TLS Cipher Suites Recommendations: A Combinatorial Coverage Measurement Approach

Dimitris E. Simos*, Kristoffer Kleine*, Artemios G. Voyiatzis*, Rick Kuhn§, Raghu Kacker§
* SBA Research, Vienna, Austria
Email: {dsimos,kkleine,avoyiatzis}@sba-research.org
§ NIST, Information Technology Laboratory, Gaithersburg, MD, USA
Email: {d.kuhn.nist.gov,raghu.kacker}@nist.gov

***Abstract*—We present a coverage measurement for TLS cipher suites recommendations provided by various regulatory and intelligence organizations such as the IETF, ENISA, the German BSI, Mozilla and the USA NSA. These cipher suites are measured and analyzed using a combinatorial approach, which was made feasible via developing the necessary input models. Besides, shedding light on the coverage achieved by the proposed recommendations we discuss implications towards aspects of test quality. One of them relates to the testing of a TLS implementation, where a system designer or tester should expand the TLS cipher suite registry and integrate the information back to the TLS implementation itself such that the (overall) testing effort is reduced.**

***Keywords*-Combinatorial testing, coverage, measurement, TLS, subsets, cipher suites.**

I. Introduction

Software implementations of the Transport Layer Security (TLS) protocol specification are a critical component for the security of the Internet communications and beyond. Software bugs and attacks, such as Heartbleed[1] and POODLE[2], still surface in implementations of the TLS protocol, despite many years of analysis (at least for open-source implementations). This can be attributed to the complexity of the protocol and its very high number of interactions.

The Handshake Protocol of the TLS 1.2 protocol specification allows a network client and a server to mutually agree on the security settings of their network connection. The named combination of the authentication, encryption, message authentication code (MAC), and key exchange algorithms to be used form a "cipher suite", as defined in the IETF RFC 5246 and its successors.

Various actors, including the IETF, ENISA, Mozilla, USA NSA, and the German BSI, have analyzed the security achieved by various combinations of cryptographic algorithms and issued recommendations regarding the support for specific cipher suite sets that are considered secure for their environment and their threat models.

System designers and integrators are faced with a challenging task: they must ensure that the TLS implementation (library) integrated in their system can handle correctly all cipher suites proposed by the other end of the communication channel and, at the same time, also conform to a desired level of security. It not enough for a TLS implementation to support a set cryptographic algorithms for various tasks (e.g., encryption or authentication) but also to ensure that only "secure" combinations of them are accepted and used.

In this paper, we perform a study of the combinatorial coverage of the TLS cipher suites. The aim of the study is to provide insights on the coverage achieved by the proposed recommendations and guide, if possible, the selection of appropriate test suites that minimize the number of combinations checked to ensure a "correct" implementation.

The rest of the paper is organized in four sections. Section II provides a short introduction on the TLS protocol, the cipher suite recommendations, and the combinatorial coverage metrics. Section III describes the modelling space problem and analyzes the metrics per recommendation. Section IV evaluates and compares the coverage measurements and the implications on software testing and quality assurance, while Section V concludes our paper and discusses future directions of work.

II. Background information

A. Transport layer security

The TLS protocol (currently at version 1.2) aims to secure network traffic at the transport layer, providing message integrity and confidentiality. The primer example being between an unauthenticated web browser (client) and an authenticated web site (server). The protocol is defined in various standards by the IETF, including among others RFCs 2246, 3546, 4346, 4366, 4680, 4492, 5246, 5288, 5746, 6176, and 6655. The origins of the TLS protocol date back in 1993 and the definition of SSL v1.0. The next major protocol revision TLS v1.3 is expected soon [1].

The protocol executes in two phases: first a handshake (or key agreement) phase and then a record layer encryption phase. In the first phase, the two communicating parties negotiate and agree on a set of cryptographic algorithms to use. Then, an authenticated key exchange cryptographic protocol is used to perform either mutual or server-to-client authentication and to establish a shared session key. In the second phase, the agreed session key is used to perform authenticated encryption.

[1] http://heartbleed.com/
[2] https://www.openssl.org/~bodo/ssl-poodle.pdf

B. Cipher suites and recommendations

The TLS protocol is quite open in the definition of cipher suites, i.e., the set of cryptographic algorithms to use for implementing security. The Internet Assigned Numbers Authority (IANA) maintains the "TLS Cipher Suite Registry", a list of registered named cipher suites for versions 1.0-1.2 of the TLS protocol [2].

IANA assigns a unique 2-byte identifier for each cipher suite. For example, the identifier {0x00, 0x3D} has a description of "TLS_RSA_WITH_AES_256_CBC_SHA256" and defines a cipher suite that uses the RSA algorithm for combined key exchange and authentication, the AES-256 algorithm in CBC mode for encryption, and the HMAC-SHA256 algorithm for MAC.

The Registry contained in early 2016 more than 300 named cipher suites, with a rich set of cryptographic algorithms for various uses. There are 28 different cryptographic algorithms (including variants) for the combined key exchange and authentication part, 25 different ones for the encryption part, and five for the MAC.

Various organizations have taken the initiative to analyze the security of the cipher suites and defined "cipher suites recommendations", i.e., sets of cipher suites that are considered strong enough to be used in the specific application scenarios.

The Mozilla Operations Security (OpSec) team recommends a set of 22 TLS cipher suites as part of its reference guide for hardened configurations of server-side implementations, including Apache, nginx, and HAProxy [3].

The European Union Agency for Network and Information Security (ENISA) commissioned a study on cryptographic protocols [4]. This study recommends a set of 24 cipher suites. The TLS protocol is part of the study and it is recommended to use version 1.2 of the protocol only and a small subset of the IANA Registry. This includes four cryptographic algorithms for the key exchange and eight for the encryption. The recommendation notes that none of the available key exchange mechanisms are particularly favorable for future (i.e. long term) use as no proof of security exists.

BSI is the Federal Agency of Germany for IT security. BSI issued and revises technical guidelines ("Technische Richtlinie") regarding the use of TLS [5]. The guidelines suggest currently the use of TLS v1.2 with 16 cipher suites.

The USA government has published guidelines for "NSA Suite B Cryptography" that define cryptographic algorithm policy for national security applications [6]. The IETF RFC 6460 defines a TLS v1.2 profile that is fully compliant with NSA Suite B. Just two cipher suites are considered for full compliance.

C. TLS software testing

The TLS protocol has evolved over many years with a non-systematic development process [4]. It has reached a complex state, trying to maintain backwards compatibility and integrate advances in cryptanalysis and security. As such, it is hard to analyze and easily prone to implementation weaknesses.

Testing TLS implementations for (security) bugs is a tedious process. The TLS protocol flexibility to setup a secure connection with many different cryptographic algorithms, using different record sizes, key sizes, and other security parameters, contributes to creating an enormous state space. It does not suffice to test a specific portion of a TLS implementation under only one specific cipher suite. Given the critical role of such libraries in the overall system security, they must be checked thoroughly for all possible setups.

D. Combinatorial coverage of cipher suites

Combinatorial coverage is an effective measurement of test quality, complementing established methods, such as structural coverage and mutation testing [7]. Empirical data show that a significant number of software failures are induced by the interaction of two or more factors, and interaction faults can be extremely difficult to identify [8].

There is an inherent complexity of the TLS protocol specification, which is also reflected in its implementations. The selection of a specific cipher suite in the first phase of the protocol plays a key role in driving the software execution towards activating different parts of the implementation at the second phase. For example, the selection of the encryption algorithm and its key size will result in initialization and use of different data structures and code segments to be used at later stages.

We propose the use of combinatorial coverage metrics against the TLS cipher suites. This can be a first step guiding the test strategies for TLS implementations towards appropriate coverage of all cases and for reasoning about the achieved coverage of existing ones.

The availability of combinatorial coverage metrics per cipher suite recommendation allows us to comment on the complexity of each recommendation (and the resulting testing effort) and adapt the testing strategies towards verifying TLS implementations for compliance with specific (or all) recommendations.

In the next two Sections, we describe how we can model the cipher suites and we report the measurements we performed using the CCM tool provided by NIST [9].

III. Modeling of cipher suites

The first step for studying the combinatorial coverage of a discrete space is to devise a methodology that partitions the space into discrete parameters and their values. This modeling technique is refered to as input parameter modelling and the resulting model is the "Input Parameter Model" (IPM) [10]. This will enable us to map existing cipher suites into discrete parameters and values that can be analyzed and evaluated using coverage measurement tools.

The cipher suite comprises a combination of three cryptographic algorithms, namely for key exchange (e.g. the RSA algorithm), encryption (e.g., the AES algorithm), and hash and pseudorandom function (e.g., HMAC-SHA1). It should be noted that key exhange is not necessarily authenticated (e.g., anonymous Diffie-Hellman). The encryption algorithm

Key exchange	Encryption	Hash
NULL	NULL	NULL
RSA	RC4-40-NULL	MD5
RSA-EXPORT	RC4-128-NULL	SHA
DH-DSS-EXPORT	RC2-40-CBC	SHA256
DH-DSS	IDEA-128-CBC	SHA384
DH-RSA-EXPORT	DES-40-CBC	
DH-RSA	DES-56-CBC	
DHE-DSS-EXPORT	3DES-168-EDE_CBC	
DHE-DSS	AES-128-CBC	
DHE-RSA-EXPORT	AES-256-CBC	
DHE-RSA	CAMELLIA-128-CBC	
DH-anon-EXPORT	CAMELLIA-256-CBC	
DH-anon	SEED-128-CBC	
KRB5	AES-128-GCM	
KRB5-EXPORT	AES-256-GCM	
PSK	ARIA-128-CBC	
DHE-PSK	ARIA-256-CBC	
RSA-PSK	ARIA-128-GCM	
ECDH-ECDSA	ARIA-256-GCM	
ECDHE-ECDSA	CAMELLIA-128-GCM	
ECDH-RSA	CAMELLIA-256-GCM	
ECDHE-RSA	AES-128-CCM	
ECDH-anon	AES-256-CCM	
SRP-SHA	AES-128-CCM_8	
SRP-SHA-RSA	AES-256-CCM_8	
SRP-SHA-DSS		
ECDHE-PSK		
PSK-DHE		

TABLE I: IPM for cipher suites in IANA Registry

Key exchange	Encryption	Hash
ECDHE-ECDSA	AES-128-GCM	SHA256
ECDHE-RSA	AES-256-GCM	SHA384
DHE-DSS	CAMELLIA-128-GCM	
DHE-RSA	CAMELLIA-256-GCM	
	AES-128-CCM	
	AES-256-CCM	
	AES-128-CCM_8	
	AES-256-CCM_8	

TABLE II: IPM for cipher suites recommended by ENISA

is characterized by the cryptographic algorithm (e.g., AES), its key length (e.g., 128 bits), and its defined mode of operation (e.g., CBC or GCM).

The modeling is straightforward: we define three parameters, one for each type of cryptographic algorithms. We consider each different encryption tuple of (algorithm, key length, mode of operation) as one different value. The values of each parameter are derived by the specification or recommendation followed.

We analyzed the IANA TLS Cipher Suite Registry and the four recommendations of Mozilla, ENISA, BSI, and NSA Suite B. The resulting IPMs are summarized in Tables I,IV,II,III, and V respectively.

Key exchange	Encryption	Hash
ECDHE-ECDSA	AES-128-CBC	SHA256
ECDHE-RSA	AES-256-CBC	SHA384
DHE-DSS	AES-128-GCM	
DHE-RSA	AES-256-GCM	

TABLE III: IPM for cipher suites recommended by BSI

Key exchange	Encryption	Hash
ECDHE-ECDSA	AES-128-CBC	SHA
ECDHE-RSA	AES-256-CBC	SHA256
DHE-DSS	AES-128-GCM	SHA384
DHE-RSA	AES-256-GCM	

TABLE IV: IPM for cipher suites recommended by Mozilla

Key exchange	Encryption	Hash
ECDHE-ECDSA	AES-128-GCM	SHA256
	AES-256-GCM	SHA384

TABLE V: IPM for cipher suites recommended by NSA Suite B

It is evident from the IPMs that the IANA Registry contains a large selection of cryptographic algorithms for all the three uses. On the other hand, the NSA Suite B is the most strict one, as it allows only two cipher suites. Mozilla, ENISA, and BSI agree on the same set of algorithms for key exchange, using both algebraic and elliptic curve problems. Mozilla and BSI agree also on the encryption algorithms tuples, while ENISA accepts Camellia as well. ENISA and BSI agree on the (keyed) hash algorithms. Mozilla allows the SHA1 algorithm as well; however this will change in the near future[3].

IV. Combinatorial coverage of cipher suites

The Combinatorial Coverage Measurement (CCM) is a tool developed by the NIST [9]. CCM produces a comprehensive set of data on the combinatorial coverage of an existing set of tests. The output can be generated in spreadsheet format to allow easy processing and graphing.

We utilized this tool in a slightly novel way so as to measure the combinatorial coverage of the TLS cipher suite recommendations based on their respective IPMs we have developed. Since our model contains three parameters, the combinatorial coverage refers to two-way and three-way coverage. Figures 1,2,3,4, and 5 depict the visual output of the CCM tool for the five cipher suites discussed in this paper. The vertical axis measures the achieved combinatorial coverage while the horizontal axis measures the percentage (0-100%) of combinations reaching a particular coverage level. A vertical line at 100% would mark a full combinatorial coverage achieved by all the available combinations.

A. CCM of the recommendations

The IANA Registry has the most rich set of parameter values. It comes to no surprise that the 2-way and 3-way coverage is quite low. Figure 1 depicts a coverage of 37.62% for 2-way and 9.06% for 3-way. In particular, for 2-way coverage 363 out of 965 combinations are covered while for 3-way coverage the IANA registry covers 317 out of 3,500 combinations.

The BSI recommendation achieves a 90.6% 2-way coverage and a 50% 3-way coverage, as depicted in Figure 2.

[3]cf. https://wiki.mozilla.org/index.php?title=Security/Server_Side_TLS&oldid=1111459

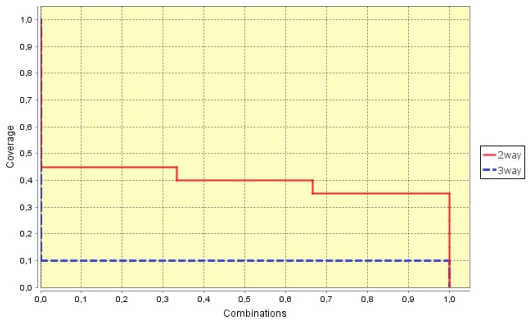

Fig. 1: CCM for IANA Registry

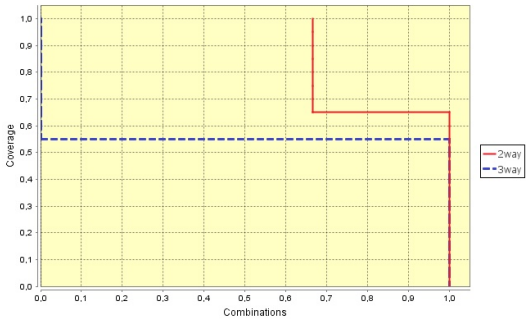

Fig. 2: CCM for BSI recommendation

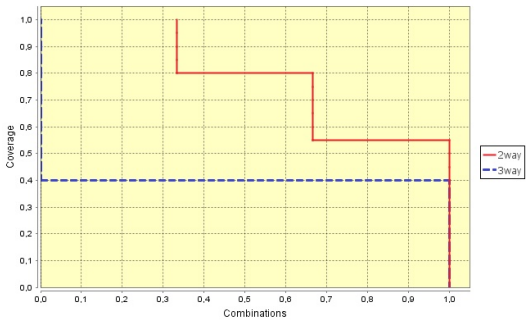

Fig. 3: CCM for ENISA recommendation

The ENISA recommendation achieves a 71.43% 2-way coverage and a 37.5% 3-way coverage, as depicted in Figure 3.

The Mozilla recommendation achieves a coverage of 87.5% and 45.83% for 2-way and 3-way respectively, as depicted in Figure 4.

The NSA Suite B recommendation achieves a coverage of 75% and 50% for 2-way and 3-way respectively, as seen in Figure 5.

It appears that in all cases, the recommended cipher suites do not achieve a full 2-way or 3-way combinatorial coverage. This means that there exist combinations (which can be generated from the respective IPM) that are not contained in the recommended cipher suites. It is a topic for future study if these missing combinations are deliberately excluded from the recommendation due to some security considerations or there is another reasoning for the exclusion.

The CCM tool is capable of generating automatically the

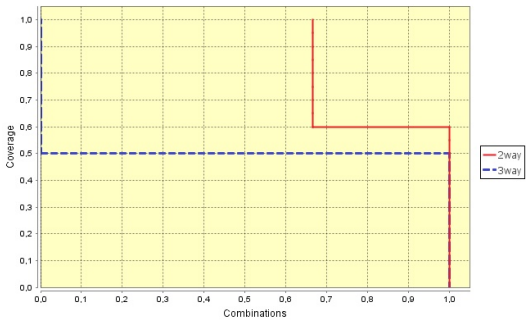

Fig. 4: CCM for Mozilla recommendation

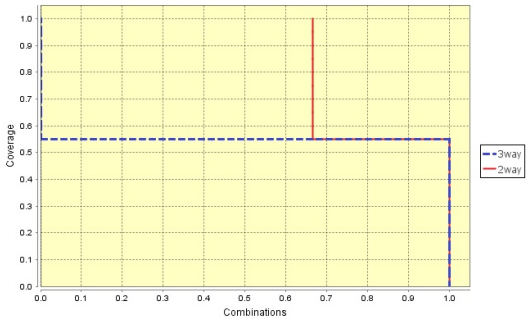

Fig. 5: CCM for NSA Suite B recommendation

missing combinations. We generated all the missing combinations for achieving full coverage for the case of Mozilla, BSI, and ENISA. Then, we compared against the IANA Registry. It appears that no missing combination is already registered there. This means that for achieving full coverage, the IANA Registry should be expanded to accommodate the missing cipher suites.

B. Implications for testing

The handshake phase of the TLS protocol allows two parties to agree on a common cipher suite to use for the forthcoming secure communication. In this sense, each different cipher suite defines a different system configuration or state that must be thoroughly tested for security. An exhaustive check of all possible cipher suites is an option. However, it would be beneficial to have a reduced set of cipher suites to test, while ensuring that all combinations are covered.

The analysis performed in the previous section suggests that none of the recommendations exhibits a full combinatorial coverage. Furthermore, the necessary combinations (i.e., cipher suites) to achieve a full 2-way and 3-way coverage are not defined in the IANA Registry. This limits the ability to automate the generation of minimal sets of cipher suites definitions (e.g., using the ACTS tool by NIST [11]): since these definitions are not mapped in the IANA registry, there is no 2-byte value describing them. Hence, there is no means through the handshake protocol to pass this definition to a software implementation of the TLS protocol. Thus, one must both extend the IANA registry *and* integrate the related

information to the TLS protocol implementation in order to be able to reduce the required testing effort.

V. Conclusions and future work

In this paper, we proposed the use of combinatorial coverage metrics against TLS cipher suite recommendations as these are specified by ENISA, BSI, Mozilla and NSA Suite B and developed the required models to perform an analysis of their coverage measurement using the CCM tool. Our findings indicate that none of the recommendations achieves full coverage. Even though we can not exclude the case that this is due to some security considerations, this is a topic that will be explored further in a future work. However, there are some implications for testing that should not go unnoticed. For example, additional actions would be required from a system designer to reduce the testing effort of a TLS implementation such as the expansion of the TLS cipher suite registry and the integration of the information back to the implementation itself.

Acknowledgment

The research presented in the paper has been funded in part by the Austrian Research Promotion Agency (FFG) under grant 851205 (Security ProtocoL Interaction Testing in Practice - SPLIT) and the Austrian COMET Program (FFG). Disclaimer: *Products may be identified in this document, but identification does not imply recommendation or endorsement by NIST, nor that the products identified are necessarily the best available for the purpose.*

References

[1] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Internet Engineering Task Force, Internet-Draft, 2014, work in Progress.

[2] E. Rescorla, "TLS Cipher Suite Registry," https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4, accessed: 2015-12-13.

[3] Mozilla, "Security/Server Side TLS," https://wiki.mozilla.org/Security/Server_Side_TLS, accessed: 2016-04-07.

[4] ENISA, "Study on cryptographic protocols," https://www.enisa.europa.eu/publications/study-on-cryptographic-protocols, accessed: 2016-04-07.

[5] BSI, "BSI TR-02102-2 Kryptographische Verfahren: Verwendung von Transport Layer Security (TLS)," https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/, accessed: 2016-04-07.

[6] NSA, "Fact sheet NSA suite B cryptography," http://www.nsa.gov/ia/programs/suiteb_cryptography/, November 2010, accessed: 2016-04-08.

[7] D. R. Kuhn, R. N. Kacker, and Y. Lei, "Combinatorial coverage as an aspect of test quality," *CrossTalk*, vol. 28, no. 2, pp. 19–23, 2015.

[8] D. Kuhn, R. Kacker, and Y. Lei, "Practical combinatorial testing," NIST Special Publication 800-142, 2010.

[9] I. Dominguez Mendoza, D. Kuhn, R. Kacker, and Y. Lei, "CCM: A tool for measuring combinatorial coverage of system state space," in *Empirical Software Engineering and Measurement, 2013 ACM / IEEE International Symposium on*, 2013, pp. 291–291.

[10] M. Grindal and J. Offutt, "Input parameter modeling for combination strategies," in *Proceedings of the 25th Conference on IASTED International Multi-Conference: Software Engineering*, ser. SE'07, Anaheim, CA, USA, 2007, pp. 255–260.

[11] L. Yu, Y. Lei, R. Kacker, and D. Kuhn, "ACTS: A combinatorial test generation tool," in *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, 2013, pp. 370–375.

Appendix

For the sake of completeness, we report the full range of cipher suites defined by each recommendation in Tables VI, VII, VIII and IX. These cipher suites were used for defining the IPM per recommendation.

Key exchange	Encryption	Hash
ECDHE-ECDSA	AES-128-GCM	SHA256
ECDHE-ECDSA	AES-256-GCM	SHA384

TABLE VI: NSA Suite B recommended cipher suites

Key exchange	Encryption	Hash
DHE-DSS	CAMELLIA-128-GCM	SHA256
DHE-DSS	AES-128-GCM	SHA256
DHE-DSS	CAMELLIA-256-GCM	SHA384
DHE-DSS	AES-256-GCM	SHA384
DHE-RSA	CAMELLIA-128-GCM	SHA256
ECDHE-RSA	CAMELLIA-128-GCM	SHA256
DHE-RSA	AES-128-GCM	SHA256
ECDHE-RSA	AES-128-GCM	SHA256
DHE-RSA	CAMELLIA-256-GCM	SHA384
ECDHE-RSA	CAMELLIA-256-GCM	SHA384
DHE-RSA	AES-256-GCM	SHA384
ECDHE-RSA	AES-256-GCM	SHA384
DHE-RSA	AES-128-CCM	SHA256
DHE-RSA	AES-128-CCM_8	SHA256
DHE-RSA	AES-256-CCM	SHA256
DHE-RSA	AES-256-CCM_8	SHA256
ECDHE-ECDSA	CAMELLIA-128-GCM	SHA256
ECDHE-ECDSA	AES-128-GCM	SHA256
ECDHE-ECDSA	CAMELLIA-256-GCM	SHA384
ECDHE-ECDSA	AES-256-GCM	SHA384
ECDHE-ECDSA	AES-128-CCM	SHA256
ECDHE-ECDSA	AES-128-CCM_8	SHA256
ECDHE-ECDSA	AES-256-CCM	SHA256
ECDHE-ECDSA	AES-256-CCM_8	SHA256

TABLE VII: ENISA recommended cipher suites

Key exchange	Encryption	Hash
ECDHE-ECDSA	AES-128-CBC	SHA256
ECDHE-ECDSA	AES-256-CBC	SHA384
ECDHE-ECDSA	AES-128-GCM	SHA256
ECDHE-ECDSA	AES-256-GCM	SHA384
ECDHE-RSA	AES-128-CBC	SHA256
ECDHE-RSA	AES-256-CBC	SHA384
ECDHE-RSA	AES-128-GCM	SHA256
ECDHE-RSA	AES-256-GCM	SHA384
DHE-DSS	AES-128-CBC	SHA256
DHE-DSS	AES-256-CBC	SHA256
DHE-DSS	AES-128-GCM	SHA256
DHE-DSS	AES-256-GCM	SHA384
DHE-RSA	AES-128-CBC	SHA256
DHE-RSA	AES-256-CBC	SHA256
DHE-RSA	AES-128-GCM	SHA256
DHE-RSA	AES-256-GCM	SHA384

TABLE VIII: BSI recommended cipher suites

Key exchange	Encryption	Hash
ECDHE-ECDSA	AES-128-CBC	SHA
ECDHE-ECDSA	AES-256-CBC	SHA
ECDHE-RSA	AES-128-CBC	SHA
ECDHE-RSA	AES-256-CBC	SHA
ECDHE-ECDSA	AES-128-CBC	SHA256
ECDHE-ECDSA	AES-256-CBC	SHA384
ECDHE-RSA	AES-128-CBC	SHA256
ECDHE-RSA	AES-256-CBC	SHA384
ECDHE-ECDSA	AES-128-GCM	SHA256
ECDHE-ECDSA	AES-256-GCM	SHA384
ECDHE-RSA	AES-128-GCM	SHA256
ECDHE-RSA	AES-256-GCM	SHA384
DHE-RSA	AES-128-CBC	SHA
DHE-DSS	AES-256-CBC	SHA
DHE-RSA	AES-256-CBC	SHA
DHE-DSS	AES-128-CBC	SHA256
DHE-RSA	AES-128-CBC	SHA256
DHE-RSA	AES-256-CBC	SHA256
DHE-RSA	AES-128-GCM	SHA256
DHE-RSA	AES-256-GCM	SHA384
DHE-DSS	AES-128-GCM	SHA256
DHE-DSS	AES-256-GCM	SHA384

TABLE IX: Mozilla recommended cipher suites