

# Automatic VPN Client Recovery from IPsec Pass-through Failures

José Carlos Brustoloni

Dept. Computer Science, University of Pittsburgh  
210 S. Bouquet St. #6111, Pittsburgh, PA 15260, USA  
Email: jcb@cs.pitt.edu

## Abstract

*Network Address Translation (NAT) is often used in routers that connect home and small-office networks to the Internet. Unfortunately, NAT may not interoperate well with many protocols, including IPsec, the security protocol suite often used by telecommuters. Many NAT implementations include heuristics commonly known as IPsec Pass-through, which may enable NAT to interoperate with IPsec under certain assumptions. We characterize IPsec Pass-through's operation and failure modes, and propose IPsec Pass-Through Automatic Client Recovery (IPTACR), a novel set of mechanisms that enable VPN clients to recover automatically from IPsec Pass-through failures. Experiments show that the proposed mechanisms are effective and impose negligible overhead.*

## 1. Introduction

Home and small-office networks typically use private IP addresses [1]. Consumers often prefer the latter because, unlike globally unique IP addresses, private IP addresses are free.

Because private IP addresses may not be globally unique, they cannot be routed on the Internet. To enable hosts in private networks to communicate with the Internet, many routers offer Network Address Translation (NAT) [2]. When a request packet is sent from a *client* (host in a private network) to a *server* (host reached via the Internet), NAT replaces the source private IP address with a global IP address, and, in the case of TCP and UDP packets, the source port number with a global port number. Usually, NAT shares a globally unique IP address among all clients, while assigning different global port numbers to each client. When NAT receives from the Internet a reply packet, NAT translates the destination global IP address and port number to the corresponding client's private IP address and port number. The router then forwards the packet to the client.

Unfortunately, NAT may not interoperate readily with protocols whose payload depends on the packet's source or destination address or port number. In many cases, a so-called application-level gateway (ALG) can modify payloads so that the latter remain consistent with the respective headers after NAT modifications. In other cases, however, such compensation is not possible. IPsec [3], the security protocol suite often used by telecommuters to access their employers' virtual private networks (VPNs) [4], is among the latter cases. Because a router's ALG does not have access to the keys that end nodes use to authenticate or encrypt IPsec payloads, any payload modifications by the ALG would be detected by the packet receiver and result in packet rejection.

However, heuristics commonly known as IPsec Pass-through enable NAT to interoperate with a useful subset of IPsec under certain assumptions. IPsec Pass-through is implemented in most home and small-office routers. However, IPsec Pass-through is not a standard, and has not been properly described in previous technical literature. There is, therefore, considerable confusion about what is meant by IPsec Pass-through, and implementations are often unnecessarily limited. This paper contributes a description of IPsec Pass-through's operation modes in Section 2 and a discussion of the corresponding limitations in Section 3.

In particular, Section 3 characterizes the cases in which IPsec Pass-through may cause clients' communication to hang. When such cases occur, currently the only recourse is to reboot the affected clients (and perhaps the router) and retry the communication. In Section 4, we propose IPsec Pass-Through Automatic Client Recovery (IPTACR), a novel set of mechanisms that automatically detect and recover from IPsec Pass-through's failures, without reboot. Experiments in Section 5 show that IPTACR effectively overcomes IPsec Pass-through's failure modes while imposing negligible overhead. We discuss related work in Section 6, and conclude in Section 7.

## 2. How IPsec Pass-through works

This section describes IPsec Pass-through's principles of operation.

IPsec packets do not have the equivalent of a TCP or UDP port number that NAT could replace in order to identify the client. (IPsec's key exchange protocol, IKE [5], is layered on top of UDP, but many IKE implementations require both source and destination port numbers to be equal to 500. UDP port translation would not interoperate with these IKE implementations.) Therefore, IPsec Pass-through can translate only IP addresses, and needs to resort to heuristics to identify the client. There are three sets of heuristics in commercial use. We call them IPsec Pass-through versions 1, 2, or 3. We discuss each version in one of the following subsections.

### 2.1. IPsec Pass-through version 1 (IPTv1)

The simplest heuristic is to allow only one IPsec client at a time. If IPTv1 receives an IPsec packet from a client and there is no active IPsec session, IPTv1 notes the client's private address. IPTv1 then forwards to the Internet IPsec packets only from that client, and forwards to that client any IPsec packets arriving from the Internet. IPTv1 needs to monitor IPsec activity and terminate the client's session if it remains inactive for too long (e.g., more than 10 minutes).

### 2.2. IPsec Pass-through version 2 (IPTv2)

A slightly more complex but more general heuristic is to allow only one IPsec client at a time for any given server. If IPTv2 receives an IPsec packet from a client and there is no active IPsec session with the packet's destination, IPTv2 notes on a table the client's private address and the server's address (equal, respectively, to the packet's source and destination addresses). IPTv2 then forwards to the server IPsec packets only from that client, and forwards to that client any IPsec packets arriving from the server. IPTv2 needs to monitor the activity of all IPsec sessions and terminate an IPsec session if it remains inactive for too long.

### 2.3. IPsec Pass-through version 3 (IPTv3)

A still more general and complex set of heuristics allows any number of IPsec sessions between clients and any given server.

IPTv3 handles IKE as follows. The IKE header contains plaintext initiator and responder cookies. Each cookie has 64 bits and is selected in the beginning of an IKE session to uniquely identify the session within the respective IPsec

peer [5, 6]. IPTv3 maintains a hash table of items containing client and server IP addresses and initiator and responder cookies. Each item corresponds to an IKE session. An IKE item is considered *outstanding* or *established* according to whether the responder cookie is null or not, respectively. When a client sends an IKE packet such that no item in the table matches the client and server IP addresses and initiator cookie, IPTv3 creates an outstanding item containing those values. Conversely, when a server sends an IKE packet such that no established item in the table matches the server IP address and initiator and responder cookies, but an outstanding item matches the server IP address and initiator cookie, IPTv3 converts this item into an established one, including the packet's responder cookie. IPTv3 then forwards the packet to the corresponding client.

IPTv3 can allow more than one client to use the same initiator cookie with the same server. Such a conflict can occur because each client selects initiator cookies independently. It can be expected, however, to be a rare event: IPsec standards recommend that each endpoint pick a cookie that is a secure hash (e.g., MD5) of both endpoints' addresses and port numbers, a local secret, and the date and time [6]. If  $n$  clients simultaneously start an IKE session with the same server, with  $n$  reasonably limited, the probability of an initiator cookie collision is therefore on the order of  $2^{-64}$ . IPTv3 can handle such collisions correctly by using the responder cookie to identify the client. To do this, IPTv3 needs to slightly modify the above rules and serialize the initial IKE handshake as follows. When a client  $c_1$  sends an IKE packet whose client and server IP addresses and initiator cookie match no item in the table, but whose server IP address and initiator cookie match another client  $c_2$ 's outstanding item, IPTv3 drops that packet. Eventually, when  $c_1$ 's IKE implementation retries the transmission, after timeouts,  $c_2$ 's item will likely already be established, and IPTv3 will forward  $c_1$ 's packet normally.

IPTv3 handles IPsec's Encapsulating Security Payload (ESP) [7] protocol as follows. ESP may add to an IP payload a message authentication code (MAC) and/or encrypt the IP payload. The Security Parameters Index (SPI) in the ESP header enables the receiver to determine what algorithms and keys to use for packet authentication and/or decryption. SPIs have 32 bits and are transmitted in plaintext. The receiver randomly selects an SPI during the IKE negotiation that establishes an ESP security association (SA). During this negotiation, SPIs are transmitted in encrypted IKE payloads and are not revealed to NAT routers. IPTv3 maintains a hash table of ESP items containing client and server IP addresses and outgoing and incoming SPIs. An ESP item is considered *outstanding* or *established* according to whether the incoming SPI is null or not, respectively. When a client sends an ESP packet such that no item in the table matches the client and server IP addresses and outgo-

ing SPI:

- If the server IP address matches no outstanding item, IPTv3 creates an outstanding item containing the client and server IP addresses and outgoing SPI;
- Otherwise, IPTv3 drops the packet.

Conversely, when a server sends an ESP packet such that no established item matches the server IP address and incoming SPI:

- If an outstanding item matches the server IP address, IPTv3 converts that item into an established one, including the packet's incoming SPI, and forwards the packet to the corresponding client;
- Otherwise, IPTv3 multicasts the packet to all clients that have recently had an IKE negotiation with the server.

To thwart denial-of-service attacks, IPTv3:

1. Times out outstanding ESP items after a short period  $T_o$ ;
2. After an outstanding ESP item of client  $c_1$  causes a packet of another client  $c_2$  to be dropped, limits to a maximum value  $N_o$  the number of packets that  $c_1$  may send matching the outstanding item; and
3. Times out established ESP items after a period of inactivity  $T_{ei}$ .

Each ESP item corresponds to an ESP tunnel's *epoch*, between successive rekeyings. We say that an epoch of an ESP tunnel comprises two SAs, one in each direction. SAs always have a limited lifetime, after which an IKE negotiation happens and establishes a new epoch's SAs with new keys and SPIs. For example, in Linux's IPsec implementation, FreeS/WAN [8], an ESP tunnel's lifetime is characterized by three parameters: *keylife* (default 8 hours), *rekeymargin* (default 9 minutes), and *rekeyfuzz* (default 100%). Suppose IKE creates or rekeys an ESP tunnel, i.e., generates two new SAs, one in each direction, at time  $t_0$ . Both peers use these SAs to send packets since  $t_0$  until some time  $t_1$ , when the tunnel is rekeyed, and accept received packets that use these SAs since time  $t_0$  until  $(t_0 + \textit{keylife})$ . At a random time between:

$$[t_0 + \textit{keylife} - (1 + \textit{rekeyfuzz}) * \textit{rekeymargin}]$$

and:

$$(t_0 + \textit{keylife} - \textit{rekeymargin}),$$

one or both peers initiate an IKE negotiation to rekey the tunnel, if none has started yet. This negotiation is expected to complete successfully by time  $t_1$ ,  $t_1 < (t_0 + \textit{keylife})$ .

### 3. IPsec Pass-through limitations

This section discusses IPsec Pass-through's limitations and failure modes.

#### 3.1. Support for IKE and ESP tunnel mode only

NAT with IPsec Pass-through is compatible with IKE and ESP tunnel mode. In IKE, if clients identify themselves using fully qualified domain names instead of IP addresses, IKE payloads and their authentication and encryption do not depend on IP headers, which NAT modifies. ESP tunnel mode encapsulates an authenticated and/or encrypted IP packet in another IP packet [7]. ESP-authenticated and/or -encrypted packet portions do not depend on the external IP header, which uses a private address for the client and is modified by NAT. The internal IP header typically uses another, VPN-assigned address for the client. Because the latter IP header is not affected by NAT, there is no need for an ALG to guarantee interoperability with protocols that may be layered on top of ESP and whose payload depends on IP addresses or port numbers, e.g. ftp and H.323.

However, NAT's address modifications invalidate packets of certain other IPsec protocols, namely Authentication Header (AH) [9] and ESP transport mode [7]. AH adds to IP packets a MAC that depends on parts of the IP header, including source and destination addresses, plus the entire IP payload. If either of those addresses is modified, the packet's MAC becomes invalid. ESP's MAC and encryption depend on the IP payload only. However, if the IP payload is a TCP segment or UDP datagram, it contains a checksum that depends on parts of the IP header, including source and destination addresses, plus the TCP or UDP payload. Unlike IPsec's tunnel mode, transport mode has only one IP header. Thus, NAT modifications of this header invalidate TCP or UDP checksums in ESP transport mode packets.

Support for IKE and ESP tunnel mode is sufficient for many VPNs. VPNs usually use ESP both for packet authentication and encryption, and do not use AH. Microsoft uses ESP transport mode to secure L2TP [10], but most VPN vendors use ESP tunnel mode instead, because the latter has less overhead.

#### 3.2. Recovery from NAT crashes but no fail-over

Because IPsec Pass-through translates IP addresses only, and not port numbers, it can readily recover from crashes. If a single global IP address is shared among all clients, as is usually the case, normal client traffic after a NAT router crash can be expected to cause IPsec Pass-through to relearn the same translations that were in place before the crash, recovering end-to-end communication.

However, IPsec Pass-through does not readily support fail-over recovery. Suppose a private network is multi-homed and is connected to the Internet via instances A and B of IPsec Pass-through with distinct global IP addresses. Suppose further that a client establishes an end-to-end ESP tunnel to a server through A. If A crashes, the client's ESP packets may be rerouted and reach the server through B. However, the server continues to have the tunnel configured with A as the other endpoint. Therefore, the server may reject the client's packets when they arrive with B as the source address. The server also continues to send its ESP packets to A, even though A is down and cannot forward the packets to the client.

### 3.3. IPTv1 and IPTv2 limitations

IPTv1's limitation of only one IPsec client at a time is problematic, e.g., in households where both spouses work. Given that IPTv2 is only slightly more complex to implement and is more general, there seems to be little reason to accept IPTv1's limitations. IPTv2 allows more than one IPsec client at a time, provided that they access different servers.

IPTv2 can still be problematic, e.g., if both spouses in a household or multiple guests at a hotel work for the same company and need to access the same VPN. This limitation can be overcome by IPTv3 at a cost of somewhat more complex implementation.

### 3.4. IPTv3 collisions and race conditions

IPTv3 allows multiple clients per server and therefore does not share IPTv2's limitation. However, when used with multiple clients per server, it may in some cases fail, as explained in the following. These failures do not occur if IPTv3 is employed with the same limitation as IPTv2, i.e., with at most one client per server.

IPTv3's hash tables can be viewed as soft and possibly incorrect state. IPTv3 loses state, e.g., when an item times out or the NAT router crashes. IPTv3 automatically reacquires state when subjected to further traffic. In IKE's case, IPTv3's state may be incomplete but not incorrect. Moreover, IKE has built-in timeouts and retry mechanisms that will promote recovery of IPTv3's state, if necessary. Note that, if  $(keylife - rekeymargin)$  is larger than IPTv3's  $T_{ei}$  parameter, by the time the tunnel needs to be rekeyed, the corresponding IKE entry in IPTv3's translation table may be already expired. In such cases, the server may be unable to initiate the tunnel rekeying. However, the client will also attempt to rekey the tunnel, and when it does so, IPTv3 reestablishes the tunnel's IKE entry and allows tunnel rekeying to proceed.

Similar recoverability does not exist, however, in ESP's case. IPTv3's state may be incomplete, incorrect (i.e., may incorrectly associate client and server IP addresses and outgoing and incoming SPIs, as explained in the following paragraphs), or both. Additionally, ESP does not have built-in mechanisms that would promote timely recovery from losses or errors in IPTv3's state.

The first source of errors in IPTv3's ESP state is incoming SPI collisions. This is possible because each client chooses incoming SPIs independently. It can be expected, however, to be a rare event. If choice of SPIs is random and  $n$  clients have an ESP tunnel to the same server, with  $n$  reasonably limited, then the probability of an incoming SPI collision each time a tunnel is created or rekeyed is on the order of  $2^{-32}$ . When such an event happens, IPTv3 will forward to the client whose item is first established all packets sent by the server using the given incoming SPI, and remaining clients with the same SPI will not receive their packets.

The second source of errors in IPTv3's ESP state is race conditions that may cause misassociation of outgoing and incoming SPIs. IPTv3 implicitly assumes that, after a tunnel is created or rekeyed, or the corresponding item in the table is timed out, or IPTv3 recovers from a crash, first the client uses the tunnel to send a packet to the server, and, shortly after receiving this packet (but not before), the server uses the tunnel to send a packet back to the client. These assumptions may be violated in a number of cases:

1. Suppose client  $c$  has a tunnel  $t$  to server  $s$ . If there is no outstanding item that matches  $s$  and  $s$  uses  $t$  before  $c$  does (e.g.,  $c$  was downloading a file when  $t$  was rekeyed), IPTv3 has to multicast the packet to all clients that have recently had an IKE negotiation with the server, as shown in Fig. 1(a).
2. Suppose clients  $c_1$  and  $c_2$  have tunnels  $t_1$  and  $t_2$  to the same server  $s$ , respectively. If  $c_1$  uses  $t_1$  before  $s$  does, and, before using  $t_1$ ,  $s$  uses  $t_2$  before  $c_2$  does, then IPTv3 incorrectly associates  $t_1$ 's outgoing SPI with  $t_2$ 's incoming SPI, and will deliver to  $c_1$  packets sent to  $c_2$ , as illustrated in Fig 1(b). Thereafter, neither  $c_1$  nor  $c_2$  will receive the respective packets.
3. Suppose clients  $c_1$  and  $c_2$  have tunnels  $t_1$  and  $t_2$  to server  $s$ , respectively. If  $c_1$  uses  $t_1$  before  $s$  does,  $t_1$ 's outstanding item times out, and  $c_2$  uses  $t_2$  before  $s$  does, then if  $s$  replies to  $c_1$  before it replies to  $c_2$ , IPTv3 will incorrectly associate  $t_2$ 's outgoing SPI with  $t_1$ 's incoming SPI, and will deliver to  $c_2$  the packet sent to  $c_1$ , as shown in Fig. 1(c). Thereafter, neither  $c_1$  nor  $c_2$  will receive the respective packets.
4. Suppose clients  $c_1$  and  $c_2$  have tunnels  $t_1$  and  $t_2$  to server  $s$ , respectively, and that  $t_1$ 's established item

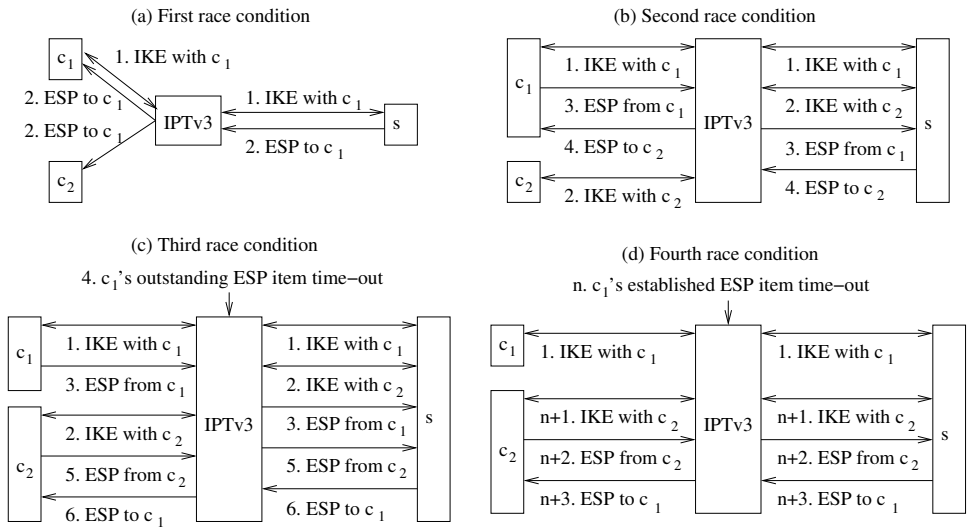


Figure 1: When IPsec Pass-through version 3’s heuristics fail, race conditions may occur if there are multiple clients accessing the same server.

times out. If  $c_2$  uses  $t_2$  before  $s$  does, but, before replying to  $c_2$ ,  $s$  uses  $t_1$ , then IPTv3 will incorrectly associate  $t_2$ ’s outgoing SPI with  $t_1$ ’s incoming SPI, and will deliver to  $c_2$  the packet sent to  $c_1$ , as illustrated in Fig. 1(d). Thereafter, neither  $c_1$  nor  $c_2$  will receive the respective packets.

In summary, IPTv3 failures may cause it to forward ESP packets to clients other than the recipient intended by the sender. Because ESP’s verification of packet authenticity and decryption depend on keys held only by a packet’s intended recipient, other clients will simply drop misdelivered packets. However, the intended recipient may not receive his or her ESP packets. Neither ESP nor IKE nor higher-level protocols (e.g., TCP) will be able to achieve recovery. IKE may clear collisions and misassociations when the lifetimes of the tunnels involved expire, but lifetimes may be long (often several hours). Clients may therefore need to reboot their computers to recover from such conditions.

#### 4. IPsec Pass-through Automatic Client Recovery

This section describes IPsec Pass-Through Automatic Client Recovery (IPTACR), a new set of mechanisms that promote automatic VPN client fail-over or recovery in cases of NAT crashes or IPsec Pass-through collisions or race conditions.

IPTACR does not require modifications in protocols or VPN gateways; it needs only three simple modifications in VPN client software. These modifications are harmless if

a user’s VPN connection does not actually involve NAT or IPsec Pass-through.

IPTACR introduces new parameters *maxidle*, *pingtime*, and *pingtries*. IPTACR’s first modification is that, immediately after a client has sent its final IKE packet for creating or rekeying an ESP tunnel, the client sends a ping (ICMP echo) request to the server using the new epoch’s outgoing SA. Additionally, if a client has not received any packets through an incoming SA for a period greater than *maxidle*, the client sends a ping request through the corresponding outgoing SA. If, having sent the last ping request more than *pingtime* ago, the client has not received any packet through the corresponding incoming SA, the client resends the ping request, up to *pingtries* times. These pings automatically reveal lack of VPN connectivity, regardless of cause, and reduce the likelihood of the first two race conditions discussed in the previous section. They also eliminate the fourth race condition altogether if  $maxidle < T_{ei}$ . If, after *pingtries* attempts, the client has not received any packet through the incoming SA, the client starts a new IKE negotiation to rekey the ESP tunnel. Tunnel rekeying with new SPIs promotes recovery from any of the collisions or race conditions discussed in the previous section.

IPTACR’s second modification is that if a client repeatedly attempts to rekey an ESP tunnel and, during these attempts, does not receive replies from the server, then the client drops the existing IKE session and starts a new IKE session. The client then uses this new IKE session to create a new ESP tunnel. The new IKE session promotes fail-over recovery when NAT crashes. The new IKE session and any ESP tunnels created by it will be routed through a back-up NAT instance, if one exists, restoring connectivity.

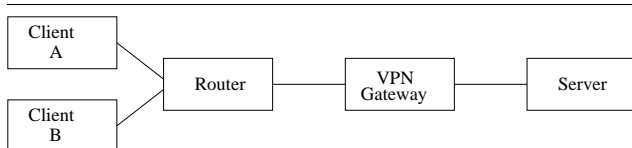


Figure 2: Experimental setup.

IPTACR’s third and final modification is that, after an ESP tunnel is created or rekeyed, the client continues to use the previous epoch’s outgoing SA to send packets (except for ping requests, as described above), until the client receives a packet through the new epoch’s incoming SA, or until the previous epoch’s time  $t_1 < (t_0 + \textit{keylife})$ , whichever occurs first. After that, the peer uses the new epoch’s outgoing SA to send packets. This rule may prevent IPsec Pass-through from dropping data sent from client to server immediately after ESP tunnel creation or rekeying. Without this modification, IPsec Pass-through drops such data if it has another client’s ESP item outstanding for the server. With this modification, however, clients wait until the ping request and reply establish the ESP item of a new epoch before using that epoch’s SAs to send data.

## 5. Evaluation

We evaluated IPTACR on the testbed shown in Fig. 5. Each of the testbed’s nodes is a PC running Linux, and the links are crossover Ethernet cables. All links operated at 10 Mbps. We configured Linux’s IPTv3 implementation, VPN Masquerade, in the Router, and IPsec in the Clients and VPN Gateway. We used ESP tunnel mode with MD5 and 3DES and a rekeying period of 1 minute.

We instrumented the software in the Clients and VPN Gateway so as to be able to recreate SPI collisions and each of the race conditions discussed in Section 3 on demand. Using this instrumented software, we verified that IPTACR automatically and correctly recovers from SPI collisions and each of IPTv3’s race conditions. We also intentionally crashed the Router when an IPsec session was active. We verified that IPTACR promotes automatic and correct recovery also from NAT crashes.

To verify IPTACR’s effect on performance, we had Client A download via FTP a file of 19 MB from the Server (on a warm buffer cache). The effective throughput was 2640 Kbps without IPTACR and 2620 Kbps with IPTACR.

These results suggest that IPTACR can significantly improve the robustness of IPsec Pass-through and has negligible impact on performance. IPTACR’s overhead is small because it consists simply of periodic pings after rekeying. These pings consume little bandwidth.

## 6. Related work

NAT Traversal is an alternative technique for allowing IPsec to interoperate with NAT. It modifies IKE so that the peers discover if there is NAT between them and, if so, use different UDP ports to communicate (i.e., do not use the standard UDP port 500). Additionally, NAT Traversal causes ESP packets to be encapsulated in UDP packets using the same port numbers as the corresponding IKE session. NAT Traversal enables NAT routers to support IKE and ESP both in transport and tunnel mode without special handling: NAT can translate the port numbers of UDP packets encapsulating IPsec packets just like those of any other UDP packets. However, UDP encapsulation imposes more overhead than does IPsec Pass-through and, like the latter, does not support AH. NAT Traversal has been implemented in a variety of mutually incompatible ways by different vendors, although there are standards-track specifications [11, 12].

Users typically have no control over whether a VPN gateway they need to access (e.g., as telecommuters) enables NAT Traversal and, if so, what variant of NAT Traversal it supports. Therefore, home and small-office NAT routers typically offer IPsec Pass-through alongside NAT Traversal. However, IPsec Pass-through may interfere with NAT Traversal. If a router implements IPTv1 or IPTv2, then the router will not enable a second IKE session before the item established by a previous IKE session times out (NAT Traversal causes IKE to switch to different ports only after establishment of an IKE item and NAT discovery). The corresponding delay may frustrate users. On the other hand, if a router implements IPTv3, it interoperates well with NAT Traversal, especially if the VPN client includes IPTACR.

Other mechanisms have been proposed to enable NAT to interoperate with IPsec. In particular, EASE [13] and RSIP [14] are experimental schemes that enable VPN clients to communicate with NAT and explicitly lease incoming SPI values. NAT therefore does not need to resort to heuristics to determine the client associated with a new incoming SPI, and can avoid IPsec Pass-through’s race conditions. However, EASE and RSIP require considerable modifications in the client’s operating system and in the NAT implementation, which users often do not control (e.g., in hotels and Wi-Fi hotspots). Therefore, they can be difficult to deploy.

IPv6 [15] makes globally unique IP addresses plentiful, and could reduce the need for NAT. However, IPv6 deployment has been slow. Additionally, if ISPs continue to charge clients per global IP address, NAT may continue to be used even after a transition to IPv6.

## 7. Conclusions

NAT is used in many homes, hotels, small offices, and other places that telecommuters, travelers, and other potential VPN users frequent. To accommodate such users, most NAT implementations include IPsec Pass-through. IPsec Pass-through consists of heuristics that enable NAT to interoperate with a useful subset of IPsec. Although commercially important, IPsec Pass-through has attracted little previous attention in the literature. We characterized IPsec Pass-through's operation and failure modes, and proposed IPTACR, a novel set of mechanisms that enable VPN clients to recover automatically from IPsec Pass-through's failures. IPTACR can also improve IPsec Pass-through's interaction with other alternatives, such as NAT Traversal. Experiments suggest that IPTACR is highly effective and has negligible performance overhead.

## Acknowledgments

This work was supported in part by NSF ITR medium ANI-0325353.

## References

- [1] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear. "Address Allocation for Private Internets," IETF, RFC 1918, 1996. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt>
- [2] P. Srisuresh and M. Holdrege. "IP Network Address Translator (NAT) Technology and Considerations," IETF, RFC 2663, 1999. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc2663.txt>
- [3] S. Kent and R. Atkinson. "Security Architecture for the Internet Protocol," IETF, RFC 2401, 1998. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc2401.txt>
- [4] R. Yuan and W. T. Strayer. "Virtual Private Networks: Technologies and Solutions." Addison-Wesley, April 2001.
- [5] D. Harkins and D. Carrel. "The Internet Key Exchange (IKE)," IETF, RFC 2409, 1998. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc2409.txt>
- [6] D. Maughan, M. Schertler, M. Schneider, and J. Turner. "Internet Security Association and Key Management Protocol (ISAKMP)," IETF, RFC 2408, 1998. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc2408.txt>
- [7] S. Kent and R. Atkinson. "IP Encapsulating Security Payload (ESP)," IETF, RFC 2406, 1998. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc2406.txt>
- [8] FreeS/WAN. Homepage at <http://www.freeswan.org/>, last accessed Mar. 2005.
- [9] S. Kent and R. Atkinson. "IP Authentication Header," IETF, RFC 2402, 1998. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc2402.txt>
- [10] B. Patel, B. Aboba, W. Dixon, G. Zorn, and S. Booth. "Securing L2TP Using IPsec," IETF, RFC 3193, 2001. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc3193.txt>
- [11] T. Kivinen, S. Swander, A. Huttunen, and V. Volpe. "Negotiation of NAT-Traversal in the IKE," IETF, RFC 3947, 2005. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc3947.txt>
- [12] A. Huttunen, S. Swander, V. Volpe, L. DiBurro, and M. Stenberg. "UDP Encapsulation of IPsec ESP Packets," IETF, RFC 3948, 2005. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc3948.txt>
- [13] J. Brustoloni and J. Garay. "Application-Independent End-to-End Security in Shared-Link Access Networks," in *Proc. Networking'2000*, IFIP, LNCS 1815:608-619, Springer-Verlag, May 2000. [Online] <http://www.cs.pitt.edu/~jcb/papers/net2000.ps>
- [14] G. Montenegro and M. Borella. "RSIP Support for End-to-End IPsec," IETF, RFC 3104, 2001. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc3104.txt>
- [15] S. Deering and R. Hinden. "Internet Protocol, Version 6 (IPv6)," IETF, RFC 2460, 1998. [Online] <ftp://ftp.rfc-editor.org/in-notes/rfc2460.txt>