IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 8, August 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.569

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 8, August 2021 ||

| DOI:10.15680/IJIRSET.2021.1008134 |

Comparative Study of SSL/TLS Cryptographic Libraries

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

Ph.D Student, Department of Computer Science, University of Lucknow, Lucknow, India[1]

Assistant Professor, Department of Computer Science, University of Lucknow, Lucknow, India[2]

ABSTRACT: Secure Communication is very challenging task in every day of life, it can be achieved by Transport Layer Security (TLS) protocols. Various libraries have been created by the researchers for the implementation of TLS functions. In this paper, comparisons of the widely used libraries such as OpenSSL, GnuTLS, BoringSSL, AWS s2n TLS, NSS, Cryptlib, based on the SSL, versions of TLS and Versions of DTLS and comparison with widely supported encryption algorithms, key exchange mechanism, hash functions and ciphers with TLS Libraries has been done. One can choose the TLS security depending on the need of TLS applications for secure communication.

KEYWORDS: Protocol**,** TLS, OpenSSL, GnuTLS, Communication.

I. INTRODUCTION

Transport Layer Security (TLS) are use to provide 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. As TLS has been evolving from SSL 1.0 over the years, some of the features like hash functions, key exchange algorithms have been changed to comply with the need to have better security. Various versions of TLS has been developed as per requirement of secure communication such as SSL1.0, SSL2.0, SSL3.0, TLS1.0, TLS1.1, TLS1.2, and 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. The support for the implementation of TLS1.3 working, which can be used, is provided by the widely used Google Chrome and Mozilla Firefox browsers. Websites like https://www.cloudflare.com/ have started using TLS 1.3 implementation.

In the case of a connectionless application, using a datagram, Datagram Transport Layer Security (DTLS) is used. DTLS is made similar to TLS except that for DTLS, it has to solve problems of packet loss and reordering [1]. DTLS implements three features:

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.

Paper is organized as follows. Section II provides the related work has been invented by various researchers. The flow diagram represents proposed methodology that is given in Section III. Section IV presents experimental results and comparative analysis. Finally, Section V presents conclusion.

II. RELATED WORK

The 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, written in C programming languages [2]. It supports various cryptographic functions for SSL/TLS. 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 [3]. 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 [4]. It is written in C, C++, Go languages. AWS s2n (Signal 2 Noise) is also

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 8, August 2021 ||

| DOI:10.15680/IJIRSET.2021.1008134 |

an open-source TLS library written in a C programming language developed by Amazon Web Services(AWS)and released in 2015 and supports various cryptographic algorithms to implement SSL/TLS [5].

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 written in C programming language [6]. 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 using C programming language [7].

Further, various researchers use AWS s2n TLS, NSS, and Cryptlib because these are the most popular TLS cryptographic libraries for formal verification. Approximately 4.66 billion internet users are global. Even though these libraries are often compromised little bit amount of code, their security and performance implications are significant. They also rarely have precise specifications and have extremely low dependency, which removes some of the main barriers to general-purpose verification. In past few years, several research have been made by researcher to verify the cryptographic library code.

Further, previous work by Galois and AWS verified the AWS s2n TLS library primitives such as HMAC and DRBG [5]. They validate production cryptographic code using Software Analysis Workbench (SAW) as a result; the primitives confirmed are more sophisticated than they are now. Since they are written using the C rather than x86, the primitives for HMAC and DRBG are essentially simpler algorithmic structures. In addition, unlike the OpenSSL-derived code we're targeting, this code was created for verification. Ye et al. additionally verified the HMAC and DRBG of OpenSSL's C versions with the foundation Verified Software Toolchain (VST) [8].

EverCrypt is a C/x86 cryptography library developed by the Everest project [9-12]. Recent findings are outstanding, with performance similar to a well-optimized OpenSSL program. EverCrypt, on the other hand, follows a different concept, where the libraries and their proofs are design jointly, 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.

EverCrypt further differs in that they uses inferences of proof methods similar to the theorem prover Coq and Isabelle. For example, proofs work with unbounded input sizes thanks to this method of reasoning. As a result, proofs tend to be longer and more verbose. Due to the proof and implementation are mixed, EverCrypt's proof size is difficult to determine, however, an earlier Formherz et.al research shows AES-GCM proof of EverCrypt uses 2,000 lines of proof libraries plus additional proof combined [10]. The implementation of AES-256-GCM takes fewer than 1000 lines of evidence in comparison to SAW, which is designed to automate reasoning whenever possible.
Researcher verifies SHA-256 in the CASM project [13]. 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. Examples include the omission of EVP and vector operations on the x86 architecture, which are two of the biggest obstacles.

However, Fiat Crypto does not apply to the algorithms proven in this study [14]. 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[15]. Similarly, Jasmin is an important synthesis method vectorized in x86 implementations with good performance are generated by it [16]. 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 none of the above TLS Library is complete forever to provide the secure communication for every environment. It does motivate me to compare the TLS library and find better one as per requirement.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 8, August 2021 ||

| DOI:10.15680/IJIRSET.2021.1008134 |

III. METHODOLOGY

The open-source libraries such as OpenSSL, GnuTLS, BoringSSL, AWS s2n, NSS, and Cryptlib are very popular SSL/TLS library. That has been taken for comparison purpose. To find the suitable TLS libraries for secure communication, we want to first compare these six different open-source libraries based on the SSL, versions of TLS and Versions of DTLS and then comparison with widely supported encryption algorithms, key exchange mechanism, hash functions and ciphers with TLS Libraries.

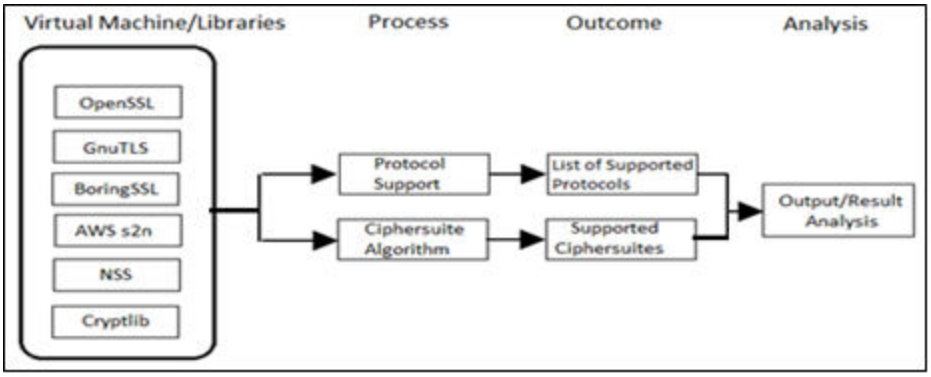

Fig. 1. Proposed Methodology for Comparison of different SSL/TLS libraries

IV. COMPARATIVE ANALYSIS

The experiment has been performed using six open-source libraries such as OpenSSL, GnuTLS, BoringSSL, AWS s2n TLS, NSS, Cryptlib. Based on the experiment, we obtained the results and compare these six TLS libraries presented in sub section 1.1, supported cipher suite algorithms present in TLS libraries in sub section 1.2.

1.1 Comparison TLS libraries with SSL,TLS versions, DTLS versions protocols

All six libraries and SSL, versions of TLS, and versions of DTLS have been tabulated in Table1. From Table 1, it is clear that the libraries OpenSSL, Boring SSL, and NSS support TLS versions till 1.3. But Gnu TLS, AWS s2n TLS, and Cryptlib do not support it. Here Boring SSL only does not support the SSL. The libraries such as OpenSSL, Gnu TLS, Boring SSL, and NSS support DTLS versions but AWS s2n TLS and Cryptlib do not support the DTLS versions. So if the datagram application is needed TLS implementation then the developer has to use libraries other than AWS s2n TLS and Cryptlib for better security.

Table 1. Comparison of different TLS libraries with SSL, TLS Versions, DTLS versions

Library	SSL	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	DTLS 1.0	DTLS 1.2
OpenSSL	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GnuTLS	Yes	Yes	Yes	Yes	No	Yes	Yes
Boring SSL	No	Yes	Yes	Yes	Yes	Yes	Yes
AWSs2nTLS	Yes	Yes	Yes	Yes	No	No	No
NSS	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Cryptlib	Yes	Yes	Yes	Yes	No	No	No

1.2 Comparison based on supported cipher suites algorithms in TLS libraries

All six libraries having various types of algorithms that facilitate Key Exchange mechanisms, Hash Functions, and Symmetric ciphers encryption services for communication have been tabulated in Table 2.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 8, August 2021 ||

| DOI:10.15680/IJIRSET.2021.1008134 |

Table 2. Supported cipher suites algorithms, TLS libraries with Key Exchange Mechanism, Hash Functions, Ciphers

	OpenSSL	GnuTLS	BoringSSL	AWS s2n	NSS	Cryptlib
Key Exchange Mechanism (Asymmetric Ciphers)	RSA, DHE-RSA, ECDHE-RSA, ECDHE-ECDSA, DHE-DSS, ECDH-ECDSA, ECDH-RSA, GOST-28147-89	RSA, DHE-RSA, DHE-DSS, ECDHE-ECDSA, ECDHE-RSA	RSA, DHE-RSA, DHE-DSS, ECDHE-ECDSA, ECDHE-RSA	RSA, DHE-RSA, ECDHE-RSA, ECDHE-ECDSA	RSA, DHE-RSA, ECDHE-RSA, ECDHE-ECDSA, DHE-DSS, ECDH-ECDSA, ECDH-RSA, GOST-28147-89	RSA, DHE-RSA, DHE-DSS, ECDHE-ECDSA
Hash Functions (Message Authentication Code)	HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384, AEAD, GOST-28147-89, GOST-R-34.11-94	HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384, AEAD	HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384	HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384	HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384	HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384
Ciphers (Symmetric Ciphers)	AES-GCM, AES-CCM, AES-CBC, Camellia-CBC, ARIA-GCM, SEED-CBC, 3DES, GOST-28147-89, ChaCha20Poly1305	AES-GCM, AES-CCM, AES-CBC, Camellia-GCM, Camellia-CBC, 3DES, ChaCha20-Poly1305	AES-GCM, AES-CBC, 3DES, Chacha20-Poly1305	AES-GCM, AES-CBC, 3DES, Chacha20-Poly1305	AES-GCM, AES-CBC, Camellia-CBC, SEED-CBC, 3DES, Chacha20-Poly1305	AES-GCM, AES-CBC, 3DES

From Table 2 it is clear that the TLS libraries OpenSSL and NSS support key exchange mechanism algorithms such as RSA, DHE-RSA, DHE-DSS, ECDH-ECDSA, ECDHE-ECDSA, ECDH-RSA, ECDHE-RSA, and GOST -28147- 89. Whereas Gnu TLS and Boring SSL support only RSA, DHE-RSA, DHE-DSS, ECDHE-ECDSA and ECDHE-RSA, but do not support ECDH-ECDSA, ECDH-RSA, and GOST-28147-89. The AWS s2n supports RSA, DHE-RSA, ECDHE-ECDSA but does not support ECDH-ECDSA, ECDH-RSA, GOST-28147-89, and DHE-DSS whereas Cryptlib support RSA, DHE-RSA, DHE-DSS, ECDHE-ECDSA but does not support ECDH-ECDSA, ECDH-RSA, GOST-28147-89.

Similarly, Hash Function Algorithms HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384 are supported by all six libraries but only OpenSSL supports additional Hash Function Algorithms such AEAD, GOST-28147-89, and GOST-R-34.11-94, except OpenSSL their algorithms are not supported by rest five libraries.
In the case of Ciphers supported by TLS libraries, OpenSSL supports AES-GCM, AES-CCM, AES CBC, Camellia CBC, ARIA GCM, SEED CBC, 3DES, GOST-28147-89, and Chacha20-Ploy1305. Whereas GnuTLS only supports AES GCM, AES CCM, AES CBC, Camellia GCM, Camellia CBC, 3DES, and Chacha20-Ploy1305.

Similarly, BorigSSL and AWS s2n supports AES GCM, AES CBC, 3DES, Chacha20-Ploy1305 but do not support Camellia CBC, Camellia GCM, ARIA GCM, and GOST-28147-89 algorithms presented in Table 2. The Library NSS only supports AES GCM, AES CBC, Camellia CBC, SEED CBC, 3DES, and Chacha20-Ploy1305 but does not support Camellia GCM, ARIA GCM, and GOST-28147-89 algorithm presented in Table 2. The Cryptlib library only supports AES GCM, AES CBC and 3DES Hash Function Algorithm presented in Table 2.
For the successful implementation of TLS, the correct combinations of ciphers should be used. Each library has a default algorithm selected which can be changed according to the need of security.

V. CONCLUSION

The work done in this paper concisely provides various differentiating factors for comparing the TLS libraries. Each has some unique features which can be used to implement cryptographic functions. At first, we saw, how various versions of the TLS implementation have been included in each of the libraries. Only TLS1.3 and DTLS support were

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 8, August 2021 ||

| DOI:10.15680/IJIRSET.2021.1008134 |

varied. Algorithm supported provides a unified view of implemented cipher and algorithms. One can choose the appropriate library or combination of library for secure communication as per requirement.

REFERENCES

[1] E. Rescorla, and N. Modadugu. "Datagram transport layer security version 1.2."), RFC 6347, Internet Engineering Task Force (IETF), January 2012, https://tools.ietf.org/html/rfc6347
[2] 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
[3] GnuTLS, Transport Layer Security Library for the GNU system, for version 3.7.1, 3 March 2021.https://www.gnutls.org/manual/gnutls.html
[4] Google. "BoringSSL." Google, 29 July 2021. https://boringssl.googlesource.com/boringssl/
[5] 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, Springer, Cham, 2018.
[6] Mozilla Developer Network, "Network Security Services", Aug 10, 2021. https://developer.mozilla.org/enUS/docs/Mozilla/Projects/NSS#Documentation.
[7] P. Gutmann, "Downloading", cryptlib, University of Auckland School of Computer Science, 07 July 2021
[8] K. Q. Ye, M. Green, N. Sanguansin, L. Beringer, A. Petcher, and A. W. Appel, "Verified correctness and security of mbedTLS HMAC-DRBG" In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 2007-2020, 2017.
[9] 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, 2017.
[10] 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, no. POPL (2019): 1-30, 2019
[11] 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.
[12] 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, 2017.
[13] 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, IEEE, 2019.
[14] 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, IEEE, 2019.
[15] 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
[16] 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.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.569

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details