A Bare PC TLS Webmail Server

Patrick Appiah-Kubi
Department of Computer & Information Sciences
Towson University
Towson, Maryland, USA
appiahkubi@towson.edu

Ramesh K. Karne
Department of Computer & Information Sciences
Towson University
Towson, Maryland, USA
rkarne@towson.edu

Alexander L. Wijesinha
Department of Computer & Information Sciences
Towson University
Towson, Maryland, USA
awijesinha@towson.edu

***Abstract*— Bare PC systems have no operating system or kernel, and can be used for building self-supporting server applications that perform better than conventional servers. The bare PC server application contains the necessary network protocols, does its own memory allocation and task scheduling, and uses direct interfaces to the hardware. We discuss the design and implementation of a TLS Webmail server that runs on a bare PC. Novel design features of the server include intertwining the TLS, HTTP and TCP protocols to reduce inter-layer communication overhead, and using a separate TLS task per connection to improve performance. We also present initial performance measurements in a LAN environment to measure the overhead due to TLS, and the possible speed-up that can be achieved compared to conventional TLS Webmail servers. The results suggest that customized bare PC servers could be designed in the future to meet the security and performance requirements of pervasive computing environments.**

Keywords-TLS; Webmail server; bare PC; operating system; performance; security

I. INTRODUCTION

Webmail systems are used extensively today due to their ability to provide anytime anyplace access to email via a Web browser. The confidentiality and integrity of Webmail messages are ensured by using TLS [1], which is a general-purpose protocol designed to secure data transferred over a TCP connection. For pervasive devices, a mini-browser running on an optimized stack is typically used to connect to a TLS-capable exchange server. When necessary, the browser may connect to a proxy server on the Internet to download compressed versions of Web pages to save bandwidth and reduce download time. In such environments, a Webmail server running a scaled-down version of TLS that retains its security strengths can be used as the exchange server.

We describe a lean implementation of TLS that is integrated within a self-supporting bare machine (or bare PC) Webmail server application. The application runs directly over the hardware with no operating system (OS) or kernel in the machine [2]. The server minimizes both system and protocol overhead including that of TLS, TCP and HTTP, and serves trimmed-down Web pages that are suited for browsers running on low-power client devices with small screen displays. Our experimental results conducted in a local environment with conventional browsers indicate that the server could be used to exchange email secured by TLS with significantly less overhead than a conventional OS-based Webmail server. Specifically, we conduct experiments in a LAN environment to compare the performance of 1) the bare PC and OS-based TLS Webmail servers; and 2) the bare PC TLS and non-TLS Webmail servers in order to determine the overhead due to adding TLS.

A conventional server provides its services with the support of an operating system (OS) or kernel, and a standard TCP/IP protocol stack. Conventional Webmail servers typically use TLS via HTTPS [3] to provide security when sending and receiving email messages. In such servers, the addition or modification of a TLS module is relatively straightforward.

In contrast, a self-supporting bare PC server contains lean versions of the necessary protocols, manages memory, schedules tasks on the CPU, and directly accesses the underlying hardware [4]-[7]. The server application runs on the hardware as a single monolithic executable. Furthermore, the application layer and transport layer protocol code have to be intertwined [4], [6], [7] with the code for the server application. The design and performance of a bare PC Webmail server are described in [4]. However, the server does not support TLS. In this paper, we describe the addition of TLS to an adaptation of the existing bare PC Webmail server.

Since they have no OS or kernel, bare PC servers are immune to attacks that target the underlying OS. They also have the following characteristics that are useful from a security viewpoint: 1) they are less complex than conventional servers and have a small code size; 2) there is no socket interface for applications; and 3) the intertwined parts of the code and the underlying task structure can be completely different for different servers.

The rest of this paper is organized as follows. Section 2 describes the architecture of a TLS bare PC Webmail server, including its design and implementation. Section 3 presents the performance measurements in a LAN environment. Section 4 provides a brief overview of related work, and Section 5 contains the conclusion.

II. SERVER ARCHITECTURE

The bare PC TLS Webmail server architecture is based on the bare machine computing paradigm [1]. The server is built by extracting the TLS code from the bare PC Web server [8] and integrating it with the code for the existing (non-TLS) Webmail server.

A. *Design*

Figure 1 compares the respective client/server message exchanges for non-TLS and TLS Webmail servers with protocol intertwining. In both cases, a TCP handshake is done as usual for connection establishment. In the case of the non-TLS Web server, the client next sends HTTP GET and POST commands to be processed. A TLS server requires an additional TLS handshake for negotiating security parameters and setting up a master key. The TLS module is responsible for this handshake and for encryption/decryption of subsequent messages including the HTTP GET and POST commands. Processing on a TLS server involves several phases [1]: the TCP handshake phase; the TLS handshake phase; the data phase during which encrypted and authenticated HTTP data and alert (TLS close-notify) messages are sent; and the TCP connection closing phase.

As shown in Figure 2, the TCP, TLS and HTTP protocols are intertwined within the Webmail server application. The CPU tasks and task scheduler are also an integral part of bare PC server design. All bare PC systems have a Main task that runs whenever other tasks are not running, and a Rcv task that handles incoming packets [4], [6], [7]. The bare PC Webmail server also has a separate TLS task that handles TLS processing as well as the HTTP POST and GET processing. Use of a single TLS task per connection also simplifies processing in case HTTP GET/POST commands come in over a long period of time when the HTTP KEEP_ALIVE option is used. The application manages and schedules the TLS tasks enabling the server to process requests concurrently. Each message request and its state information are stored in a TCP Control Block (TCB) table. In addition, parameters used by re-entrant code are also stored in this table. The simultaneous sharing of resources is avoided by allocating resources independently for each request and maintaining the necessary state information in the TCB table.

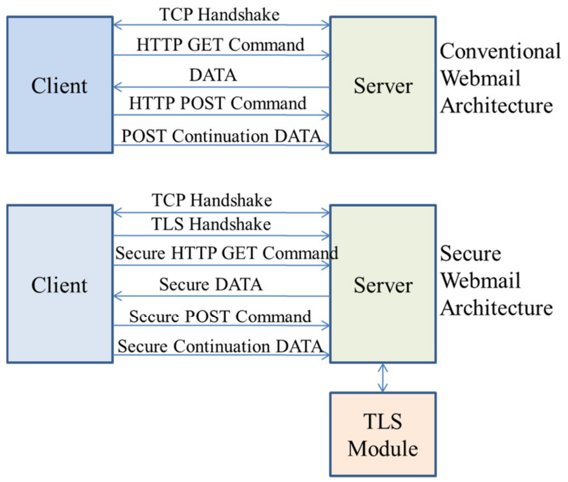


Figure 1. Non-TLS/TLS Webmail messages

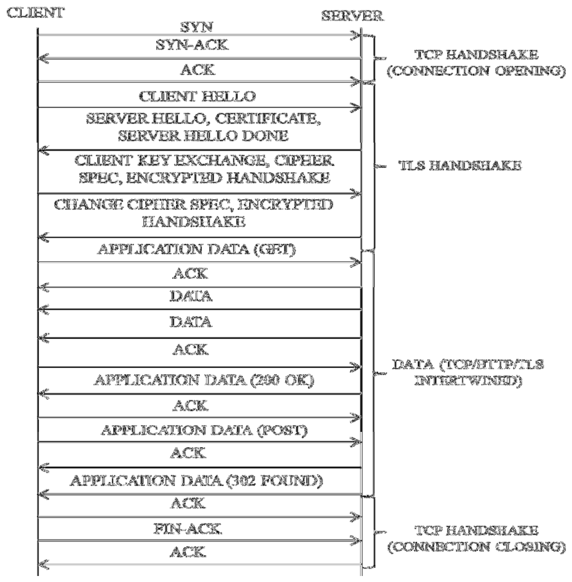

CLIENT
SERVER
SYN
SYN-ACK
ACK
CLIENT HELLO
SERVER HELLO, CERTIFICATE, SERVER HELLO DONE
CLIENT KEY EXCHANGE, CIPHER SPEC, ENCRYPTED HANDSHAKE
CHANGE CIPHER SPEC, ENCRYPTED HANDSHAKE
APPLICATION DATA (GET)
ACK
DATA
DATA
ACK
APPLICATION DATA (200 OK)
ACK
APPLICATION DATA (POST)
ACK
APPLICATION DATA (302 FOUND)
ACK
FIN-ACK
ACK
TCP HANDSHAKE (CONNECTION OPENING)
TLS HANDSHAKE
DATA (TCP/HTTP/TLS INTERTWINED)
TCP HANDSHAKE (CONNECTION CLOSING)

Figure 2. TLS/TCP/HTTP protocol intertwining

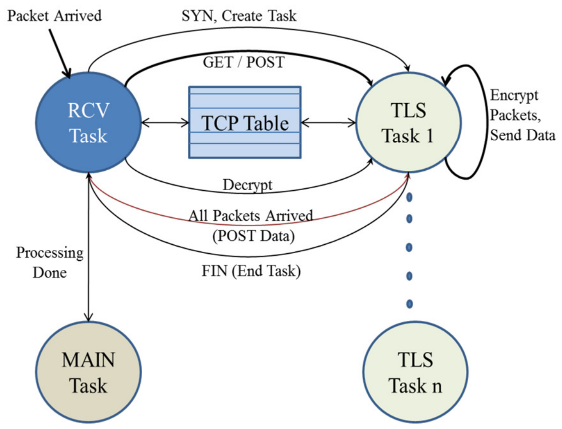
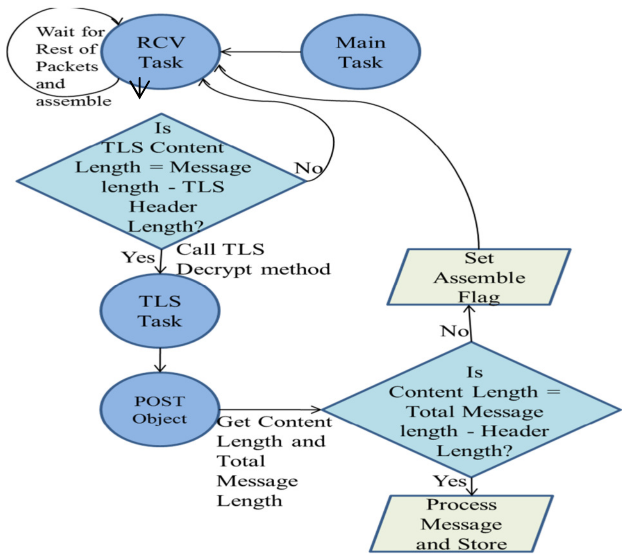


Figure 3. Task Interactions

The TCB information is also used in task scheduling, using a programmer-designed task schedule. A task executes as a single thread of execution, thus eliminating the need for a centralized scheduler. Whenever a TLS or HTTP task needs to wait for an event, it is suspended until the relevant event occurs. A TLS task is initially created when the TCP SYN segment is received. After the TLS handshake is complete, client GET/POST requests and associated data are decrypted or encrypted under the control of the Rcv and TLS tasks. These events and the associated task interactions are shown in Figure 3. For a POST command, the encrypted HTTP POST data arrives in one or more TLS fragments. The TLS CONTENT_LENGTH is used to check whether all fragments have arrived so that decryption can be performed. After decryption, the HTTP content length in the HTTP header is used to determine if all the HTTP data has arrived. If the HTTP data is fragmented, the assemble flag is set and the server waits for the remaining packets. Figure 4 illustrates the logic for handling TLS and HTTP fragments.

In the bare PC Webmail server, all memory required by encryption or hash algorithms is pre-allocated at compile-time, and only RSA key exchange, AES/CBC cipher and SHA MAC are supported. The RSA certificates and keys are pre-generated outside the server (on Linux), converted from ASCII into Hex and transferred into the server's memory. The user interface is simplified by designing static PHP pages that are pre-parsed and indexed for efficient processing.



Wait for Rest of Packets and assemble
RCV Task
Main Task
Is TLS Content Length = Message length - TLS Header Length?
No
Yes
Call TLS Decrypt method
TLS Task
POST Object
Get Content Length and Total Message Length
Set Assemble Flag
No
Is Content Length = Total Message length - Header Length?
Yes
Process Message and Store

Figure 4. Handling TLS and HTTP fragments

B. *Implementation*

The self-contained bare PC TLS Webmail server application is saved to a bootable USB flash drive. The USB includes the boot program, startup menu, application executable, and the persistent file system (used for user profiles, emails, and attachments). This USB content is generated by a tool (designed and run on MS-Windows) that creates the bootable bare PC server application. The tool, which generates the boot load sector and copies the executable and associated files to the USB, consists of 500 lines of C++ code.

The server application is developed using a standard MS-Windows environment, Visual C++, and the MASM Assembler. However, the application does not use any OS-related libraries or system calls. Instead, it has direct hardware interfaces designed for bare PC computing [9], most of which are implemented in assembly language using software interrupts. The size of this assembly code is approximately 2000 lines. The direct hardware interfaces include display, keyboard, timers, task management, and real/protected mode switching. The 3COM 905CX NIC driver code is written in C, except for approximately 1400 lines of assembly code. Similarly, the USB driver is written in C except for approximately 140 lines of assembly code.

The C++ program for the TLS Webmail server application consists of 58,284 lines of code including 19,485 lines of comments, and 29,380 executable statements. The size of the TLS module is 15,645 lines. The single monolithic executable occupies 512 sectors (389 KB).

The server application was functionally tested by using the Firefox and Internet Explorer browsers. Packets sent by the client and the server were captured using Wireshark. These packets were then analyzed and their contents were validated against the packet contents of a conventional server to ensure correctness of the server implementation.

The bare PC TLS Webmail server and other bare PC servers do not use a local disk (they only require detachable mass storage). The server application directly communicates with the hardware (in this case an X86-based CPU). This approach can also be used to build pervasive devices, gateways, routers, or sensors that host a small efficient bare machine TLS Webmail server.

III. Performance

We first conducted experiments to compare the performance of the bare PC Webmail server with OS-based Webmail servers. Then we compared the performance of the bare PC non-TLS and TLS Webmail servers to determine the overhead due to TLS.

A. *Experimental Setup*

For the experiments, a 100-Mbps Ethernet test LAN with servers and a client running on ordinary Dell Optiplex GX520 PCs (3.2GHz Intel Pentium 4 Processor with 1GB RAM) was set up. In addition to the TLS and non-TLS bare PC Webmail servers, the OS-based TLS Webmail servers were Atmail [10] on Linux, and Icewarp [11] on Windows XP. The client was a Firefox browser on Windows XP.

B. *Results*

Figures 5 and 6 respectively show the processing time for a single Get and Post (Application Data) request sent to the bare PC and Atmail servers (processing times for the Windows server are omitted since they were much larger). In Figure 5, the processing time for the Atmail server spikes between the client's Application data (GET) request and the server's ACK. The processing time for the bare PC Webmail server does not spike and is approximately linear. In Figure 6, the processing time for the POST request is shown. The processing time for the Atmail server now spikes between server's Application Data (302 found) and ACK response, but that for the bare PC Webmail server shows only a small increase.

Figure 7 shows the total time (in milliseconds) required when composing an email message with message sizes varying from 1KB to 16 KB. The processing time for the bare PC Webmail server is small and approximately the same for all message sizes. In contrast, the processing times for the Atmail (Linux) and Icewarp (Windows) servers are larger with a maximum for 6KB messages.

Figure 8 shows the total processing time when reading email messages with message sizes varying from 1KB to 16KB. The processing time in this case includes the time to retrieve a message from the user inbox and display it on the screen. The processing time for the bare PC Webmail server never exceeds 50 ms, whereas those for the OS-based servers can exceed 200 ms. Figure 9 shows the total time each server spent to process an inbox request that involves 10 email messages. The bare PC Webmail server processes the request in a total time of 42 milliseconds; this is approximately 7 times faster than Icewarp, and approximately 6 times faster than Atmail.

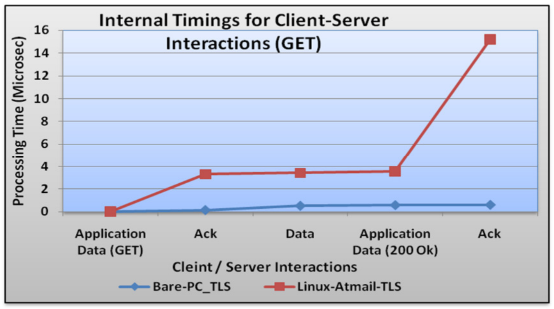


Figure 5. GET request

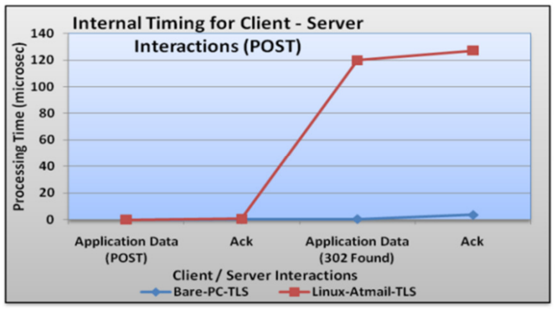


Figure 6. POST request

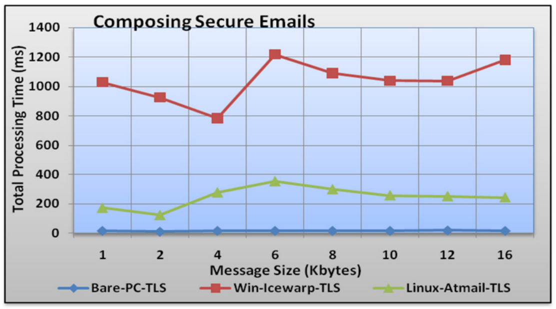


Figure 7. Compose

Figure 10 shows the total processing time when composing email messages. TLS increases the processing time by at least 5 ms for most message sizes and doubles the processing time for message sizes that are 10 KB or larger. More studies are needed to explain why the processing time for the non-TLS server is slightly larger for 6-KB messages.

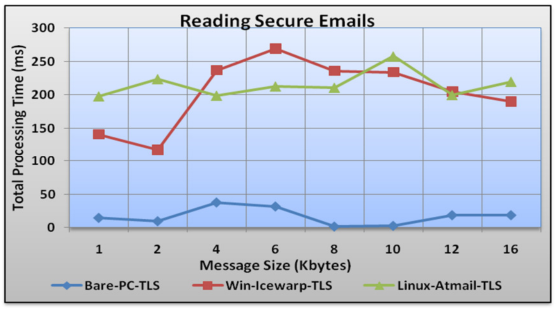


Figure 8. Message retrieval

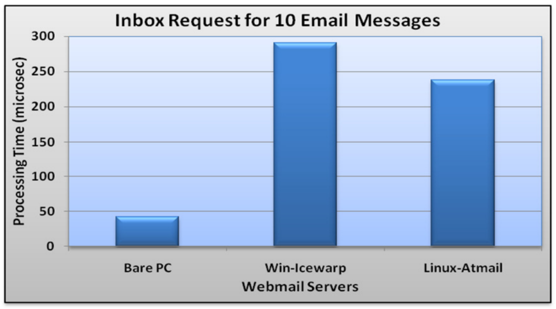


Figure 9. Inbox request for 10 email messages

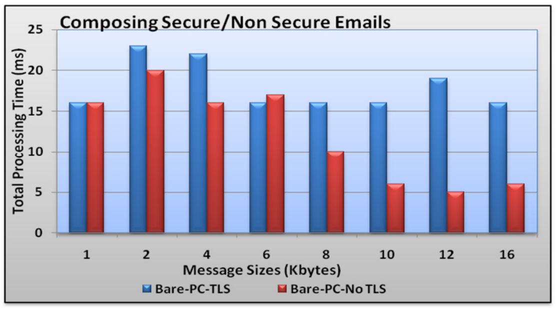


Figure 10. Compose

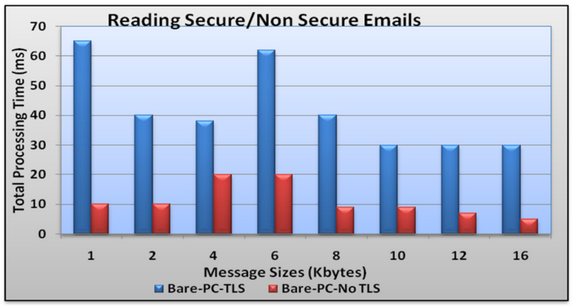


Figure 11. Message retrieval

Figure 11 shows the processing time for message retrieval. There is at least a three-fold increase in processing time due to TLS for most message sizes. Figure 12 shows the processing times when sending email messages with an attachment. The increase in processing time due to TLS is smaller for message sizes that are 8 KB or larger. Figure 13 shows that the processing time with TLS when retrieving email messages with an attachment is always larger.

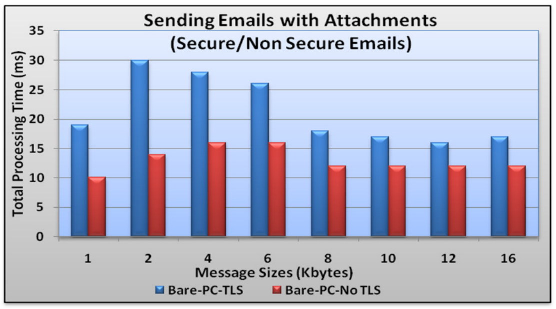


Figure 12. Sending attachments

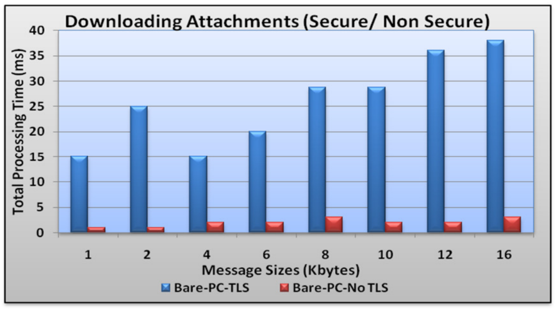


Figure 13. Retrieving attachments

IV. Related Work

Many Webmail servers use HTTPS/TLS to protect email messages in transit. There are alternate approaches to email security. S/MIME [12] provides encryption, authentication, message integrity and non-repudiation for MIME messages exchanged between users (i.e., end-to-end). The design and implementation of a secure email system that provides encryption and signing, and additional features such as elimination of spam and prevention of harmful attachments is described in [13]. The implementation of a secure Webmail system that uses CallerID for access is discussed in [14], and the specification of a Web-based system for secure transmission of email messages is given in [15].

V. Conclusion

We described the implementation of a TLS bare PC Webmail server and compared its performance in a LAN environment with Linux and Windows servers. Although the bare PC server outperforms the OS-based servers, the TLS protocol overhead on all systems is found to be non-negligible. The performance advantages of the bare PC server are due to the absence of an OS, the minimization of context switching overhead, and the ability to intertwine lean versions of the necessary protocols. These results indicate that bare PC server applications could be designed in the future to meet the security requirements as well as the bandwidth, power and performance limitations of pervasive client devices.

References

[1] T. Dierks, and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[2] R. K. Karne, K. V. Jaganathan, N. Rosa, and T. Ahmed, "DOSC: Dispersed Operating System Computing", OOPSLA '05, 20th Annual ACM Conference on Object Oriented Programming, Systems, Languages, and Applications, Onward Track, ACM, San Diego, CA, October 2005, pp. 55-62.

[3] D. Schatzmann, W. Muhlbauer, T. Spyropoulos, and X. Dimitropoulus, "Digging into HTTPS: flow-based classification of Webmail traffic", Proceedings of the 10th Annual Conference on Internet Measurement (IMC), 2010, pp. 322-327.

[4] P. Appiah-Kubi, R. K. Karne, and A. L. Wijesinha, "The Design and Performance of a bare PC Webmail Server", 12th IEEE International Symposium of Advances on High Performance Computing and Networking (AHPCN) 2010, pp. 521-526.

[5] G. Ford, R. Karne, A. L. Wijesinha, and P. Appiah-Kubi, The Performance of a Bare Machine Email Server", 21st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2009, pp. 143-150.

[6] L. He, R. Karne, and A. Wijesinha, "The Design and Performance of a Bare PC Web Server", International Journal of Computers and Their Applications, vol. 15, June 2008, pp. 100 - 112.

[7] G. Ford, R. K. Karne, A. L. Wijesinha, and P. Appiah-Kubi, "The Design and Implementation of a Bare PC Email Server", 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC), 2009, pp. 480-485.

[8] A. Emdadi, R. K. Karne, and A. L. Wijesinha, "Implementing the TLS Protocol on a Bare PC", 2nd International Conference on Computer Research and Development (ICCRD), May 2010, pp. 293-297.

[9] R. K. Karne, K. V. Jaganathan, and T. Ahmed, "How to run C++ Applications on a bare PC", SNPD 2005, Proceedings of SNPD 2005, 6th ACIS International Conference, IEEE, May 2005, pp. 50-55.

[10] Atmail-Linux Webmail Server, http://atmail.com/

[11] IceWarp Mail Server, http://www.icewarp.com/

[12] B. Ramsdell and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[13] A. Ghafoor, S. Muftic, and G. Schmolzer, "CryptoNET: Design and implementation of the Secure Email System", Proceedings of the 1st International Workshop on Security and Communication Networks (IWSCN), 2009, pp. 1 – 6.

[14] A. A. Tomar, S. Chaudhari, J. Patil, and A. Rawat, "Implementation and Security Analysis of a CallerId Augmented 2FA Setup for Secure Web-mail Access", Proceedings of the 2010 IEEE International Conference on Advances in Communication Network and Computing (CNC), 2010, pp. 346-348.

[15] S. Kaushik, D. Wijesekera, and P. Amman, "BPEL Orchestration of Secure Webmail", Proceedings of the 3rd ACM workshop on Secure Web Services (SWS), 2006, pp. 85–94..