The Effects of Authentication on AX.25 Packet Radio Data Transmission Time

Paul D. Wiedemeier, Ph.D., KE5LKY
Assistant Professor of Computer Science
The University of Louisiana at Monroe
Computer Science and Computer Information Systems Department
College of Business Administration
700 University Avenue, Monroe, Louisiana 71209
318-342-1856 (Work) or 318-396-1101 (Fax)
wiedemeier@ulm.edu, wiedemeierp@gmail.com, or KE5LKY@arrl.net

Abstract

This paper quantifies the time required to transmit 4 Kilobyte (KB), 8 KB, and 16 KB text files over a 2-meter AX.25 packet radio network using Gnu Privacy Guard (GPG), Secure Socket Layer and Transport Layer Security (SSL/TLS), and Internet Protocol Security (IPsec) authentication software. Our results show that less time is required to transmit data using GPG authentication than either SSL/TLS authentication or IPsec authentication. The discussion contained in this paper will benefit those amateur radio operators who provide data communication for organizations that have signed a *Memorandum of Understanding* with the American Radio Relay League, such as the American Red Cross and the Salvation Army.

Key Words

AX.25, Packet Radio, Authentication Software, Gnu Privacy Guard, Secure Socket Layer, Transport Layer Security, Internet Protocol Security, Call Sign Spoofing, Message, Data, Transmission Time

Introduction

Within an AX.25 packet radio network, call sign "spoofing" is a trivial action because an unscrupulous individual can easily configure their AX.25 software to transmit messages using any United States Federal Communication Commission (FCC) call sign. Amateur radio operators who receive a "spoofed" message are often unable to determine whether (1) the message was actually transmitted by the individual associated with the call sign and/or (2) the received message was the one actually transmitted. A solution to this problem is to use authentication software. With respect to amateur radio, the FCC Part 97.219 rule requires that stations *"authenticate the identity of the station from which it accepts communications on behalf of the system"* (USFCC, 2009).

In 2004, the American Radio Relay League's High-Speed Multimedia & Networking Workgroup published a report requesting "… the support of the ARRL Board of Directors for development and filing of a 'Notice of Proposed Rulemaking' permitting the use of encryption and strong security protocols on domestic transmissions above 50 MHz" (Toth, 2004). Specifically, the authors' claimed that "… licensees in the Amateur Radio Service need to be free to utilize ... industry-standard security

and authentication tools to protect the integrity of their stations". These views are shared by (Champa, 2004) and (Rotolo, 2006).

We, too, advocate the use of authentication software when transmitting messages over AX.25 packet radio networks. As such, this paper explores the use of Gnu Privacy Guard (GnuPG or GPG), Secure Socket Layer and Transport Layer Security (SSL/TLS), and Internet Protocol Security (IPsec) authentication software when transmitting messages (i.e. data). Specifically, we wish to quantify the time required to transmit data using these three authentication software compared to unauthenticated data transmissions.

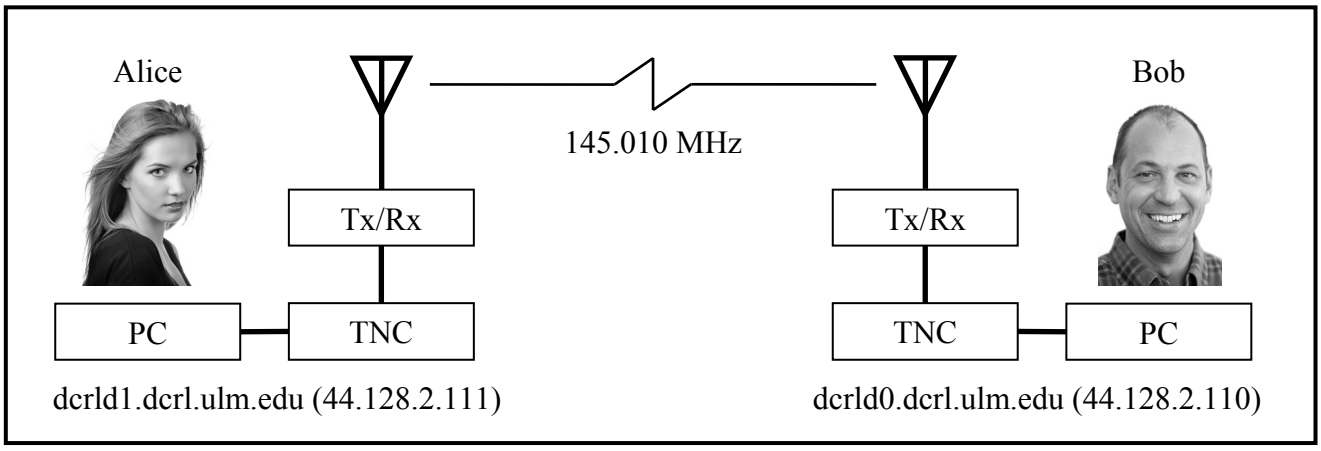

Figure 1: The logical hardware configuration of our AX.25 packet radio stations dcrld0 and dcrld1.

For this research, authentication refers to the ability of an individual or station to determine whether (1) the sender of a received message is who they assert they are and/or (2) the message received is what was transmitted (Stallings, 2007). To adhere to the FCC Part 97.113 rule, messages were not, at any time during the transmission, encrypted or *"encoded for the purpose of obscuring their meaning"* (USFCC, 2009). While GPG, SSL/TLS, and IPsec, by default, provide data encryption and authentication, we only used authentication when transmitting messages.

Table 1: Specific software used to conduct our research.

Software	Associated Website or RFC
Apache Web Server	http://www.apache.org/
cURL	http://curl.haxx.se/
UNIX time command	http://www.kernel.org/doc/man-pages/online/pages/man1/time.1.html
Gnu Privacy Guard	http://www.gnupg.org/
OpenSSL	http://www.openssl.org/
Secure Socket Layer/Transport Layer Security	http://datatracker.ietf.org/doc/rfc5246/
Internet Protocol Security	http://datatracker.ietf.org/doc/rfc4301/
Wireshark	http://www.wireshark.org/

Materials

To conduct our research, we constructed an AX.25 packet radio network from a pair of Kenwood TM-271 2-meter transceivers, two Kantronics KPC-3+ 1200 bits per second terminal node controllers, a Diamond X30A antenna, and a Diamond X50A antenna. To transmit data, we used two Dell OptiPlex GX270 personal computers (PC) running Fedora Linux, core 8, which we named dcrld1.dcrl.ulm.edu (dcrld1) and dcrld0.dcrl.ulm.edu (dcrld0). Figure 1 shows the logical hardware configuration of our AX.25 packet radio network. For a thorough discussion of how we configured our computers, transceivers, and terminal node controllers, we refer the reader to (Wiedemeier, 2009 & 2008).

Table 1 lists the software we used to conduct our research. Our software choices were driven by five requirements. First, we wanted to investigate how application layer, transport layer, and network layer authentication software influence data transmissions over AX.25 packet radio networks. See Figure 2. Second, we required the use of data transmission server software (e.g. FTP server or web server) that would allow us to evaluate each authentication software independently. In this way, we could determine each authentication software's overall effect on data transmission time.

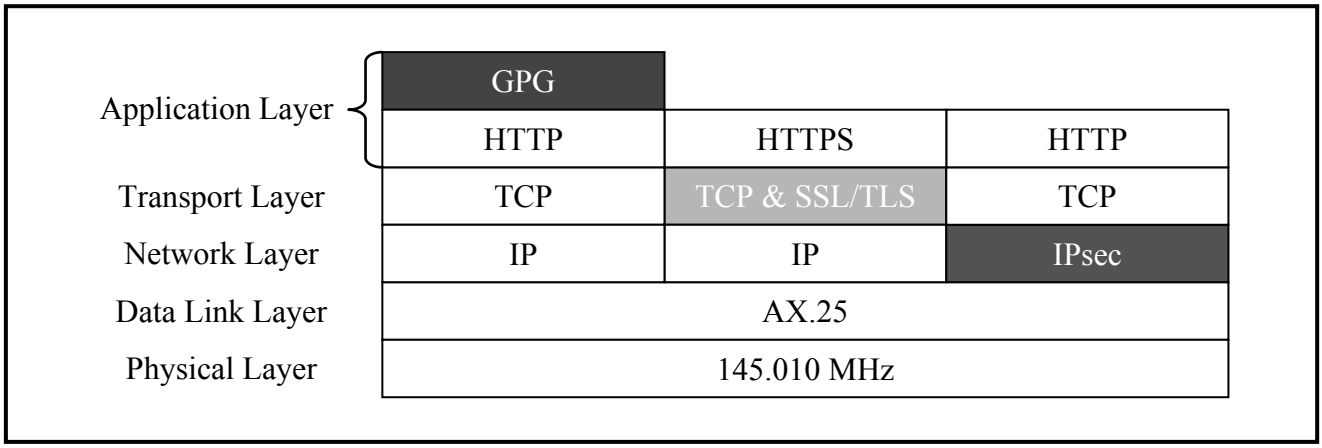

Figure 2: An "authentication enabled" generic data communication protocol stack.

Third, we required the use of command line oriented client software that would allow us to retrieve data from the data transmission server software we chose to use. Additionally, we required that the client software display the elapsed time associated with a data transmission. Fourth, we required the use of network protocol analyzer software to inspect every packet transmitted between the client and server. Last, the software we used must be open source and must be installed on the Fedora Linux operating system.

The data transmission server software we chose was Apache web server, version 2.2.9. Our decision to use an Apache web server was due to two unique features associated with the software. First, adding authentication support to an Apache web server is well documented and manageable for a knowledgeable UNIX system administrator. Second, the client software we chose can direct a secure Apache web server to use a specific encryption and authentication cipher when transmitting data.

We installed and configured a standard Apache web server, as well as a secure Apache web server, on PC dcrld1. We next created three text files of size 4 Kilobyte (KB), 8 KB, and 16 KB, which we named

text4KB.txt, text8KB.txt, and text16KB.txt respectively. These three files were copied to the directory /var/www/html on PC dcrld1. We refer the reader to (Wiedemeier, 2009 & 2008) for a discussion about the contents of these three files.

We used the cURL client software on PC dcrld0 to retrieve the text files text4KB.txt, text8KB.txt, and text16KB.txt from the standard and secure Apache web servers on PC dcrld1.dcrl. We chose cURL because it is able to send and receive data using many data communication protocols, including HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP, or FILE (Stenberg, 2010). We also chose cURL because it has a *--ciphers* command line argument that can instruct a secure web server to use specific authentication and encryption ciphers during data transmission.

While cURL will display the elapsed time in seconds associated with each file transmission, we also used the ubiquitous UNIX *time* command to verify that the elapsed transmission time returned by cURL matched that returned by the UNIX *time* command. For each file transmitted, the date of transmission and elapsed transmission time were recorded in a Microsoft Excel spreadsheet.

The application layer authentication software we chose to use was Gnu Privacy Guard (GnuPG or GPG).

> "GnuPG stands for GNU Privacy Guard and is GNU's tool for secure communication and data storage. It can be used to encrypt data and to create digital signatures. It includes an advanced key management facility and is compliant with the proposed OpenPGP Internet standard as described in RFC 2440. As such, it is aimed to be compatible with PGP from PGP Corp. and other OpenPGP tools" (Ellmenreich & Koch, 2010).

The transport layer authentication software we chose to use was Transport Layer Security (TLS). "TLS and its predecessor, Secure Socket Layer (SSL), are cryptographic protocols that allow client/server applications to communicate across the Internet in a way designed to prevent eavesdropping and message tampering". (Dierks & Rescorla, 2008). SSL/TLS is often used by organizations to secure data transmitted between web browsers/clients and a web server.

The network layer authentication software we chose to use was Internet Protocol (IP) Security. Internet Protocol Security (IPsec) is a protocol suite for providing secure IP communications by authenticating and encrypting each IP packet transmitted. IPsec also includes protocols for establishing authentication between users and hosts at the beginning of a communication session and negotiation of the cryptographic keys to be used. (Kent, 2005).

As discussed in the Introduction, the FCC Part 97.113 rule prohibits the transmission of encrypted data over amateur frequencies. Because SSL/TLS and IPsec, by default, encrypt and authenticate transmitted data, we configured both software to use RSA authentication and NULL encryption. We then used the wireshark network protocol analyzer to ensure that the software performed only authentication during data transmission.

Methods

The activities associated with installing, configuring, and transmitting data using the cURL, the UNIX *time*, and the GPG, SSL/TLS, and IPsec authentication software are discussed in the following subsections. To illustrate how we installed, configured, and used the authentication and associated

software, let us assume that two individuals, Alice and Bob, wish to transmit data over an AX.25 packet radio network. From Figure 1, we see that Alice and Bob own and manage PCs dcrld1 and dcrld0 respectively.

Apache Web Server

To install the standard Apache web server, as root on dcrld1, Alice executes the command *yum groupinstall "Web Server"*. This command installs several software packages, including httpd, httpd manual, https modules, mod_ssl, Apache modules, PHP, perl, python. To configure a secure Apache web server Alice completes the activities discussed in the SSL/TLS Authentication subsection below.

Wireshark Network Protocol Analyzer Configuration

To ensure that all data transmissions are conducted without encryption, Bob uses the wireshark network protocol analyzer to capture and view all packets transmitted between dcrld0 and dcrld1. To use wireshark, he completes the following activities.

1. As root on dcrld0, Bob executes the command *yum install wireshark* to install the network protocol analyzer software.
2. Using his account on dcrld0, Bob executes the command *wireshark* to start the network protocol analyzer.
3. After the wireshark program starts, he selects the "capture" tab and then selects "interfaces" from the dropdown menu.
4. On the "Capture Interfaces" pop-up window, he clicks the "option" button associated with the "any" interface.
5. On the "Capture Options" pop-up window, he enters the text "net 44.128.2.0 mask 255.255.255.0" in the "Capture Filter" textbox.
6. Before Bob initiates a file request, he selects the "capture" tab and then selects "start" from the dropdown menu to begin capturing packets.
7. He can now select and view any or all packets transmitted between dcrld0 and dcrld1.
8. To end the packet capture, Bob selects the "capture" tab and then selects "stop" from the dropdown menu.

No (i.e. "None") Authentication

To transmit data without authentication (i.e. "None"), together, Alice and Bob complete the following activities.

1. As root on dcrld1, Alice places the text files text4KB.txt, text8KB.txt, and text16KB.txt in the directory /var/www/html.
2. Using his account on dcrld0, Bob executes the command *time curl **http**://dcrld1.cs.ulm.edu/text4KB.txt > /tmp/text4KB.txt*.
3. The httpd daemon on dcrld1.cs.ulm.edu receives the request and sends file text4KB.txt to the cURL program executed by Bob.
4. Bob records the transmission time, in minutes and seconds, returned by the UNIX *time* command in a Microsoft Excel spreadsheet. See Table A1 in the Appendix.

5. To obtain twenty transmission of the file text4KB.txt, Bob completes activities 2 through 4 nineteen additional times.
6. Bob computes an average transmission time from the twenty transmission times collected and records this data in the Microsoft Excel spreadsheet. See Table A1 in the Appendix.
7. Bob completes activities 2 through 6 for files text8KB.txt and text16KB.txt. See Table A1 in the Appendix.

GPG Authentication

To transmit data using GPG authentication, together, Alice and Bob complete the following activities. Notice in activity 7, Alice and Bob must securely exchange electronic copies of their GPG public keys.

1. As root on dcrld1, Alice executes the command *yum install gpg* to install the GPG software on dcrld1.
2. As root on dcrld0, Bob executes the command *yum install gpg* to install the GPG software on dcrld0.
3. Using her account on dcrld1, Alice creates a GPG public and private key pair by executing the command *gpg --gen-key*.
4. Alice creates a text file that contains her GPG public key by executing the command *gpg --export --armor "Alice" > Alice_GPG_public_key.txt*.
5. Using his account on dcrld0, Bob creates a GPG public and private key pair by executing the command *gpg --gen-key*.
6. Bob creates a text file that contains his GPG public key by executing the command *gpg --export --armor "Bob" > Bob_GPG_public_key.txt*.
7. In a secure manner, Alice and Bob exchange electronic copies of the text files that contain their respective GPG public keys.
8. Using her account on dcrld1, Alice places Bob's GPG public key on her GPG key ring by executing the command *gpg --import Bob_GPG_public_key.txt.*
9. Alice determines the fingerprint of Bob's GPG public key by executing the command *gpg --list-keys --fingerprint "Bob"*.
10. Alice signs Bob's GPG public key by executing the command *gpg --sign-key [Bob's GPG public key fingerprint here]*. In doing so, Alice now "trusts" all files signed by Bob's GPG private key.
11. Using his account on dcrld0, Bob places Alice's GPG public key on his GPG key ring by executing the command *gpg --import Alice_GPG_public_key.txt*.
12. Bob determines the fingerprint of Alice's GPG public key by executing the command *gpg --list-keys --fingerprint "Alice"*.
13. Bob signs Alice's GPG public key by executing the command *gpg --sign-key [Alice's GPG public key fingerprint here]*. In doing so, Bob now "trusts" all files signed by Alice's GPG private key.
14. Using her account on dcrld1, Alice "clearsigns" the text file text4KB.txt using her GPG private key by executing the command *gpg --clearsign text4KB.txt*. This command creates a new file named text4KB.txt.asc. An example of a GPG clearsigned file is shown in Figure 3. The unencrypted data portion of the file shown in Figure 3 has been truncated due to space constraints.
15. As root on dcrld1, Alice renames the file text4KB.txt.asc as text4KB.txt.dcrld1.asc and places it in the directory /var/www/html.

16. Using his account on dcrld0, Bob executes the command *time curl **http**://dcrld1.cs.ulm.edu/text4KB.txt.dcrld1.asc > /tmp/text4KB.txt.dcrld1.asc*.
17. The httpd daemon on dcrld1.cs.ulm.edu receives the request and sends file text4KB.txt.dcrld1.asc to the cURL program executed by Bob.
18. Bob records the transmission time, in minutes and seconds, returned by the UNIX *time* command in a Microsoft Excel spreadsheet. See Table A2 in the Appendix.
19. To verify that the file text4KB.txt.dcrld1.asc was (1) signed by Alice's GPG public key and (2) that the contents of the file were not changed or modified during transmission, Bob executes the command *gpg --verify /tmp/text4KB.txt.dcrld1.asc*.
20. To obtain twenty transmission of the file text4KB.txt.dcrld1.asc, Alice and Bob complete activities 14 through 19 nineteen additional times.
21. Bob computes an average transmission time from the twenty transmission times collected and records this data in the Microsoft Excel spreadsheet. See Table A2 in the Appendix.
22. Alice and Bob complete activities 14 through 21 for files text8KB.txt.dcrld1.asc and text16KB.txt.dcrld1.asc. See Table A2 in the Appendix.

Figure 3: An example GPG "clearsigned" file.

SSL/TLS Authentication

To transmit data using SSL/TLS authentication, together, Alice and Bob complete the following activities. Notice in activity 1.g, Alice must find a secure method to provide Bob with an electronic copy of dcrld1's SSL/TLS certificate. Notice, also, in activities 1.r, 2.f, and 3.a, Alice and Bob are requesting files from dcrld1's secure Apache web server.

1. As root on dcrld1, Alice completes the following activities.

    a. She executes the command *cd /etc/pki/tls/certs* to enter this directory.
    b. She executes the command *make dcrld1.key* to generate a RSA private key.
        i. Executing the command *openssl genrsa -des3 1024 > dcrld1.key* accomplishes the same task.
    c. She executes the command *make dcrld1.csr* to create a *certificate signing request* (CSR) for an SSL/TLS certificate.
        i. Executing the command *openssl req -utf8 -new -key dcrld1.key -out dcrld1.csr* accomplishes the same task.
    d. She executes the command *make dcrld1.crt* to generate a self-signed SSL/TLS certificate.
        i. Executing the command *openssl req -utf8 -new -key dcrld1.key -x509 -days 365 -out dcrld1.crt -set_serial 0* accomplishes the same task.
    e. To move the private key to the appropriate directory, Alice executes the command *mv dcrlc1.key ../private/dcrld1.key*.
    f. She executes the command *openssl x509 -text -in dcrld1.crt > dcrld1.pem* to create a *privacy enhanced mail* (PEM) formatted file of the dcrld1.crt self-signed SSL certificate. PEM files are used to exchange SSL/TLS certificates between computers.
    g. In a secure manner, Alice provides Bob with an electronic copy of the file dcrld1.pem.
    h. She executes the command *hostname dcrld1* to set the hostname of her computer.
    i. She executes the command *domainname dcrl.ulm.edu* to set the domainname of her computer.
    j. To add network support for dcrld0, she executes the command *echo "44.128.2.110 dcrld0.dcrl.ulm.edu dcrld0" >> /etc/hosts*.
    k. To add network support for dcrld1, she executes the command *echo "44.128.2.111 dcrld1.dcrl.ulm.edu dcrld1" >> /etc/hosts*.
    l. She executes the command *vi /etc/httpd/conf/httpd.conf* to edit this file.
        i. She modifies the line "*ServerName*" so that it reads "*ServerName dcrld1.dcrld.ulm.edu*".
        ii. She modifies the line "*ServerAdmin*" so that it reads "*ServerAdmin [Alice's email address]*".
    m. She executes the command *vi /etc/httpd/conf.d/ssl.conf* to edit this file.
        i. She modifies the line "*ServerName*" so that it reads "*ServerName dcrld1.dcrld.ulm.edu*".
        ii. She modifies the line "*SSLCertificateFile*" so that it reads "*SSLCertificateFile /etc/pki/tls/certs/dcrld1.crt*".
        iii. She modifies the line "*SSLCertificateKeyFile*" so that it reads "*SSLCertificateKeyFile /etc/pki/tls/private/dcrld1.key*".
        iv. She modifies the line "*SSLCipherSuite*" and adds "*:NULL*" at end to add support for NULL encryption message transmission.
    n. She executes the command *cp ca-bundle.crt ca-bundle_ORIG.crt* to retain a copy of dcrld1's original certificate bundle file.
    o. She executes the command *cat dcrld1.crt >> ca-bundle.crt* to append dcrld1's certificate to dcrld1's certificate bundle file.
    p. She executes the command *service httpd stop* to stop the httpd daemon.
    q. She executes the command *service httpd start* to start the httpd daemon.

r. She executes the command *curl --verbose **https**://dcrld1.dcrld.ulm.edu* to test dcrld1's secure Apache web server.
s. She places the text files text4KB.txt, text8KB.txt, and text16KB.txt in the directory /var/www/html.

2. As root on dcrld0, Bob completes the following activities.

a. He executes the command *cd /etc/pki/tls/certs* to enter this directory.
b. He executes the command *cp ca-bundle.crt ca-bundle_ORIG.crt* to retain a copy of the dcrld0's original certificate bundle file.
c. He executes the command *cat dcrld1.pem >> ca-bundle.crt* to append dcrld1's certificate to dcrld0's certificate bundle file, where dcrld1.pem is the PEM file created by Alice.
d. To add network support for dcrld0, he executes the command *echo "44.128.2.110 dcrld0.dcrl.ulm.edu dcrld0" >> /etc/hosts*.
e. To add network support for dcrld1, he executes the command *echo "44.128.2.111 dcrld1.dcrl.ulm.edu dcrld1" >> /etc/hosts*.
f. He executes the command *curl --verbose **https**://dcrld1.dcrld.ulm.edu* to test dcrld1's secure Apache web server.

3. Using his account on dcrld0, Bob completes the following activities.

a. Bob executes the command *time curl --ciphers rsa_null_md5 **https**://dcrld1.dcrl.ulm.edu/text4KB.txt > /tmp/text4KB.txt*. The cURL *--ciphers rsa_null_md5* command line argument instructs the secure web server on dcrld1 to use RSA authentication and NULL encryption during transmission.
b. The httpd daemon on dcrld1.cs.ulm.edu receives the request and sends file text4KB.txt to the cURL program executed by Bob.
c. Bob records the transmission time, in minutes and seconds, returned by the UNIX *time* command in a Microsoft Excel spreadsheet. See Table A3 in the Appendix.
d. To obtain twenty transmission of the file text4KB.txt, Bob completes activities 3.a, 3.b, and 3.c nineteen additional times.
e. Bob computes an average transmission time from the twenty transmission times collected and records this data in the Microsoft Excel spreadsheet. See Table A3 in the Appendix.
f. Bob completes activities 3.a through 3.e for files text8KB.txt and text16KB.txt. See Table A3 in the Appendix.

IPsec Authentication

To transmit data using IPsec authentication, together, Alice and Bob complete the following activities. Notice, unlike GPG and SSL/TLS authentication, Alice and Bob do not exchange keys or certificates when using IPsec authentication.

1. As root on dcrld1, Alice completes the following activities.

a. She executes the command *system-config-network* to start the Network Configuration tool and then completes the following activities after the program starts.
i. She selects the "IPsec" tab and then click the "New" button to create new IPsec configuration

ii. She enters "packet" as the "Nickname", but does not check the "Activate the connection when the computer starts" check box.
iii. She selects the "Host to Host encryption" radio button.
iv. She selects "Auto encryption mode selection via IKA (raccoon)" radio button.
i. She enters "44.128.2.110", which is dcrld0's IP address, as the "Remote IP address".
v. She enters "SOMEAUTHPHRASE" as the "Authentication key". We suggest using the authentication phrase used to create your GPG public and private keys.
vi. She applies the IPsec configurations.

b. She executes the command *vi /etc/raccoon/raccoon.conf* to edit this file.
   i. In the *sainfo {}* block, she modifies the "*encryption_algorithm*" line to read "*encryption_algorithm null_enc;*".
c. She starts the IPsec "packet" interface by executing the command *ifup packet*.
   i. Alice can stop the IPsec "packet" interface by executing the command *ifdown packet*.
d. Alice places the text files text4KB.txt, text8KB.txt, and text16KB.txt in the directory /var/www/html.

2. As root on dcrld0, Bob completes the same activates completed by Alice shown above, except 1.d. With respect to activity 1.a.i. above, he would enter "44.128.2.111" as dcrld1's "Remote IP address".

3. Using his account on dcrld0, Bob completes the following activities.

   a. Bob executes the command *time curl **http**://dcrld1.cs.ulm.edu/text4KB.txt > /tmp/text4KB.txt*.
   b. The httpd daemon on dcrld1.cs.ulm.edu receives the request and sends file text4KB.txt to the cURL program executed by Bob.
   c. Bob records the transmission time, in minutes and seconds, returned by the UNIX time command in a Microsoft Excel spreadsheet. See Table A4 in the Appendix.
   d. To obtain twenty transmission of the file text4KB.txt, Bob completes activities 3.a, 3.b, and 3.c nineteen additional times.
   e. Bob computes an average transmission time from the twenty transmission times collected and records this data in the Microsoft Excel spreadsheet. See Table A4 in the Appendix.
   f. Bob completes activities 3.a through 3.e for files text8KB.txt and text16KB.txt. See Table A4 in the Appendix.

Results

The time required to transmit the 4 KB, 8 KB, and 16 KB text files between our two AX.25 packet radio stations using the GPG, SSL/TLS, and IPsec authentication software is shown in Figure 4 and displayed as "long dash", "square dot", and "round dot" lines respectively. The "solid" "None" line represents data transmission time without authentication. The data used to create the plots shown in Figure 4 are shown in Tables A1, A2, A3, and A4 in the Appendix. A plot of no (i.e. "None") authentication is shown in Figure A1 in the Appendix and plots of GPG, SSL/TLS, and IPsec authentication versus no (i.e. "None") authentication are shown in Figures A2, A3, and A4 in the Appendix.

With respect to Figure 4, the data show that more time is required to transmit the text files using GPG, SSL/TLS, and IPsec authentication compared to no (i.e. "None") authentication. However, two "features" should be noted. First, less time is required to transmit the 4 KB, 8 KB, and 16 KB text files using GPG authentication versus SSL/TLS authentication. Second, more time is required to transmit the 4 KB text file using IPsec authentication compared to GPG authentication, but less time is required to transmits the same file using IPsec authentication compared to SSL/TLS authentication. However, more time is required to transmit the 8 KB and 16 KB text files using IPsec authentication compared to both GPG authentication and SSL/TLS authentication. We discuss both "features" in the following paragraphs.

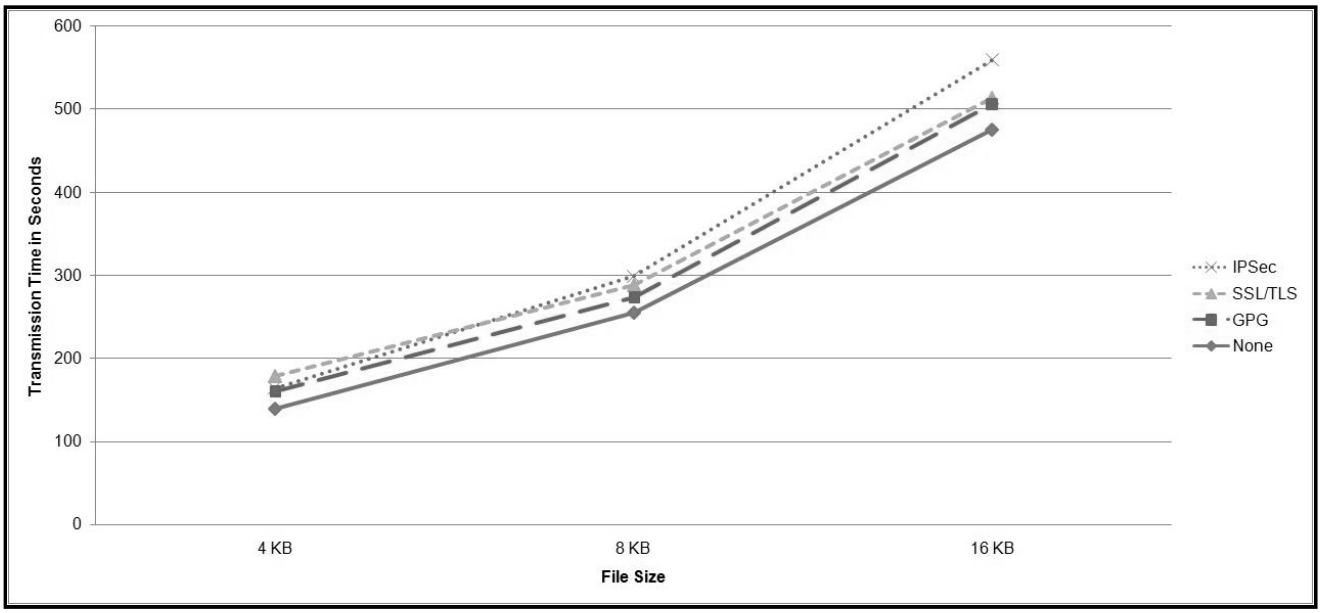

Figure 4: GPG, SSL/TLS, IPsec, and No (i.e. "None") Authentication Data Transmission Time.

As shown in Figure 4, and Figures A2 and A3 in the Appendix, the plots of GPG authentication and SSL/TLS authentication transmission times mirror that of no (i.e. "None") authentication transmission times. The reason for this "mirroring" is based on the amount of data transmitted by each authentication software. For example, when transmitting a 4 KB file using no (i.e. "None") authentication, the amount of data transmitted is 4 KB plus the overhead data transmitted by the Apache web server. The amount of data transmitted using GPG authentication is the 4 KB file size plus the size in bytes of the GPG clearsigned authentication header plus the overhead associated with the Apache web server. The amount of data transmitted using SSL/TLS authentication is 4 KB file size plus the SSL/TLS authentication header plus the overhead associated with the Apache web server. Overall, we see that the amount of authentication in bytes required by GPG and SSL/TLS to transmit the 4 KB, 8 KB, and 16 KB text files remains constant as the file size increases. This result is illustrated in Figure 5, where the percentage of GPG an SSL/TLS authentication decreases as file size increases. For text files larger than 16 KB, we expect the percentage of GPG and SSL/TLS authentication, relative to file size, to decrease because the size of GPG and SSL/TLS authentication headers remains fixed.

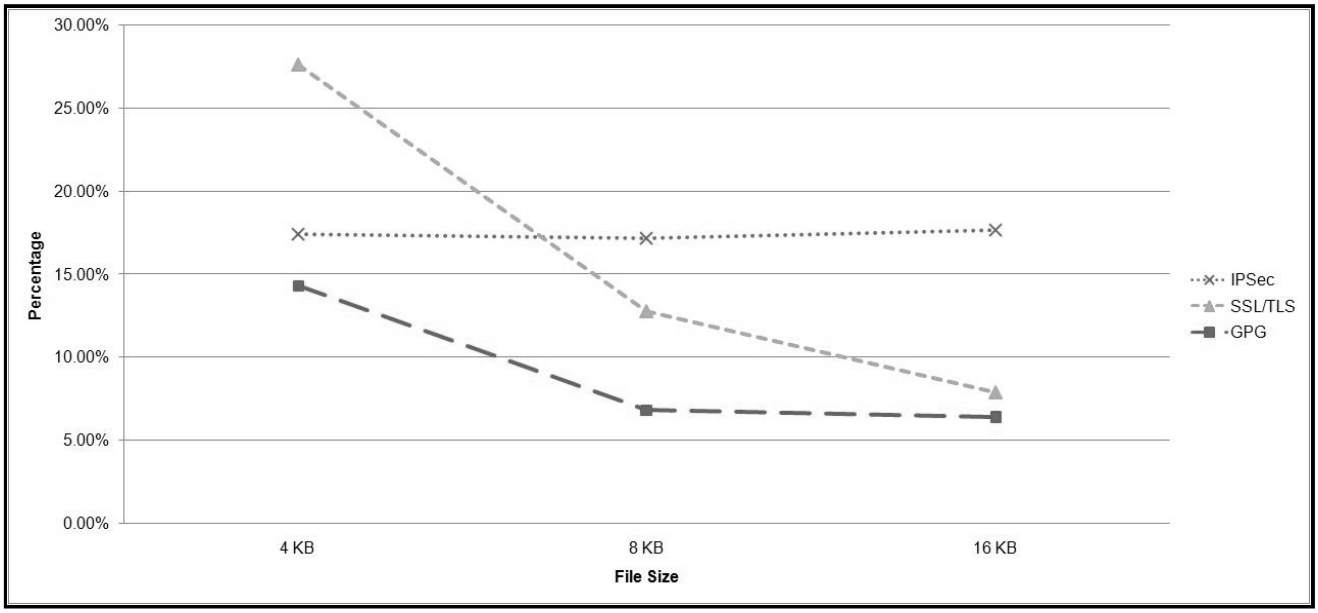

Figure 5: GPG, SSL/TLS, and IPsec Authentication Data Transmission Time as a Percentage of No (i.e. "None") Authentication Data Transmission Time.

As shown in Figure 4 and Figure A4 in the Appendix, the time required by IPsec authentication to transmit 4 KB, 8 KB, and 16 KB text files does not mirror that required by no (i.e. "None") authentication. This is because IPsec authenticates each packet transmitted. Specifically, as file size increases, the number of packets transmitted increases, which increases the amount of IPsec authentication transmitted. As shown in Figure 5, the percentage of authentication overhead required by IPsec, compared to no (i.e. "None") authentication is approximately 17% of the file size transmitted.

Conclusion

From our results, we advocate using GPG authentication when transmitting messages over AX.25 packet radio networks because it requires less transmission time compared to SSL/TLS and IPsec authentication. However, what the data in our figures and tables do not show is the time required to install and configure the authentication software, and the knowledge required by the individual responsible for configuring the Apache web server. The author hopes that the activities discussed in the Methods section of this paper, which Alice and Bob must complete to authenticate data, provide the reader with a general "feel" for the work and knowledge required to use the GPG, SSL/TLS, and IPsec authenticate software.

Acknowledgements

The author graciously thanks Allison M.D. Wiedemeier, Ph.D., for reading the first draft of this paper. The author's research is supported through funds provided by The University of Louisiana at Monroe (ULM) College of Business Administration and the ULM Digital Communication Research Laboratory.

References

Champa, John (K8OCL). (2004). HSMM and Information Security. *CQ VHF Magazine*, Fall, pp. 53-56.

Dierks, T. & Rescorla, Eric. (2008, August). *The Transport Layer Security Protocol, Version 1.2*. Request for Comments 5246. Retrieved July 26, 2010 from http://datatracker.ietf.org/doc/rfc5246/.

Ellmenreich, Nils & Koch, Werner. (2010). *Gnu Privacy Guard Frequently Asked Questions*. Retrieved July 26, 2010 from http://www.gnupg.org/documentation/faqs.html.

Kent, Stephen. (2005, December). *IP Authentication Header*. Request for Comments 4302. Retrieved July 26, 2010 from http://datatracker.ietf.org/doc/rfc4302/.

Rotolo, Don (N2IRZ). (2006, August). Data Encryption is Legal! *CQ Magazine*, vol. 62, no. 8, pp. 50-52.

Stallings, William. (2007). *Data and Computer Communication*, pp. 713. Upper Saddle River, NJ: Pearson Prentice Hall Publishers.

Stenberg, Daniel. (2010). *cURL man Page*. Retrieved July 26, 2010 from http://curl.haxx.se/docs/manpage.html.

Toth, Paul (NA4AR). (2004, June). *Security & Data Integrity on a Modern Amateur Radio Network*. American Radio Relay League. High Speed Multimedia & Networking Working Group.

United States Federal Communication Commission (USFCC). (2009). *Part 97 – Amateur Radio Service*. Retrieved July 26, 2010 from http://www.gpo.gov/fdsys/pkg/CFR-2009-title47-vol5/pdf/CFR-2009-title47-vol5-part97.pdf.

Wiedemeier, Paul (KE5LKY). (2009, September). "Comparing 2-Meter Packet Radio Data Unicasts and Multicasts". *Proceedings of the 28th ARRL and TAPR Digital Communications Conference*, pp. 114-129.

Wiedemeier, Paul (KE5LKY). (2008, November). Using Udpcast to IP Multicast Data over Packet Radio Networks. *American Radio Relay League QEX Magazine*, November/December, issue 251, pp. 43-49.

Biography

Dr. Paul D. Wiedemeier is an Assistant Professor of Computer Science at The University of Louisiana at Monroe and the principle investigator of the ULM Digital Communication Research Laboratory. Dr. Wiedemeier obtained a Ph.D. in Computer Engineering and Computer Science from the University of Missouri – Columbia, a M.S. in Computer Science from Michigan Technological University, and a B.S. in Computer Science from Drake University. He is a member of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, the American Radio Relay League, the Tucson Amateur Packet Radio Corporation, the Consortium for Computing Sciences in Colleges, and the Louisiana Academy of Sciences. Dr. Wiedemeier currently holds a General Amateur Radio License (KE5LKY) issued by the United States of America Federal Communications Commission.

Appendix

Table A1: No (i.e. “None”) Authentication Data Transmission Time.

4 KB File Size			8 KB File Size			16 KB File Size		
Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds
2	43.9	163.9	4	43.5	283.5	7	49.7	469.7
2	27.2	147.2	4	29.9	269.9	7	56.1	476.1
2	22.4	142.4	4	25.1	265.1	7	39.7	459.7
2	25.3	145.3	3	58.3	238.3	7	39.1	459.1
2	32.5	152.5	4	48.2	288.2	8	36.0	516.0
2	32.6	152.6	4	30.5	270.5	7	30.0	450.0
2	29.4	149.4	4	44.1	284.1	7	35.2	455.2
2	32.5	152.5	4	4.0	244.0	7	38.8	458.8
2	41.2	161.2	4	8.1	248.1	8	46.8	526.8
2	1.8	121.8	4	7.8	247.8	7	57.6	477.6
1	59.5	119.5	3	49.2	229.2	7	21.4	441.4
2	11.5	131.5	4	29.1	269.1	8	2.5	482.5
1	57.9	117.9	4	3.2	243.2	8	6.9	486.9
2	16.0	136.0	4	23.5	263.5	9	4.8	544.8
2	21.2	141.2	3	55.1	235.1	7	21.0	441.0
2	7.2	127.2	4	4.3	244.3	8	4.8	484.8
2	21.5	141.5	3	52.0	232.0	7	38.7	458.7
2	9.2	129.2	3	49.7	229.7	7	47.8	467.8
2	3.1	123.1	3	59.4	239.4	8	6.7	486.7
2	16.0	136.0	4	42.7	282.7	7	45.4	465.4
Avg. = 2	**19.6**	**139.6**	**Avg. = 4**	**15.4**	**255.4**	**Avg. = 7**	**55.5**	**475.5**

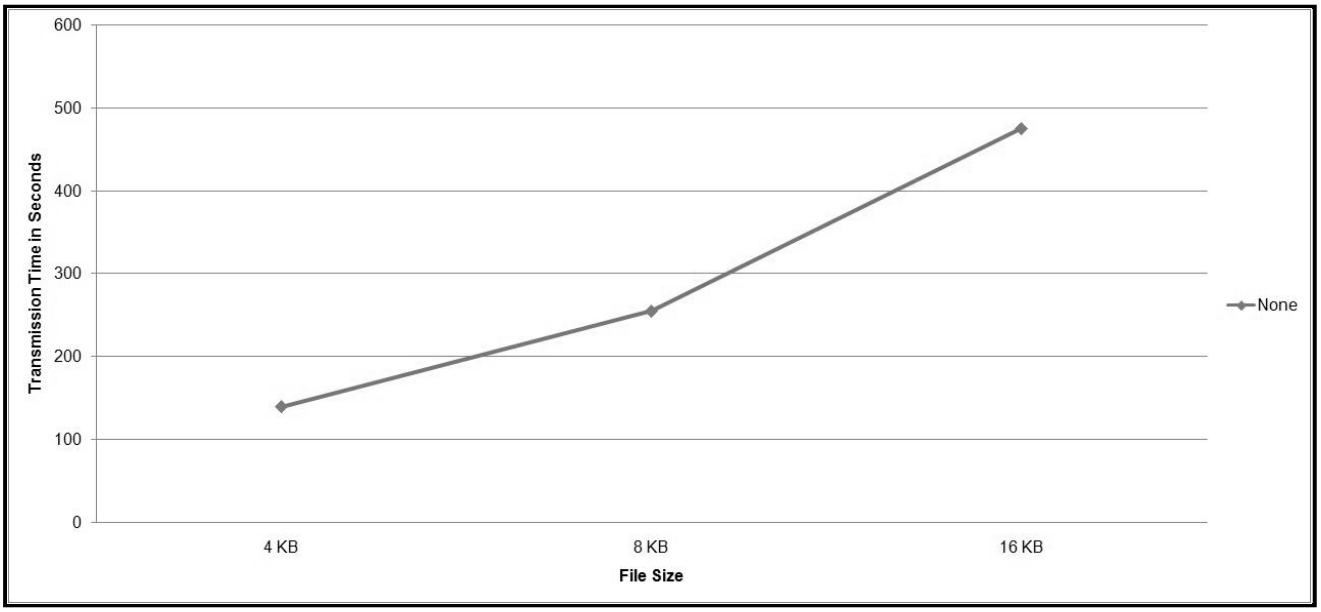

Figure A1: No (i.e. “None”) Authentication Data Transmission Time.

Table A2: GPG Authentication Data Transmission Time.

4 KB File Size			8 KB File Size			16 KB File Size		
Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds
2	45.3	165.3	4	17.1	257.1	8	20.8	500.8
3	1.5	181.5	5	17.7	317.7	8	27.3	507.3
2	45.8	165.8	4	50.8	290.8	8	3.3	483.3
2	50.6	170.6	4	28.2	268.2	8	10.2	490.2
3	2.7	182.7	4	25.8	265.8	8	26.8	506.8
2	36.2	156.2	4	21.0	261.0	8	33.6	513.6
2	34.5	154.5	4	56.3	296.3	8	0.3	480.3
2	39.9	159.9	4	35.8	275.8	8	22.1	502.1
2	42.0	162.0	4	29.5	269.5	8	33.7	513.7
2	48.1	168.1	4	23.1	263.1	8	19.1	499.1
2	27.5	147.5	4	29.5	269.5	8	35.6	515.6
2	33.8	153.8	4	36.5	276.5	9	59.3	599.3
2	33.0	153.0	4	20.3	260.3	8	21.6	501.6
2	26.3	146.3	4	26.5	266.5	9	14.6	554.6
2	35.1	155.1	4	24.0	264.0	8	57.4	537.4
2	36.0	156.0	4	12.4	252.4	7	48.9	468.9
2	20.5	140.5	4	43.6	283.6	8	4.8	484.8
2	36.4	156.4	4	29.0	269.0	8	0.9	480.9
2	36.0	156.0	4	58.9	298.9	8	0.3	480.3
2	41.9	161.9	4	11.7	251.7	8	16.9	496.9
Avg. = 2	**39.6**	**159.6**	**Avg. = 4**	**32.9**	**272.9**	**Avg. = 8**	**25.9**	**505.9**

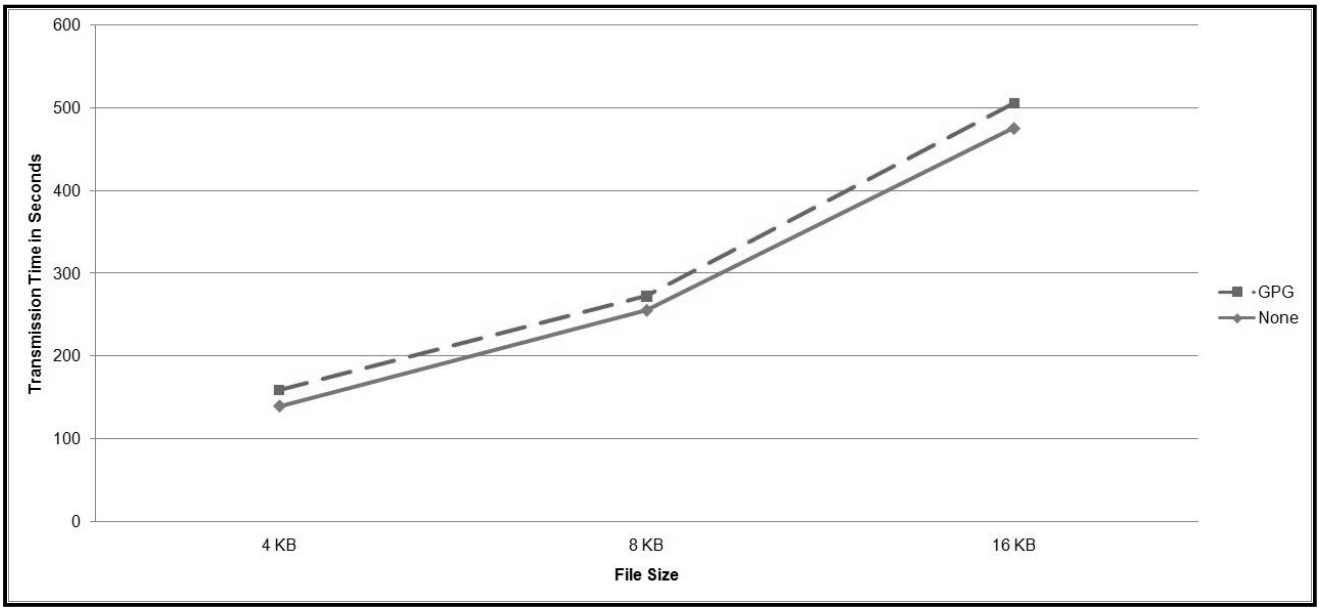

Figure A2: GPG versus No (i.e. “None”) Authentication Data Transmission Time.

Table A3: SSL/TLS Authentication Data Transmission Time.

4 KB File Size			8 KB File Size			16 KB File Size		
Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds
3	7.229	187.229	4	41.161	281.161	8	5.448	485.448
3	11.592	191.592	4	45.074	285.074	8	28.201	508.201
3	8.138	188.138	4	56.729	296.729	8	21.342	501.342
2	52.178	172.178	4	49.687	289.687	8	26.049	506.049
3	1.261	181.261	4	42.905	282.905	8	32.086	512.086
3	3.439	183.439	4	32.558	272.558	9	59.710	599.710
2	55.176	175.176	5	3.247	303.247	8	38.896	518.896
2	55.939	175.939	5	10.347	310.347	8	16.157	496.157
2	55.944	175.944	4	37.428	277.428	8	42.894	522.894
2	47.833	167.833	4	47.589	287.589	8	32.428	512.428
2	59.637	179.637	4	56.182	296.182	8	32.558	512.558
2	54.729	174.729	4	47.547	287.547	8	28.249	508.249
2	47.330	167.330	5	13.141	313.141	8	29.373	509.373
3	6.193	186.193	4	44.040	284.040	8	34.060	514.060
2	54.642	174.642	4	47.437	287.437	8	35.762	515.762
3	12.722	192.722	4	32.580	272.580	8	25.431	505.431
2	45.137	165.137	4	42.396	282.396	8	20.305	500.305
3	1.252	181.252	4	47.479	287.479	8	13.329	493.329
2	41.856	161.856	4	32.290	272.290	8	58.979	538.979
3	1.826	181.826	4	50.553	290.553	8	21.028	501.028
Avg. = 2	**58.202**	**178.202**	**Avg. = 4**	**48.018**	**288.018**	**Avg. = 8**	**33.114**	**513.114**

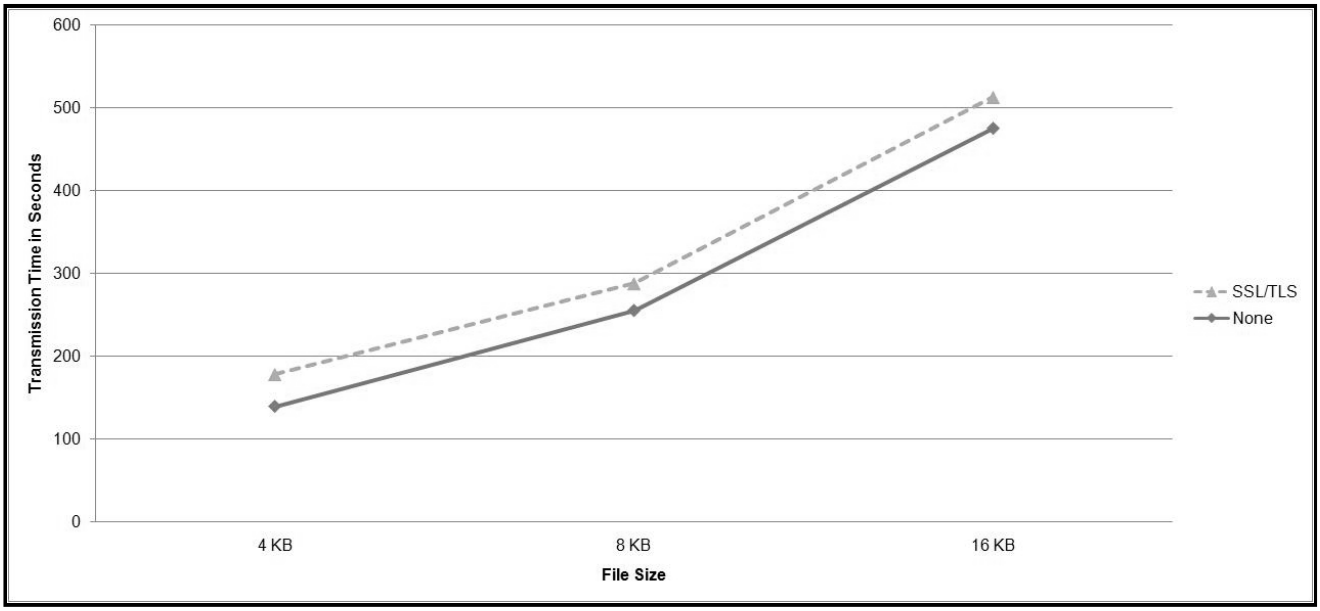

Figure A3: SSL/TLS versus No (i.e. “None”) Authentication Data Transmission Time.

Table A4: IPsec Authentication Data Transmission Time.

4 KB File Size			8 KB File Size			16 KB File Size		
Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds	Minutes	Seconds	Total Seconds
2	37.7	157.7	4	52.4	292.4	9	15.758	555.758
2	51.5	171.5	4	44.0	284.0	9	31.873	571.873
2	48.6	168.6	5	7.6	307.6	9	28.879	568.879
2	31.7	151.7	5	9.9	309.9	9	28.415	568.415
3	14.0	194.0	4	46.2	286.2	9	18.422	558.422
2	56.9	176.9	4	50.8	290.8	9	33.273	573.273
2	24.6	144.6	4	51.9	291.9	9	27.613	567.613
2	49.5	169.5	5	44.2	344.2	9	25.129	565.129
2	47.1	167.1	4	52.7	292.7	9	18.001	558.001
2	51.6	171.6	5	3.3	303.3	9	16.655	556.655
2	42.0	162.0	4	55.7	295.7	9	11.184	551.184
2	55.5	175.5	5	19.4	319.4	9	8.883	548.883
2	51.9	171.9	4	55.7	295.7	8	57.12	537.12
2	27.9	147.9	5	5.8	305.8	9	29.929	569.929
2	32.8	152.8	4	56.7	296.7	9	19.04	559.04
2	52.5	172.5	4	43.1	283.1	9	18.607	558.607
2	26.2	146.2	5	3.8	303.8	9	4.183	544.183
2	28.7	148.7	4	55.6	295.6	9	26.092	566.092
2	46.3	166.3	4	49.3	289.3	9	18.792	558.792
2	40.6	160.6	4	57.4	297.4	9	12.338	552.338
Avg. = 2	**43.9**	**163.9**	**Avg. = 4**	**59.3**	**299.3**	**Avg. = 9**	**19.5093**	**559.5093**

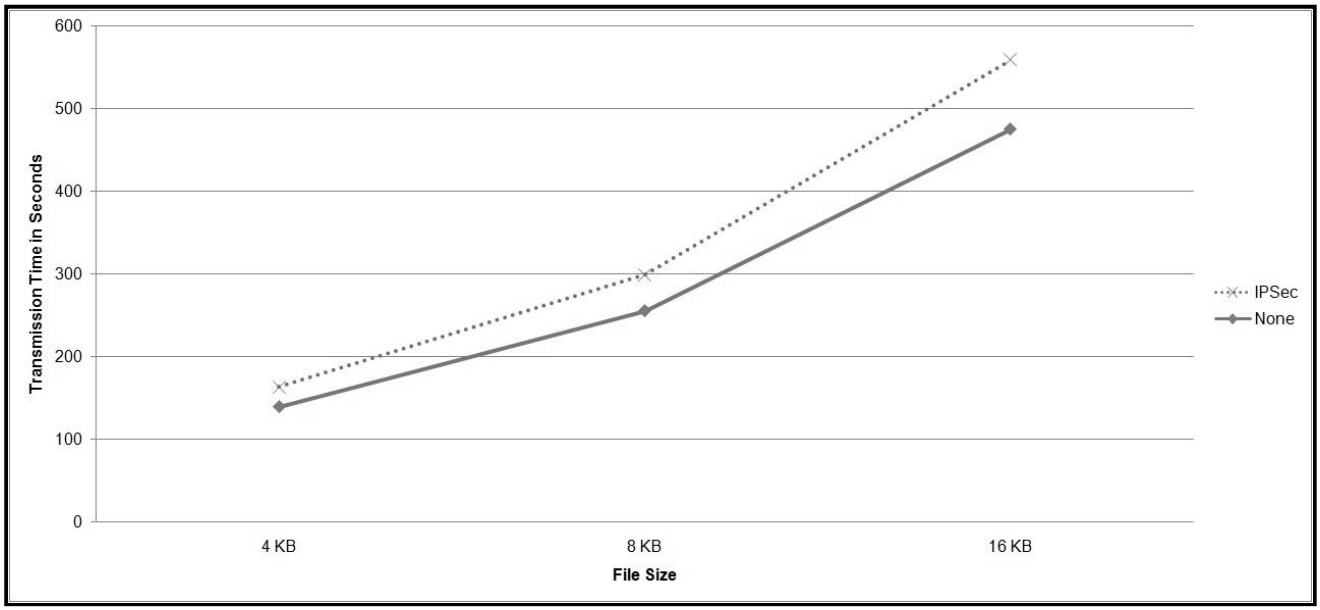

Figure A4: IPsec versus No (i.e. "None") Authentication Data Transmission Time.