An analysis of IPSec deployment performance issues in high and low powered devices

John Ronan[1], Steven Davy[1], and Judith Rossebø[23]

[1] Telecommunications Software & Systems Group,
Waterford Institute of Technology, Waterford, Ireland (jronan,sdavy@tssg.org)
[2] Telenor Research & Development, Telenor Communication II AS,
N-1331 Fornebu, Norway
[3] The Norwegian University of Science and Technology (NTNU),
Norway (judith.rossebo@telenor.com)

Abstract. Virtual Private Networks (VPNs) use the Internet or other network service as a backbone to provide a secure connection across a potentially hostile WAN. Such security guarantees provide the motivation for VPN deployment. This security does, however, come at a performance cost brought about by the increased processing overhead. This paper presents an investigation into these overheads. In particular, this investigation will consider different user resource availability based on the client platform in addition to router type and encryption algorithms.

Keywords: Security, Privacy, VPN, IPSec performance

1 Introduction

This paper summaries the results from the simulations of IPSec VPNs implemented on a range of devices and for several broad categories of users. User resources depend on the organisation and can vary from researchers with high speed networking capabilities (such as GEANT) to SMEs with more constrained bandwidth links. These organisations have very different resource availabilities and as such suffer from different consequences of IPSec performance overhead. Other associated users may be equipped with low powered devices such as PDAs and laptop computers and need to utilize IPSec services.

The work done for this paper uses both the IPv6 and IPv4 protocol stacks. The results presented in this paper address the need for evaluating the performance of IPSec under a range of conditions. These conditions are representative of various software and hardware platforms commonly available, corresponding to different user categories and usage patterns.

A number of user categories are represented in terms of bandwidth; specifically 100Mbps to replicate Fast Ethernet, 51Mbps to replicate OC-1 fiber optic links, 44Mbps

to replicate T3 speed, 34Mbps to replicate E3 speed, 10 Mbps to replicate Ethernet and fast DSL, and for those with very limited resources 4Mbps, 2Mbps and 1Mbps. High resource users are represented to have similar capabilities to our high-end servers and routers. The systems we use are a Compaq DL380 G2 Dual Xeon 2.4Ghz, a Dual AMD Opteron 64bit, and Dual or Single Pentium III's. For portable devices we propose to evaluate different client platforms (Linux/Windows/Macintosh) for their respective IPSec performance. For low powered devices we chose to evaluate Compaq iPAQs with PocketPC 2003. Tests were done for the TCP transport protocol and we tested to see how the additional overhead of IPSec affects performance both from a network and computational perspective.

1.1 Background

IPSec Protocol Suite IPSec is the security architecture for the Internet Protocol (IP). This protocol is applicable to both IPv4 and IPv6. The architecture is defined in [10] and addresses the following 4 elements:

- Security Protocols: Authentication Header (AH) [8] and Encapsulating Security Payload (ESP)[3].
- Security Associations: Definition, management and processing.[11]
- Key Management: The Internet Key Exchange (IKE) [11],[12],[13],[14].
- Algorithms: Requirements of the authentication and encryption algorithms.

Security Protocols Traffic Security is provided by two security protocols:

- The Authentication Header protocol [8] provides connectionless integrity and data origin authentication. There is also an optional anti-replay service available.
- The Encapsulating Security Payload protocol [3] [15] potentially provides two types of security service. The first is confidentiality via encryption and limited traffic flow confidentiality. The second type is connectionless integrity, data origin authentication and an anti-replay service.

Either of these protocols can be applied alone or in combination, thus providing the desired level of security. The IPSec security protocols are represented by headers that appear before the IP header in the IP packet. In our throughput tests we do not use the AH [8] as well as ESP as there is no evidence to say that the authentication available with ESP is any less secure than AH [9, p. 4].

Security Associations Information pertaining to the cryptographic algorithms and the associated parameters are distributed by the transmission of a *Security Parameter Index* (SPI). This index combined with the destination IP addresses and the type of protocol header (AH or ESP) determines the parameters of the IPSec processing which are represented by a Security Association (SA). There are two types of SAs:

– *Transport Mode SA*: This is a security association between two hosts, generally used to secure the traffic of the upper layer protocols.
– *Tunnel Mode SA*: This is a security association in an IP-in-IP tunnel, generally used in connecting to security gateways.

Key Management IPSec mandates support for two separate methods of cryptographic key and SA management: *Manual Key Management* and *Automatic Key and SA Management*. Manual key management is suitable in small static situations. For larger deployment scenarios Automatic Key and SA management is provided by using *Internet Key Exchange* (IKE). In these tests, automatic keying was used.

Algorithms The IPSec protocol suite does not define the authentication and encryption algorithms used in implementations. These are defined in individual RFCs per algorithm. Previous work [1] has also shown that ESP [3] using AES [5] to provide its encryption services is preferable in terms of both security and performance. For similar reasons MD5 [6, 7] is used to provide its authentication services. Algorithms used in these tests were:

– AES [5]
– HMAC-MD5 [6] [16]

Packet Payload Size IPSec Tunnel mode will add typically up to 100 bytes to an ethernet frame, 40 bytes for new IP Header, and up to 60 bytes for ESP related data. An Ethernet frame can send 1500 bytes in its payload, and from this we also subtract 20 bytes for the TCP header information and another 40 bytes for the original IP header as per [2], this means that before IPSec, TCP could transfer 1440 bytes of data, it can now only fit about 1340 bytes of data into the same frame, depending on the algorithm used. The IPSec gateway will have to fragment the packet before it is transmitted. The experiments performed below indicate this rule.

AH vs ESP In this paper we only analysed ESP and not AH, because ESP tunnel mode can provide the same level of protection as AH, when authentication is used. However when ESP is used in transport mode, the IP Header remains unprotected because only the packet payload is secured.

1.2 Objectives

We intend to show how IPSec (using AES) behaves under a variety of circumstances and taking the guesswork out of how one would have to dimension a security gateway or VPN endpoint to be capable of dealing with a certain level of traffic.The throughput figures may also be used e.g. in connection with a risk analysis to help determine whether IPSec VPNs protection provided as a countermeasure to ensure confidentiality is cost-efficient. In other words, do the benefits of applying IPSec VPNs outweigh the costs?

Also, we aim to show realistic throughput figures that can be expected from various client devices and operating systems. All measurements are done as per [17]. The Throughput measured in this paper refers to Tunnel Throughput mentioned in section 10.1.1 of [17] and is defined as the maximum rate through an IPSec tunnel at which none of the offered frames are dropped by the device under test. This is measured through the use of Iperf [21] traffic measurement utility. Also as per [4] a range of packet sizes were used. With these recommendations in mind we were able to define our scenarios so we could measure the necessary parameters correctly.

2 Scenarios

The version of IPSec used for this paper is built into the 2.6+ linux [18] kernels, for this paper the 2.6.5 kernel was used. There are a number of tools [19] that come with the kernel to aid in setting up the security associations needed for two hosts to speak IPSec, setkey command is used here. Setkey can setup either tunnel or transport mode security connection. For the iPaq, an evaluation version of movianVPN for the Pocket PC 2003 [24] was used.

The first scenario is as follows: a pair of routers route secure traffic between two separate networks, where the third network is assumed untrusted and simulates the Internet. After this scenario is set up, the bandwidth on the links is shaped to replicate common configurations in real world scenarios. The configurations that are replicated represent researchers with the high speed network capabilities (GEANT) and SME's with more constrained bandwidth links. This translates into running the experiments over 100Mbps, 51Mbps, 44Mbps, 34 Mbps and 10Mbps networks. The Clients are communicating with a Server on the far network and use varying packet sizes to replicate different usage patterns.

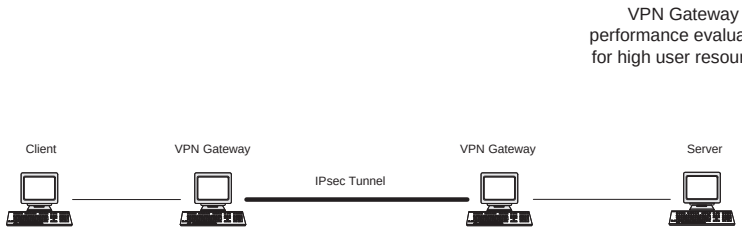


Fig. 1. High Powered Devices

In the second scenario, mobile and portable devices are configured in order to replicate real world scenarios. The configurations represent mobile users operating in an unsecured environment or in roadwarrior situations.

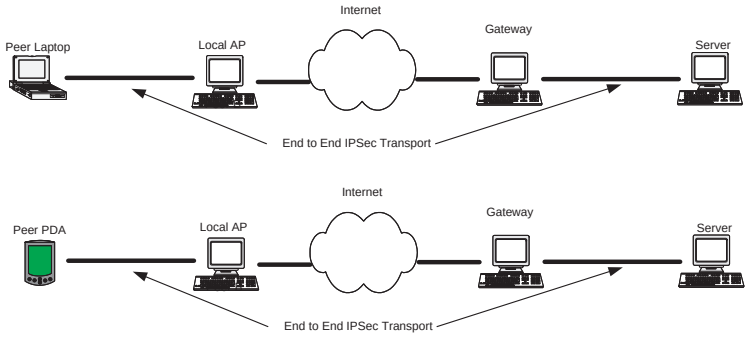


Fig. 2. Low Powered Devices

3 Descriptions of the Different Configurations

3.1 Hardware & Software configurations

There are three networks in this experiment all of which are IPv6 enabled. 2001:770:20:71::/64 is the Server network, 2001:770:20:70::/64 is the Client network and 2001:770:20:73::/64 is the assumed untrusted network. The Server is located at 2001:770:20:71::240, and the Client is located at 2001:770:20:70::4. The routers are set up to route traffic from Server to Client and the return path. Note though that IP forwarding must first be enabled with the following command on both routers before traffic can be routed. After this is set up, any packet received by the router, that is not destined for one of its interface addresses, is routed through the appropriate interface. This interface is determined from the routes set up on the router.

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

3.2 Equipment used

All of the machines involved were using Linux Kernel version 2.6.5 unless otherwise specified

- SERVER = Dual AMD Opteron 2400
  2001:770:20:71::240
- CLIENT = PIII 700 - PIII 550
  2001:770:20:70::/64
- GW70 = Dual Xeon 2.4Ghz Compaq DL380-G3
  2001:770:20:70::251 , 2001:770:20:73::251
- GW71 = Dual Xeon 2.4Ghz Compaq DL380-G3
  2001:770:20:71::4 , 2001:770:20:73::4
- PDA = iPaq with IPv4
- Laptop = Toshiba /w Linux , Windowx XP

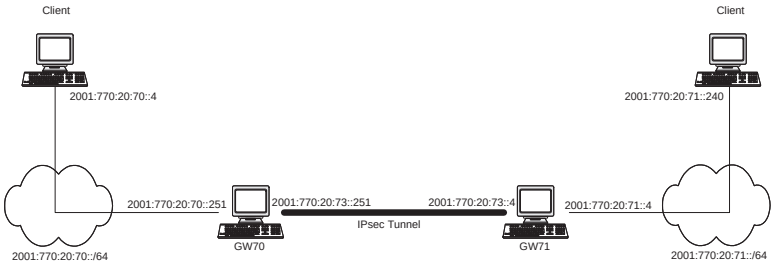


Fig. 3. Network Layout

Low Powered Devices These devices are a HP5500 iPaq with PPC2003 , a Tosihba Laptop with Windows and Linux, and a Mac OSX Power book. They will each connect to the VPN Server and communicate via iperf to measure performance, expect the clients here are IPSec endpoints. Only Tunnel throughput was measured, due to inconsistent method of CPU measurement and CPU power among the devices. In the case of the iPaq, an FTP client was used to down load a file from the server, and the time was compared with and without IPSec enabled.

3.3 Setting up the routes

The Server machine needs to know how to route traffic to the Client network, located in the 2001:770:20:70::/64 network. This traffic is routed to the Server Router.

```
ip -6 route add 2001:770:20:70::/64 via 2001:770:20:71::4
```

The Client machine needs to know how to route traffic to the Server network, located in the 2001:770:20:71::/64 network. This traffic is routed to the Client Router.

```
ip -6 route add 2001:770:20:71::/64 via 2001:770:20:70::251
```

The Server router *GW71* can route traffic directly to the Client router *GW70* via the network 2001:770:20:73::/64, so any traffic this router receives destined for the Client network is sent to GW70

```
ip -6 route add 2001:770:20:70::/64 via 2001:770:20:73::251
```

The Client router *GW70* can route traffic directly to the Server router *GW71*, so any traffic this router receives destined for the Server network is sent to the GW71.

```
ip -6 route add 2001:770:20:71::/64 via 2001:770:20:73::4
```

3.4 Setting up IPSec Tunnel mode

We used automatic keying to initialize the IPSec tunnel SA's for both authentication and encryption with the use of racoon [22] ike tool. The security association provides

confidentiality, data origin authentication, and connectionless integrity through the use of ESP. The cryptographic algorithm used for ESP confidentiality is AES, and for ESP authentication is MD5. The following is applied to the setkey command (through the -f option) on GW70.

```
flush;
spdflush;

spdadd 2001:770:20:70::/64 2001:770:20:71::/64 \
any -P OUT ipsec
esp/tunnel/2001:770:20:73::251
-2001:770:20:73::4/require;
spdadd 2001:770:20:71::/64 2001:770:20:70::/64 \
any -P IN ipsec
esp/tunnel/2001:770:20:73::4
-2001:770:20:73::251/require;
```

And the router on the GW71

```
flush;
spdflush;

spdadd 2001:770:20:70::/64 2001:770:20:71::/64 \
any -P IN ipsec
esp/tunnel/2001:770:20:73::251
-2001:770:20:73::4/require;
spdadd 2001:770:20:71::/64 2001:770:20:70::/64 \
any -P OUT ipsec
esp/tunnel/2001:770:20:73::4
-2001:770:20:73::251/require;
```

Spdflush and flush erase all IPSec associations in the kernel then, the ones that are used for the experiments are added. A Security Association (SA) was set up on the GW70 to GW71 and vice-versa. In summary, the above commands tell the kernel to apply ESP tunnel mode IPSec to packets originating from address 2001:770:20:71::/64 and with destination address 2001:770:20:70::/64. An association in the kernel is created for the opposite direction also. The packets are tunneled through the assumed untrusted network 2001:770:20:73::/64.

To setup racoon we modified the default configuration supplied with racoon so it would use aes and md5, a sample file can be found here [23].

A preshared key file was used to authenticate client connecting to the VPN server and to authenticate each VPN Gateway with each other.

3.5 Setting up IPSec Transport mode

IPSec Transport mode with authentication still dose not protect the outer IP header, because only the payload is secured. If AH were used with unauthenticated ESP, then the whole packet would be authenticated and the payload would be secure. Even though the header is still visible, it cannot be tampered with in this mode. Setup for this mode

is almost identical to that of IPSec Tunnel mode, only modifications in the setkey file have to be modified and can be seen below in the updated setkey configuration. The setup is now end to end and will protect packets between a specific source and destination.

```
flush;
spdflush;
spdadd 2001:770:20:70::4 2001:770:20:71::240 \
any -P IN ipsec
esp/transport//require;
spdadd 2001:770:20:71::251 2001:770:20:70::4 \
any -P OUT ipsec
esp/transport//require;
```

Setting up tc tc is a linux program used to control and shape traffic on an interface, although this program has quite a complex set of options, only one was used for our purposes.

```
tc qdisc add dev eth1 root tbf rate 34mbit \
burst 10kb latency 70ms
```

The above command will constrain the related network interface to a bandwidth capacity of 34 Mbps. The aim is to use tc to constrain the speed of the link between the two VPN Gateways. A command similar to this was issued on both routers before Iperf was started.

Setting up cyclesoak Cyclesoak [20] is a tool for accurately measuring system load and used to measure CPU utilization, by soaking up all idle cycles and calculating the load on the CPU. This tool is run on both routers, as it was here that we were interested in the processor overhead.

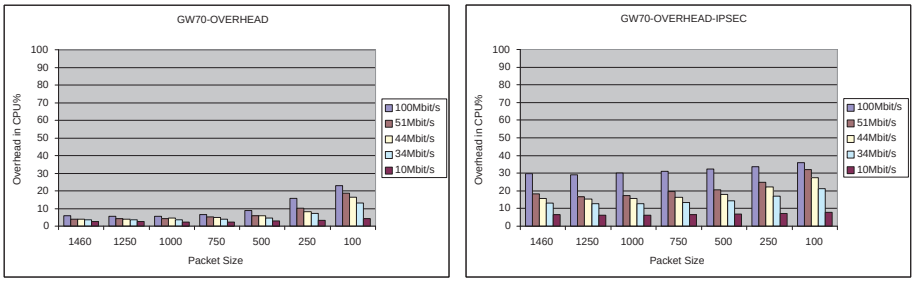
4 Performance Measurements

The main performance measure this paper is focused on is the Tunnel Throughput as described in section 10.1.1 of [17]. The throughput was measured with Iperf. Iperf [21] is a freely available, multi-platform bandwidth measurement tool. Iperf can measure many network related parameters, in this case we were really only interested in bandwidth, the processor overhead generated on the router and the relationship between them. As we wished to measure lower bandwidths than 100Mbps, the linux traffic controller *tc* was used to achieve lower throughputs. Iperf can control whether the traffic generated is UDP or TCP,and also the size of the packets to be generated can be specified. This is useful in simulating different usage patterns.

5 Performance Tests

5.1 High Powered Devices

Cron was used to synchronize the tests to be performed, NTP was used to synchronize the clocks on all participating machines where GW70 was the NTP server. xntpd was run on GW70 and ntpdate was run on all remaining participants. The objective of the client cron file was to run iperf for a duration of five minutes and record the throughput result to a file, perform a Client Iteration, and then run iperf again with a different packet size. The packet sizes considered here are 1460, 1250, 1000, 750, 500, 250 and 100 all in bytes and were chose to give a variety of packet sizes, and to demonstrate the optimal packet size rule. Iperf was iterated so that all packet sizes were tested, this was a Client Cycle. The Server Gateways were configured to modify the link speed after a Client Cycle has been completed. After each Client Iteration, the Server records the CPU utilization for that iteration. This translated into the Client doing a Client Cycle for each link speed being tested, ie 100 Mbps, 51 Mbps, 44 Mbps, 34 Mbps, and 10 Mbps. All the data was recorded to disk and graphs were generated using Microsoft Excel. The above tests were performed with no IPSec enabled and with IPSec enabled, and then compared. The graphs generated are outlined below.

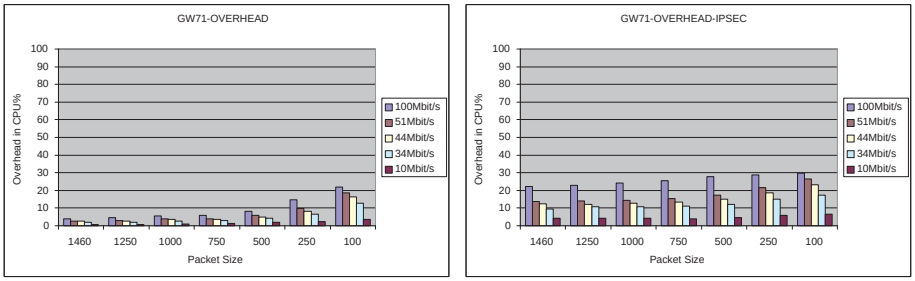


(a) (b)

Fig. 4. (a) GW70 Overhead No IPSec, (b)GW70 Overhead IPSec

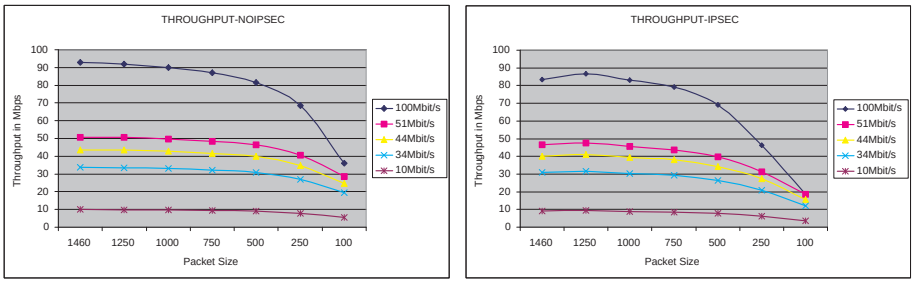
6 Conclusion

The results show that for less than 100Mbps links the IPSec throughput is about 90% of the throughput achieved where no IPSec is used on the link, when packet size is above 750 bytes, this is a promising result. This can be seen in Figure. 7 (a) for high powered devices. Figure. 7 (a) also shows that when the packet size is below 750 bytes, IPSec becomes less efficient. This can be explained by looking at Figure. 4 (b), and Figure. 5 (b) where more processing power is used when packet size is small, introducing latency that becomes more obvious when bandwidth increases. The same effect can be seen in low powered devices, when throughput is higher packet size becomes more of an



(a) (b)

Fig. 5. (a) GW71 Overhead No IPSec, (b)GW71 Overhead IPSec



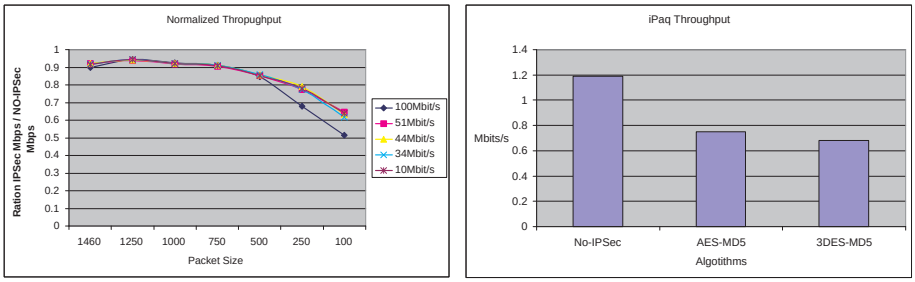
(a) (b)

Fig. 6. (a) Throughput No IPSec, (b) Throughput IPSec

issue. In Figure. 4 (a) and Figure. 5 (a) we can see that smaller packet size still has an effect on CPU overhead, and demonstrates that it must be a TCP issue. In actual fact the size of the packet effects the throughput when it is larger or below about 1340 bytes, which is calculated above as being the optimal packet payload size. Throughput begins to drop when the size of the packet payload is either smaller than the optimal or larger and needs to be fragmented. Figure. 7 (b) iPaq throughput shows results for low-powered devices such as the iPAQ that are consistent with previous measurements for high-powered devices, as it is clearly seen that AES-MD5 performs better than 3DES-MD5.

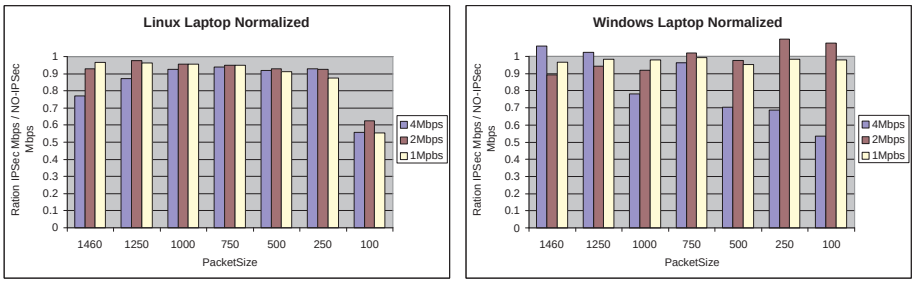
The work carried out in this paper will lead on into further investigation of the effect of packet size on higher bandwidth links, such as Gigabit links and multi-Gigabit links. Also we intend on doing further research in developing an IPSec Performance Evaluation Suite using IPSec Hardware acceleration to realise some methodology to measure IPSec performance parameters defined in [17].

Acknowledgments The research on which this paper reports has partly been supported by SEINIT, a European Union-supported IST FP6 Integrated project (001929): and the IKT 2003 project SARDAS (15295/431) funded by the Research Council of Norway.



(a) (b)

Fig. 7. (a) Normalized-Throughput, (b) iPaq Throughput



(a) (b)

Fig. 8. (a) Linux Normalized, (b) Windows Normalized

References

1. J. Ronan, P. Malone, M. Ó Foghlú, *Overhead Issues for Local Access Points in IPsec enabled VPNs*, IPS Workshop, Salzburg, February 2003. Retrieved: 3. April, 2003 from `http://www.ist-intermon.org/workshop/papers/09_01_vpn-overhead.pdf`
2. I. Raicu, S. Zeadally, *Impact pf IPv6 on End-User Applications*, ICT2003, Tahiti, February 2003.
3. S. Kent, R. Atkinson, *RFC 2406: IP Encapsulating Security Payload.*, November 1998.
4. S. Bradner, *RFC 1242: Benchmarking terminology for network interconnection devices.*, July 1991.
5. S. Franker, S. Kelly and R. Glenn, *RFC 3602: The AES Cipher Algorithm and Its Use with IPsec*, September 2003.
6. H. Krawczyk, M. Bellare, and R. Canetti, *RFC 2104: HMAC: Keyed-Hashing for Message Authentication*, February 1997.
7. C. Madson, R. Glenn, *RFC 2403: The Use of HMAC-MD5-96 within ESP and AH.*, November 1998.
8. S. Kent, R. Atkinson, *RFC 2402: IP Authentication Header*, November 1998.
9. R.Glynn, S. Kent, *The NULL Encryption Algorithm and Its Use With IPsec*, November 1998.
10. S. Kent, R. Atkinson,*RFC 2401: Security Architecture for the Internet Protocol.* November 1998.

11. D. Maughan, M. Schertler, M. Schneider, and J. Turner, *RFC 2408: Internet Security Association and Key Management Protocol.*, November 1998.
12. H. Orman, *RFC 2412: The OAKLEY Key Determination Protocol.*, November 1998.
13. D. Piper, *RFC 2407: The Internet IP Security Domain of Interpretation for ISAKMP.*, November 1998.
14. D. Harkins, D. Carrel, *RFC 2409: The Internet Key Exchange (IKE)*, November 1998.
15. R. Pereira, R. Adams, *RFC 2451: The ESP CBC-Mode Cipher Algorithms.*, November 1998.
16. C. Madson, R. Glenn, *RFC 2403: he Use of HMAC-MD5-96 within ESP and AH.*, November 1998.
17. M. bustos, T. Van Herck, M. Kaeo, *Terminology for Benchmarking IPSec Devices draft-ietf-bmwg-ipsec-term-03*, March 2004.
18. Linux, *http://www.linux.org*, Last Accessed, April 29^{th} 2004.
19. IPsec-Tools, *http://ipsec-tools.sourceforge.net*, Last Accessed, April 29^{th} 2004.
20. Cyclesoak, *http://www.zipworld.com.au/ akpm/linux/#zc*, Last Accessed, April 29^{th} 2004.
21. Iperf, *http://dast.nlanr.net/Projects/Iper*, Last Accessed, April 29^{th} 2004.
22. Racoon, *http://www.kame.net/racoon/*, Last Accessed, June 10^{th} 2004.
23. Racoon sample file, *http://www.kame.net/racoon/racoon.conf.5*, Last Accessed, June 10^{th} 2004.
24. movianVPN for PPC2003, *http://www.certicom.com/index.php?action=product, mvpn*, Last Accessed, June 10^{th} 2004.