN using UDP for its network transport vs OpenVPN using TCP suggest that the use of UDP may be more robust and less fingerprintable when faced with periodic packet drops, as discussed in 1.2.

We discuss a variety of related observations in Section 5, but note that, to the best of our knowledge, none of these focused specifically on VPNs.

1.1 Cluster fingerprints of different ciphers

Figure 1 shows 100-packet timings of an `scp` file transfer through IPsec, with Rijndael-CBC and Blowfish-CBC, where each 5th packet is dropped. This experiment was done on a fast local network with minimal noise. As a result, almost all 100-packet timings fell into one of 4 clusters (same for each ciphers) - but the proportion of packets in each cluster is different for the two ciphers.

We found that, for big file transfers done with the same cipher on the same fast network, the relative sizes of clusters in different captures were very similar, whereas these relative sizes differed greatly between captures done with different ciphers. Thus, the relative sizes of clusters of 100-packet timings could be used as a fingerprint. As shown in figure 4, when, while using IPsec and dropping every 5th packet, we switched from using `scp` to using `netcat`, interesting timing artifacts entirely disappeared from our captures. Each 100 packets appeared to take the same time to go through, no matter what cipher.

1.2 OpenVPN with UDP vs OpenVPN with TCP

Seeing that using `netcat` eliminated some of the interesting fingerprintable timings from our captures, we wondered if there are other simple ways to make the timings less informative.

In figures 7 and 9 are 100-packet timings from our slow-connection experiments with OpenVPN over UDP and over TCP, with Blowfish-CBC and each 5th packet dropped. The UDP timings break up in 2 clusters, similar to our fast network OpenVPN `netcat` experiments, whereas the TCP timings display a significantly more interesting picture – and entirely different from figure 11, where the same experiment is conducted with AES-256 for the cipher.

2. Fingerprintable effects are most pronounced on fast (Gigabit) intranet links, but also present on slower Internet paths.

Figure 1: IPsec Rijndael-CBC vs Blowfish-CBC with `scp`, every 5th packet dropped

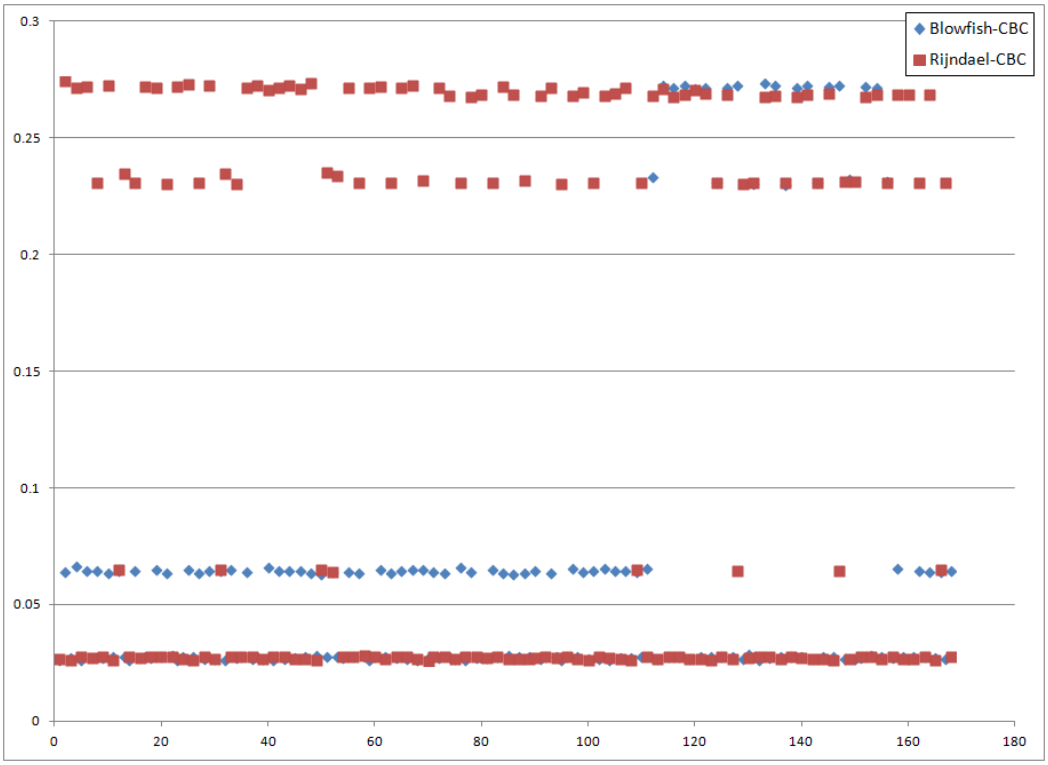


Figure 2: IPsec Rijndael with scp, every 5th packet dropped, spectral view

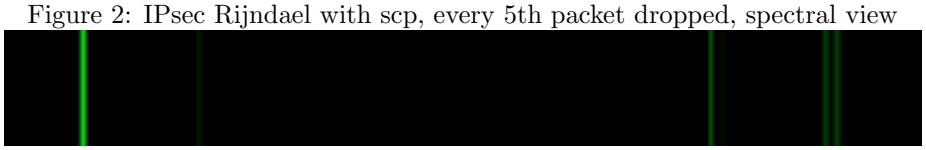


Figure 3: IPsec Blowfish with scp, every 5th packet dropped, spectral view

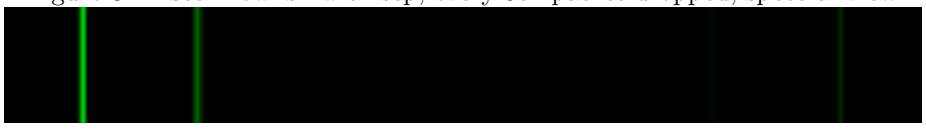


Figure 4: IPsec Rijndael-CBC vs Blowfish-CBC with `netcat`, every 5th packet dropped

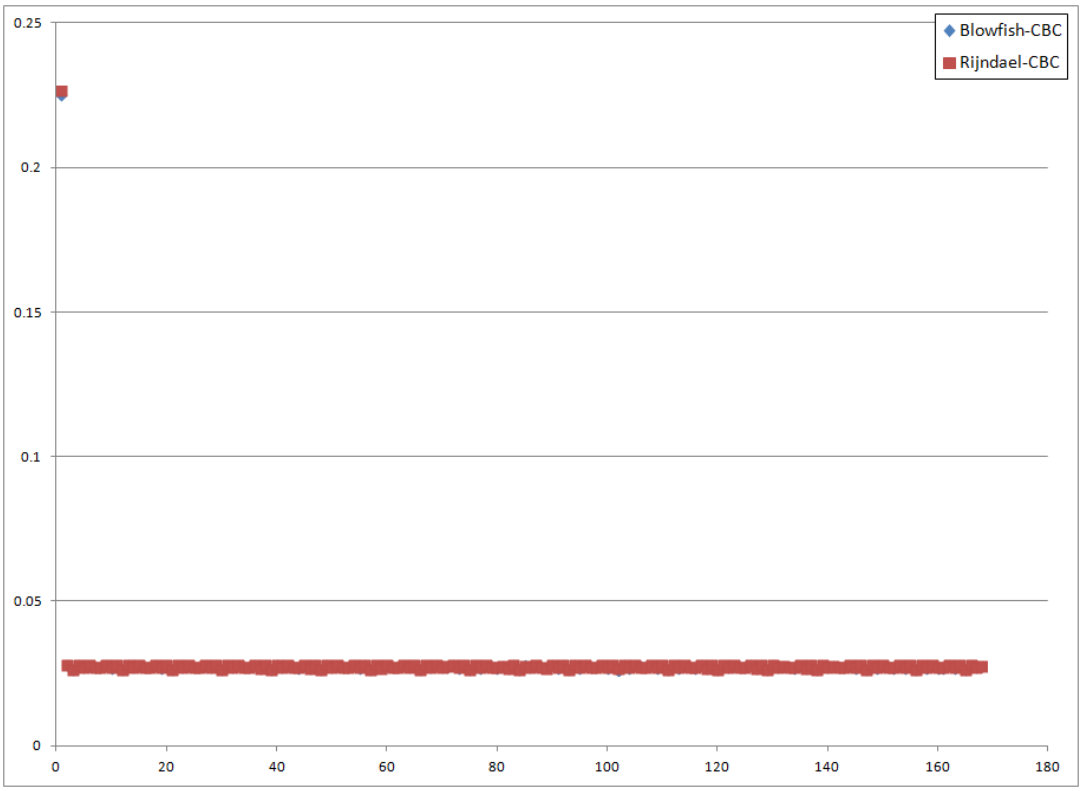


Figure 5: IPsec Rijndael with netcat, every 5th packet dropped, spectral view

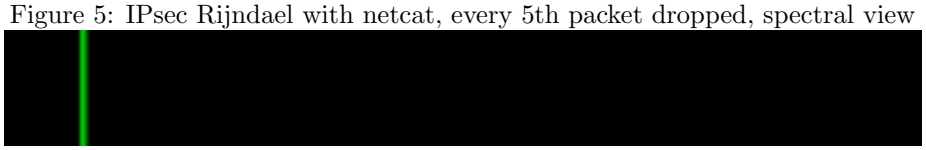


Figure 6: IPsec Blowfish with netcat, every 5th packet dropped, spectral view

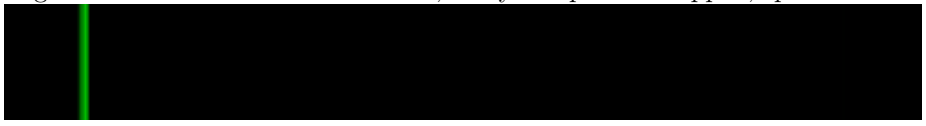


Figure 7: OpenVPN over UDP with Blowfish over slow connection, every 5th packet dropped

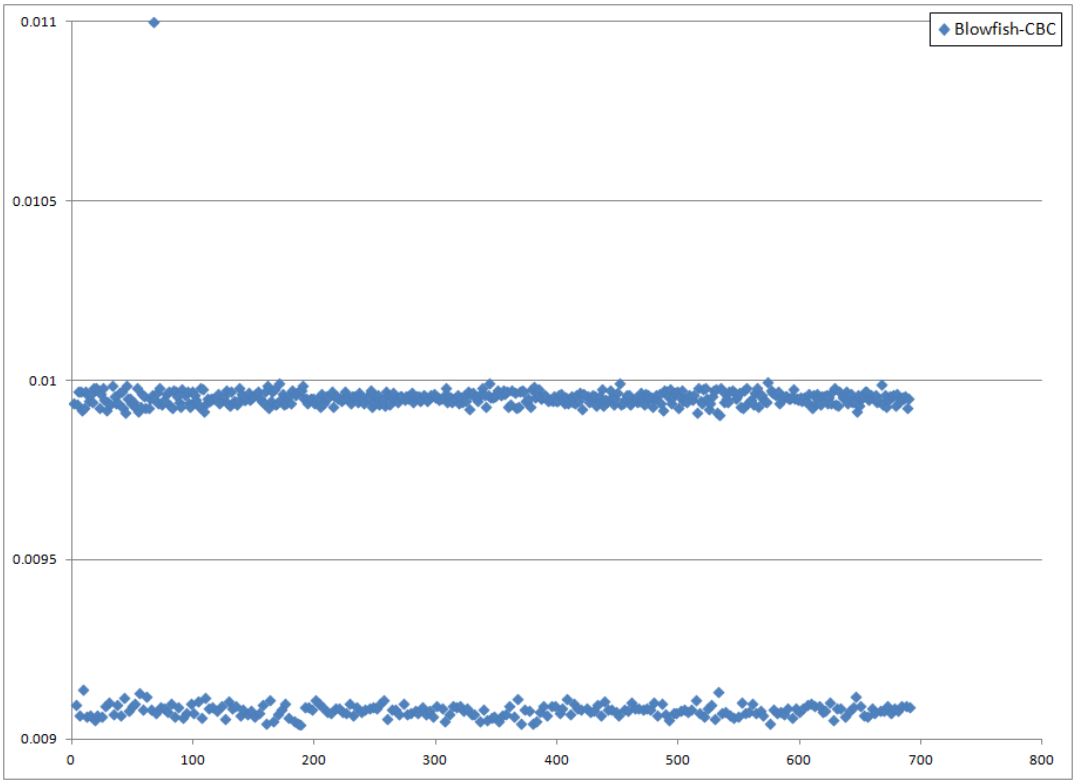


Figure 8: OpenVPN over UDP with Blowfish over slow connection, every 5th packet dropped, spectral view

Figure 9: OpenVPN over TCP with Blowfish over slow connection, every 5th packet dropped

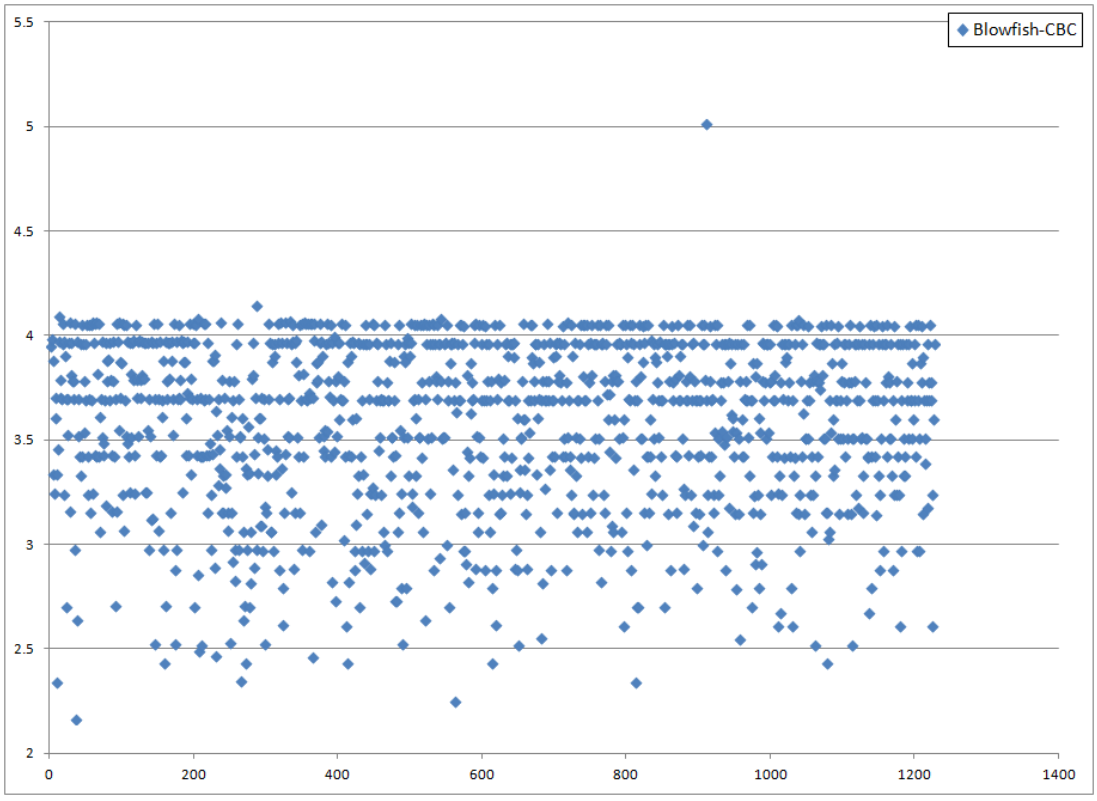


Figure 10: OpenVPN over TCP with Blowfish over slow connection, every 5th packet dropped, spectral view

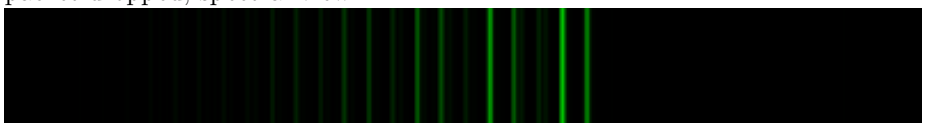
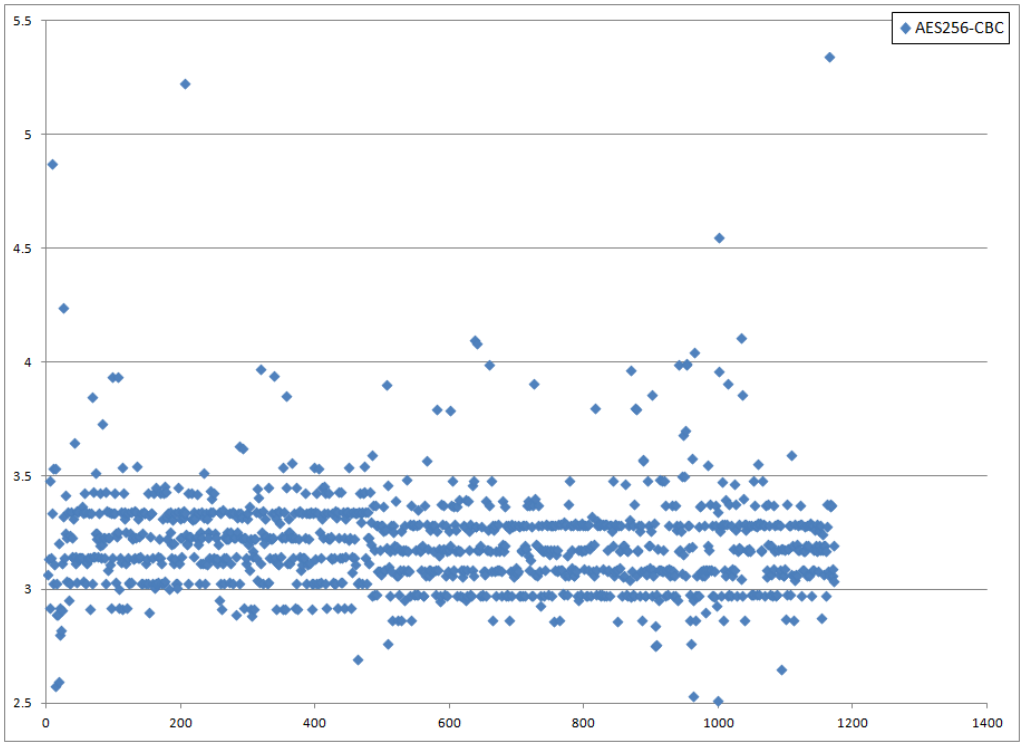


Figure 11: OpenVPN over TCP with AES-256 over slow connection, every 5th packet dropped



We conducted data transfers both on a fast local network and from a computer in New Hampshire to a computer in California. Although timings of fast local transfers did produce significantly more uniform results, with clear-cut clusters visible with the unarmed eye, timings of remote transfers did produce patterns that could be compared against each other by using clustering algorithms.

3. The fingerprints we generated can be masked by several different means:

   - use of simple, non-buffered, non-encrypted file transfer protocols (e.g., `netcat`), ("no double encryption");
   - introduction of random packet delays (e.g., by way of Linux queuing

Figure 12: OpenVPN over TCP with AES-256 over slow connection, every 5th packet dropped, spectral view

disciplines).

- renegotiating ciphers thoughout file transfer.

We suspect that Linux Berkeley packet filter queues can significantly distort timing, but we have not explored this effect. The patterns of timestamps we observed when we did packet filtering at the time of capture rather than after the time of capture appeared to bear no similarity whatsoever to the patterns where packet filtering was done after capturing.

2 Key Encryption Type Signals for Packet Loss Fingerprinting

Periodic packet loss produces distinct clustering patterns in average packet rates. In particular, on Gigabit links with stable latency, the typical cluster configuration has 2-4 clusters of varying relative sizes for each cipher, whereas a transfer without the induced packet loss produces just one "cluster" at the same scale. We hypothesize that the patterns are due to interaction of various buffers and queues on the path; introduction of another queue or buffering scheme splits the clusters.

We calculated the times $t_1, \ldots, t_n$ it takes for 100 data packets to go through our connection. We then ran k-means clustering on $t_1, \ldots, t_n$, letting the algorithm determine the best number of clusters. We then sorted the clusters in the order of their increasing centers and computed the weights of the clusters $w_1, \ldots, w_k$.

The set of pairs $\{(c_1, w_1), \ldots, (c_k, w_k)\}$, where c_i, w_i are respectively the center and the weight of the ith cluster, and $c_1 < c_2 < \ldots < c_k$ is our fingerprint.

3 Designing Improved Network Encryption Stacks

Our results show that double encryption, and, generally, double buffering, in encrypting VPNs should be avoided. It has long been understood that the use of double buffering in VPNs has undesirable effects because of interactions of buffer time-out schemes such as "TCP over TCP" (e.g., tunneling of TCP connections over `SSH` severely degrades performance in presence of packet loss or delays). We show that similar performance degradation effects occur when double encryption is used.

Our results suggest that the reliability layer and the cryptographic layer of a VPN implementation interact considerably in the presence of packet loss (and that such interactions are fingerprintable, with the effect of information disclosure).

Such interaction is easiest to reduce in case when the VPN transport and encryption are done at the same layer, not in two different layers.

4 Steps for Improving VPN Connections using VPN Underground Techniques

Induced periodic packet loss and delay significantly interfere with both the initial handshake of VPNs (and can in fact make establishing a connection virtually impossible, which some web browsing traffic still succeeds), and with transmission of data. Our results suggest that relying on a stream transport to implement the encryption on top of an independent stream layer may be at fault.

We recommend that VPN implementors use our fingerprinting methods to measure effects on authentication and transmission, and adjust their implementations accordingly. It appears that cryptographic schemes that align their messages to packets (e.g., OpenVPN over UDP) survive periodic packet loss patterns better than packet-agnostic stream-based ones.

5 Related Work

Effects of TCP layer implementation timeouts, especially those of the Nagle back-off algorithm, on performance of TCP-backed transports, have long been known; this is why tunneling via SSH is known as "poor man's VPN".[2] However, analysis of the adverse effects of such tunnels focuses on poor throughput and latency in presence of random packet loss, not fingerprinting or information disclosure.

Another research program investigated the effects of specially inducing packet loss on TCP, known under the name of "reduction of quality" (RoQ) attacks. In these attacks, the quality of TCP connections on a particular network path was severely reduced by artificially creating short bursts of congestion, calculated to cause packet loss on the path's routers. Unlike the TCP over TCP analysis, the induced packet loss was deliberate and non-random. However, these attacks, although an important milestone in understanding attack susceptibility of TCP, did not consider VPNs or any encryption in the targeted sessions. A research summary of RoQ can be found at `http://csr.bu.edu/roq/`.

Finally, the effects of fragmentation on IPsec sessions recently attracted attention of cryptography researchers as a potential source of oracles. [3] Although we do not perform any cryptographic analysis of the consequences of our fingerprinting, we note that our findings may be due in part to the same fragmentation effects. More research is required here.

[2] E.g., a popular explanation can be found at `http://sites.inka.de/~W1011/devel/tcp-tcp.html`, see also `http://www.ispl.jp/~oosaki/papers/Honda05_ITCom.pdf`, `http://rockhoppervpn.sourceforge.net/techdoc_tcpotcp.html`, etc.

[3] "Security of Symmetric Encryption in the Presence of Ciphertext Fragmentation", `http://www.cs.bris.ac.uk/eurocrypt2012/Program/Thurs/Degabriele.pdf`