International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 9, Issue.9, September 2021 E-ISSN: 2347-2693

Analysis of Cryptographic Libraries(SSL/TLS)

Suresh Prasad Kannojia[1*], Jitendra Kurmi[2]

[1,2]Department of Computer Science, University of Lucknow, Lucknow, India

*Corresponding Author: spkannojia@gmail.com, Mobile: +91-8840880224

DOI: https://doi.org/10.26438/ijcse/v9i9.5962 | Available online at: www.ijcseonline.org

Received: 30/Aug/2021, Accepted: 20/Sept/2021, Published: 30/Sept/2021

***Abstract*—** Secure communication in Computer Network is very important which can be achieved by Transport Layer Security (TLS) protocol. Various libraries have been created for the implementation of TLS functions by the researchers, of which each has wide support of the encryption algorithms, key exchange mechanism from which one can implement TLS for secure communications. In this paper, to find the best suitable SSL/TLS library, relative analysis of the six widely used libraries has been done based on various affecting parameter such Languages, Cryptographic Token Interface - PKCS#11, Thread Safety, and CPU Assisted Cryptography with AES-NI. Any organization can use an effective and efficient library that will provide the appropriate security and fulfill the expectation of the application.

***Keywords*—**Thread Safety, TLS, AES-NI

I. INTRODUCTION

Security in communication is essential to facilitate reliability, data integrity, and confidentiality, Transport Layer Security (TLS) can provide these aspects for secure connection over computer networks. It can be used in Email, VOIP (Voice over Internet Protocol), Web browsing, Bank transactions for the prevention of eavesdropping and tampering of data. Various versions of TLS have been developed as per requirement of secure communication such as SSL1.0, SSL2.0, SSL3.0, TLS1.0, TLS1.1, TLS1.2, TLS1.3. Each cipher suite contains authentication, message authentication code (MAC), key exchange, and encryption algorithms.

As per the report of Internet Engineering Task Force (IETF) for secure communication use of cipher suite below TLS 1.0 are not useful, as those are less secure and most of the browsers provide warning if some site is having old version. In the case of connectionless application, Datagram Transport Layer Security (DTLS) is used and it is similar to TLS except that for DTLS, it has to solve problems of packet loss and reordering. Mainly DTLS allows three features as

1) Packet Retransmission: Lost packets retransmission mechanism.
2) Sequencing for Packets: Assign sequence number to datagram for reordering and packet loss.
3) Replay detection: Used to avoid duplicate packets and discarding old received packets.

In this paper, we have taken six libraries such as OpenSSL, GnuTLS, BoringSSL, AWS s2n TLS, NSS, Cryptlib for relative analysis purpose and to find the best suited libraries for secure communication.

Organization of the paper is as follows- Section II provides chronicles of the related work of researchers, proposed methodology for relative analysis as per parameters has been given in Section III, Section IV provides the results and analysis as per parameters, and Section V presents the conclusion and future scope.

II. RELATED WORK

Various TLS libraries supported for connection oriented and connectionless application. To solve the issues of packet loss and reordering DTLS is used [1]. The Network Security Services (NSS) is an open-source TLS library designed to support cross-platform server-side and hardware smart cards on the client-side. NSS is developed by Netscape in 1997[2]. OpenSSL is an Open-Source TLS library used for secure communication over the computer network founded in Dec 1998 by Eric Andrew Young and Tim Hudson[3]. The Gnu TLS is a TLS library that allows client applications to start a secure session over the computer network using available protocols developed in March 2003 by Nikos Mavrogiannopoulos[4]. Whereas Cryptlib is an open-source security toolkit library that supports various cryptographic libraries to implement the secure sessions in SSL/TLS, developed by Peter Gutmann in Dec 2003[5].

Boring SSL is designed and developed in June 2014 by Google to meet Google's needs and supports various cipher suite algorithms for secure communication[6]. AWS s2n (Signal 2 Noise) is also an open-source TLS library developed by Amazon Web Services (AWS) and supports various cryptographic algorithms to implement SSL/TLS [7].

EverCrypt is a C/x86 cryptography library developed by the Everest project [8-11]. Recent findings are outstanding, with performance similar to a well-optimized OpenSSL program. EverCrypt, on the other hand, follows a different concept, where the library and proof are co-designed, and in some situations, code is synthesized. AWS-LC, BoringSSL, and OpenSSL are examples of handwritten libraries that could be replaced by such libraries in the future, according to this approach.

Researcher verifies SHA-256 in the CASM project. The CASM toolchain uses symbolic execution as well as SMT solvers. While SHA-256 as a whole can be analyzed, CASM only looks at functions over message blocks. Not even the most optimized variants of this algorithm are checked by CASM[12]. However, Fiat Crypto does not apply to the algorithms proven in this study [13]. A high-level specification is used to build portable C field arithmetic implementations, which are then used to generate the implementations. Fiat Crypto's code has already been incorporated into OpenSSL[14]. Similarly, Jasmin is an important synthesis method vectorized in x86 implementations with good performance are generated by it [15]. When compared to other hand-optimized implementations, the Jasmin implementation of ChaCha20-Poly1305 performs far better. SHA-256 and AES-GCM have not been implemented in Jasmin.

Therefore, it is clear that various TLS Libraries are available to provide the secure communication. It motivates me to do the relative analysis of these TLS libraries based on supported parameters and find better one as per requirement.

III. METHODOLOGY

Various TLS libraries are exist but it is very challenging to choose one library out of them as per requirement. Because each libraries has their own advantage and disadvantages To find the suitable TLS libraries, comparisons of the six widely used libraries such as OpenSSL, GnuTLS, BoringSSL, AWS s2n TLS, NSS, Cryptlib has been taken based on various affecting parameter like supported Languages, Cryptographic Token Interface-PKCS#11, Thread Safety, CPU Assisted Cryptography with AES-NI have been shown in fig. 1.

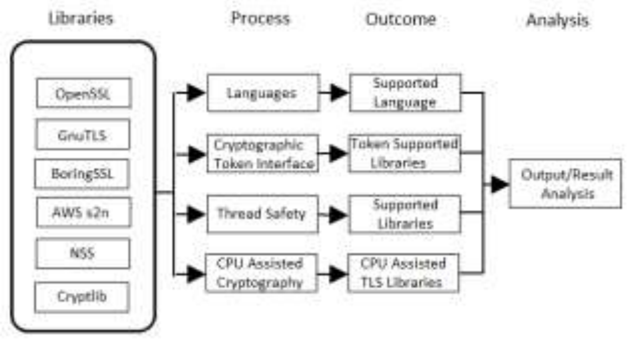

Figure 1. Proposed Methodology for relative analysis of different SSL/TLS libraries with parameters as supported Languages, Cryptographic Token Interface, Thread Safety, CPU Assisted Cryptography.

IV. RESULTS AND ANALYSIS

The comparison of six TLS libraries criteria includes, Languages, Cryptographic Token, Thread safety, and CPU assisted Cryptography. Each type libraries provides a unique specification for the libraries and selection criteria have been tabulated in Table 1.

Based on Languages supported: These six libraries are written in C language, whereas NSS and Boring SSL having extra support for C++ language, that is tabulated in Table 1. It is clear that one of the unique parts of NSS is having cross-platform support. The same library has modules and functions that can be used directly with UNIX-based or Windows machines. As for others, some wrapper package of patches needs to be added.

Table 1. TLS libraries and Support/Languages

Support/ Language	**OpenSSL**	**Gnu TLS**	**Boring SSL**	**AWS s2n**	**NSS**	**Cryptlib**
Languages	C	C	C or C++, Go, Assembly	C	C or C++	C
Cryptographic Token Interface-PKCS#11	Not Present natively	Present	Not Present	Not Present	Present	Not Present
Thread Safety	Two callback functions with POSIX or win Threads	Use POSIX or Win Thread	Yes	No	Yes	Use POSIX send() and recv(). API to supply your replacement.
CPU Assisted Cryptography with AES-NI	Yes	Yes (+VIA Padlocks)	Yes	Yes	Yes	Yes (+VIA Padlocks)

Based on Cryptographic Token Interface: It's provides a programming interface to create and manipulate cryptographic tokens. It has a Platform-Independent Application Programming Interface (API) for Hardware Security Modules (HSM) and Smart Cards. This API has been included in Public Key Cryptographic Standard #11, commonly known as PKCS #11, "Cryptoki".

From table 1 we can notice that GnuTLS and NSS have support for PKCS #11 natively, but it's not the case with OpenSSL. To use the API in OpenSSL, the API engine needs to be added externally through a patch. Now selecting a correct engine as per the requirement will be essential for successful token implementations. So in this criterion, selecting native implementation of PKCS #11, which is present in NSS or GnuTLS, will be uncomplicated and effortless. The library Boring SSL, AWS s2n, and Cryptlib does not support a cryptographic token interface.

Based on Thread Safety: It's ensures safe handling of shared data structures, by which all the threads behave correctly without violating their specification. From table 1 it's clear that, even if OpenSSL supports thread safety with POSIX or Windows Threads, earlier versions than 1.1.0, it can safely be used in multi-threaded applications provided that at least two callback functions are set, locking function and threaded function and for later versions, with some limitations. An SSL connection cannot be used concurrently by multiple threads. GnuTLS, Boring SSL, NSS, and Cryptlib also provide thread safety with the use of POSIX or windows thread. Whereas GnuTLS utilizes mutual exclusion locks for the data structure protection with Random number generator locks are set up by GnuTLS on library initialization, but AWS s2n does not have the thread safety support.

Based on CPU Assisted Cryptography. This mechanism uses hardware acceleration for cryptographic functions with a supported instruction set. By using accelerators, intensive cryptographic operations can be run more efficiently than running on a general-purpose CPU.

Advanced Encryption Standard New Instructions (AES-NI) is an extension for x86 instructions set architecture, used for Intel or AMD processor. It can be used to improve the speed of applications performing encryption and decryption using the Advanced Encryption Standard (AES). From table 1 it is clear that four libraries such as OpenSSL, BoringSSL, AWS s2n, and NSS can take advantage of this feature. But the GnuTLS and Cryptlib have additional support for VIA PadLock Security Engine in VIA x86 processors. It uses a different instruction set than AES-NI for AES acceleration. So GnuTLS can provide support for systems with VIA processors.

V. CONCLUSION AND FUTURE SCOPE

In this paper, comparative analysis of the six TLS libraries such as OpenSSL, GnuTLS, BoringSSL, AWS s2n TLS, NSS, Cryptlib has been done to find the best library for TLS based on various affecting parameter of libraries such Languages support, Cryptographic Token Interface-PKCS#11, Thread Safety, and CPU Assisted Cryptography with AES-NI. After analysis it is found that GnuTLS, NSS are best libraries that support all affecting parameters, whereas CPU Assisted Cryptography with AES-NI support is found in all these six libraries. In future comparison of TLS libraries can done on the basis of execution time and CPU usage to find the best TLS library on different machine.

REFERENCES

[1] E. Rescorla, and N. Modadugu. "*Datagram transport layer security version 1.2.*" **2012.**

[2] Mozilla Developer Network, "*Network Security Services*", Aug 10, 2021.https://developer.mozilla.org/enUS/docs/Mozilla/Projects/NSS#Documentation.

[3] OpenSSL, Cryptography and SSL/TLS Toolkit - Threads, 1.0.2 manpages, 10 August 2021 https://www.openssl.org/docs/man1.0.2/crypto/threads.html

[4] GnuTLS, Transport Layer Security Library for the GNU system, for version 3.7.1, March 2021. https://www.gnutls.org/manual/gnutls.html

[5] Gutmann, Peter, "Downloading", cryptlib, University of Auckland School of Computer Science, 07 July 2021

[6] Google. "BoringSSL." Google, **29 July 2021.** https://boringssl.googlesource.com/boringssl/

[7] A. Chudnov, N. Collins, B. Cook, J. Dodds, B. Huffman, C. MacCárthaigh, S. Magill, "*Continuous formal verification of Amazon s2n*", In International Conference on Computer Aided Verification, pp. 430-446, Oxford, UK, **2018.**

[8] B. Bond, C. Hawblitzel, M. Kapritsos, K. R. M. Leino, J. R. Lorch, B. Parno, A. Rane, S. Setty, and L. Thompson, "*Vale: Verifying high-performance cryptographic assembly code*", In 26th {USENIX} Security Symposium ({USENIX} Security 17), pp. 917-934, VANCOUVER, BC, CANADA **2017.**

[9] Fromherz, Aymeric, N. Giannarakis, C. Hawblitzel, B. Parno, A. Rastogi, and N. Swamy, "*A verified, efficient embedding of a verifiable assembly language*", Proceedings of the ACM on Programming Languages 3, pp. 1-30, New York, United States, 2019

[10]T. Bingmann, "*Speedtest and comparsion of open-source cryptography libraries and compiler flags*", Timo Bingmann– 2008. https://panthema. net/2008/0714-cryptography-speedtest-comparison (2008), **2018.**

[11]J.K Zinzindohoué, K. Bhargavan, J. Protzenko, and B. Beurdouche, "*HACL*: A verified modern cryptographic library*", In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1789-1806,New York, United States 2017.

[12]J. P. Lim, and S. Nagarakatte, "*Automatic equivalence checking for assembly implementations of cryptography libraries*", In 2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), pp. 37-49, Washington, DC, USA, 2019.

[13]A. Erbsen, J. Philipoom, J. Gross, R. Sloan, and A. Chlipala, "*Simple high-level code for cryptographic arithmetic-with proofs, without compromises*", In 2019 IEEE Symposium on Security and Privacy (SP), pp. 1202-1219, San Francisco, CA, USA , **2019.**

[14]V. Gopal, J. Guilford, E. Ozturk, S. Gulley, W. Feghali, "*Improving OpenSSL* Performance"*, in IA Architects Intel Corporation, October 2011, https://software.intel.com/sites/default/files/open-sslperformance-paper.pdf

[15]B. Boston, S. Breese, J. Dodds, M. Dodds, B. Huffman, A. Petcher, and A. Stefanescu, "*Verified Cryptographic Code for Everybody*", In International Conference on Computer Aided Verification, pp. 645-668, Springer, Cham, **2021.**

AUTHORS PROFILE

Suresh Prasad Kannojia is working as an Assistant Professor in the Department of Computer Science, University of Lucknow, Lucknow, since 2005. He has completed his Ph.D in 2013 from the University of Lucknow, Lucknow. His current area of research interest includes pattern recognition, image security, software quality, system security, data warehousing and data mining, Soft computing. He has also organized three national conferences and one national research scholars meet. He is the author of three books (two National and one International publisher). He has published 20 research papers in national and international journal/conferences.

Jitendra Kurmi is a Research Scholar in the Department of Computer Science, University of Lucknow, Lucknow, India. He received his Master of Technology degree in Computer Science and Engineering from Lovely Professional University, Jalandhar, Punjab, India, and Bachelor of Technology degree in Computer Science and Engineering from Integral University, Lucknow, India. His main research interest focuses on Cryptography Algorithms, Network Security, Transport Layer Security, Soft Computing, and Image Processing.