Speaking in Tongues
Practical Evaluation of TLS Cipher Suites Compatibility

Manuel Koschuch[1], Taro Fruhwirth[2], Alexander Glaser[2], Silvie Schmidt[2] and Matthias Hudler[1]
[1]*Competence Centre for IT-Security, FH Campus Wien, University of Applied Sciences*
Favoritenstrasse 226, 1100 Vienna, Austria
[2]*FH Campus Wien, University of Applied Sciences, Favoritenstrasse 226, 1100 Vienna, Austria*

Keywords: OpenSSL, O-Saft, Bettercrypto, Openssl-compare, Applied Crypto Hardening, Cipher Suite, Cipher String.

Abstract: The Transport Layer Security (TLS) protocol is still the de-facto standard for secure network connections over an insecure medium like the internet. But its flexibility concerning the algorithms used for securing a channel between two parties can also be a weakness, due to the possible agreement on insecure ciphers. In this work we examine an existing white paper (Applied Crypto Hardening) giving recommendations on how to securely configure SSL/TLS connections with regard to the practical feasibility of these recommendations. In addition we propose an additional configuration set with the aim of increasing compatibility as well as security. We also developed a small Cipher Negotiation Crawler (CiNeg) to test TLS-handshakes using given cipher configurations with a supplied list of websites and show its practical usability.

1 INTRODUCTION

Since its initial public specification in 1995, the Transport Layer Security (TLS) protocol (Dierks and Rescorla, 2008), originally and until v3.0 known as Secure Sockets Layer (SSL) (Freier et al., 2011), has become the de-facto standard for secure network communications over an insecure channel. One of the main reasons for its widespread usage (from web-browsers to mobile apps to embedded systems) is the flexibility this protocol offers with regard to the cryptographic algorithms used in a session.

To achieve this flexibility, the concept of *cipher suites*, described by *cipher strings*, is employed. Section 2 gives an overview of these strings and how they are used in the SSL/TLS handshake process. But this mechanism also creates practical problems: due to compatibility issues, or simple misconfiguration, a potentially very large number of SSL/TLS secured systems using insecure configurations exist. In addition, the cipher suites that — given a specific cipher string configuration — are actually negotiated with a specific server often remain unclear.

Several guides exist to support administrators on how to choose secure and compatible cipher strings, usually mostly focused on a single (or small range of) product(s). One guide that tries to take a broader approach to this topic is the *Applied Crypto Hardening (ACH)* white paper (Breyha et al., 2015), which is currently (2/2015) still in draft status and in near constant flux. In Section 2.4 we give a more detailed description of the recommendations presented in this guide.

Our main motivation for this work was now to determine how well the cipher strings recommended in (Breyha et al., 2015) are usable in practice, how they scale with different SSL/TLS versions, and what cipher suites are effectively negotiated when using the given strings in a practical setting. We also developed a small Cipher Negotiation Crawler (CiNeg) to perform TLS-handshakes with a given list of websites and show its practical usability. Section 3 details our results.

Finally, we propose another cipher string, trying to find a balance between security and compatibility, and evaluate its practical applicability in Section 4, summing up our results and findings in Section 5.

2 TRANSPORT LAYER SECURITY PROTOCOL

Transport Layer Security (TLS) is, as well as its predecessor SSL, a hybrid cryptographic protocol (SSL specifications were last updated in (Freier et al.,

Koschuch M., Fruhwirth T., Glaser A., Schmidt S. and Hudler M..
Speaking in Tongues - Practical Evaluation of TLS Cipher Suites Compatibility.
DOI: 10.5220/0005507900130023
In *Proceedings of the 6th International Conference on Data Communication Networking* (DCNET-2015), pages 13-23
ISBN: 978-989-758-112-0

2011), TLS is specified in (Dierks and Allen, 1999) and was most recently updated in (Popov, 2015)). It employs asymmetric cryptography during the initial handshake phase to verify the authenticity of usually at least one (the server side) of the communicating parties, as well as to exchange a symmetric key between those parties. The data communication itself is then encrypted and integrity protected using symmetric techniques.

The initial handshake can be divided into four phases, as detailed in Figure 1, where the individual phases perform the following functions:

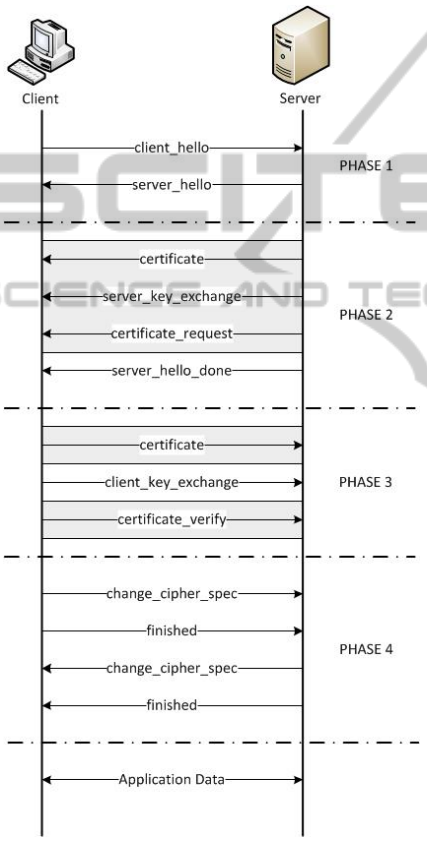

Figure 1: TLS Basic Handshake (cf. (Stallings, 2008)). The shaded messages are optional or situation dependent.

during *phase 1* security capabilities are established. This includes information about the protocol version, session ID, cipher suite, compression method, as well as the initial random number.

Phase 2 contains three optional messages by the server, i.e. certificate, key exchange, and the request for a client certificate. Usually, in the context of a web-server communicating with a client via HTPP over SSL (HTTPS), at least the server certificate is sent to authenticate the server.

In *phase 3* the client may send its certificate if requested by the server. In any case the client sends key exchange information, depending on the actual cipher suite agreed upon in phase 1. At the end of phase 3 the client may send a certificate verification (forcing it to employ the private key corresponding to the one present in the client's certificate).

During *phase 4* the *change_cipher_spec* messages are sent both ways, and the handshake is finished by exchanging a message symmetrically encrypted with the agreed upon key and cipher.

The most interesting part of this handshake for our work is phase 1, in particular the negotiation of the cipher suites. Usually the client offers a list of supported suites in its *client_hello* message, with the server selecting a suite from its own configuration shared with the client and communicates it back to the client in the *server_hello* message. The cipher suites supported by server and client depend on their respective configuration, as well as on the SSL/TLS version used.

2.1 Cipher Suites

A cipher suite defines the cryptographic algorithms used during an SSL/TLS connection, and usually contains information about the primitives for key exchange, authentication, symmetric encryption, and integrity protection.

An example suite, given in the format used by OpenSSL, would be DHE-RSA-AES128-SHA: ephemeral Diffie-Hellman (Diffie and Hellman, 2006) is used for agreeing on a symmetric key, RSA (Rivest et al., 1978) signatures for checking the authenticity of the parties involved, AES (NIST, 2001) with a 128-bit key for symmetric bulk data encryption, and SHA-1 (NIST, 2012) for calculation of the HMAC integrity protection value.

Supported cipher suites vary by SSL/TLS version and specific implementations, and are, among others, specified in (Dierks and Allen, 1999; Medvinsky and Hur, 1999; Chown, 2002; Moriai et al., 2005; Lee et al., 2005; Eronen and Tschofenig, 2005; Dierks and Rescorla, 2006).

The most commonly used algorithms for the different cryptographic primitives are:

Key Exchange: RSA, or Diffie-Hellman (DH) or its equivalent over elliptic curves (ECDH), both in ephemeral and non-ephemeral varieties ((EC)DH(E))

Authentication: RSA, or the Digital Signature Algorithm (DSA) or its equivalent over elliptic curves (ECDSA)

Encryption: AES, RC4, or CAMELLIA

Hash: MD5, or a member of the SHA family (-1,-2,-256,-384)

Most of the software products employing SSL/TLS allow for some kind of control over which cipher suites are available for negotiation, thereby enabling the users and/or administrators for enforcing certain minimum security requirements or enabling compatibility with a wider range of devices. The desired cipher suites can either be explicitly enumerated or, as in the case of OpenSSL, given as logical compositions of classes of algorithms (e.g. "*no SHA-1 AND no RC4 AND AES*", see https://www.openssl.org/docs/apps/ciphers.html for a detailed discussion of the expressions allowed).

Regardless of the approach taken, in practice it is often unclear or at least quite cumbersome to determine which actual cipher suite is being negotiated when connecting to a particular server. In this work we try to introduce a structured tool-assisted approach to answer this question, as well as use this approach to evaluate the practical applicability and compatibility of an existing project (Breyha et al., 2015) giving cipher string recommendations.

2.2 OpenSSL

The OpenSSL[1] project offers an open source toolkit written in C to implement SSL/TLS protocols, together with comprehensive utilities for creating and verifying digital certificates. OpenSSL is also the default cryptographic library used in the Apache[2] and nginx[3] HTTP servers. Both servers combined comprise about 67,26% of current domains served (according to Analyzer.cc[4]).

Analyzer.cc also gives the most used version (as of 1/2015) as 0.9.8e (25.3%), followed by version 1.0.0 and 0.9.8g, with 14.3% and 13.2%, respectively. Table 1 shows the top five OpenSSL versions according to Analyzer.cc.

Table 1: Top 5 OpenSSL versions in use (according to http://technology.analyzer.cc/application/openssl).

OpenSSL Version	Installations	%
0.9.8e	119,839	25.31%
1.0.0	67,937	14.35%
0.9.8g	62,510	13.20%
1.0.1e	56,580	11.95%
0.9.8o	36,191	7.64%
other	130,343	27.53%

[1]https://www.openssl.org/

[2]http://httpd.apache.org/

[3]http://nginx.org/

[4]http://technology.analyzer.cc/application/openssl

2.3 SSL/TLS Version Distribution

To get recent statistics on SSL/TLS usage and its versions' dissemination we used SSL Pulse[5]. As of December 2014, 1.5 million public websites are SSL/TLS enabled; due to efficiency concerns SSL Pulse recently decided to use the top 200,000 sites (according to Alexa's list[6]) for its monitoring and the resulting statistics. SSL Pulse offers various statistics concerning SSL/TLS trends. From that we find that - as of Dec.7th, 2014 - almost a quarter (22.6%) of the websites scanned uses weak or insecure cipher suites, i.e. these sites support symmetric ciphers with a key length lower than 128 bits. This percentage of websites is 1.1% less than a month earlier, i.e. there is a positive trend concerning cipher strength. Furthermore, SSL Pulse examines the usage of SSL/TLS versions, i.e. SSL v2.0, SSL v3.0, TLS v1.0, TLS v1.1, and TLS v1.2.

Table 2 shows the usage of SSL/TLS versions in December 2014 compared to November 2014. It is obvious that SSL support is declining and the percentage of sites deploying TLS is growing, yet there is still a considerable number of servers available that are able to fallback to SSL when a client requests to do so, making it all the more important to choose cipher strings in a way to avoid this from happening.

2.4 Available Tools

As already mentioned before, several projects exist trying to give administrators guidelines on how to securely configure SSL/TLS installations by deploying specifically crafted cipher strings. In this work our interest is twofold: first we want to examine the recommendations given in the Applied Crypto Hardening (ACH) project (Breyha et al., 2015) on their practical usability and compatibility. In addition, we try to define our own new cipher string, trying to strike a balance between the two configurations given in (Breyha et al., 2015). We chose the ACH project mainly for two reasons: the decision finding process is open and well documented on a public mailing list (http://lists.cert.at/cgi-bin/mailman/listinfo/ach) as well as in the corresponding Git repository (https://git.bettercrypto.org/ach-master.git).

And the contributors to the document come from academic research as well as from the industry, giving a broad perspective on the topic. These two properties lend credibility to the notion that a configuration created and recommended this way is actually sensible

[5]https://www.trustworthyinternet.org/ssl-pulse/

[6]http://www.alexa.com/

Table 2: SSL/TLS Trends: absolute and relative numbers of sites supporting the respective version. (Data from https://www.trustworthyinternet.org/ssl-pulse/, 1/2015).

Protocol	Nov.2014 abs.	Nov.2014 %	Dec.2014 abs.	Dec.2014 %	Trend
SSL v2.0	23,238	16.6%	25,096	15.5%	- 1.1%
SSL v3.0	91,526	60.6%	80,085	53.5%	- 7.1%
TLS v1.0	149,132	99.5%	150,293	99.6%	+ 0.1%
TLS v1.1	68,595	45.4%	70,933	47.4%	+ 2.0%
TLS v1.2	72,701	48.1%	75,022	50.1%	+ 2.0%

in practice. Two cipher strings are provided by ACH (as of draft revision 1333f7a):

Cipher String A is a configuration for strong security, i.e. using strong ciphers, but offering less compatibility, i.e. fewer clients (Breyha et al., 2015): EDH+ aRSA +AES256: EECDH +aRSA +AES256: !SSLv3.

Cipher String B is configured with weaker ciphers, but it offers better compatibility (Breyha et al., 2015): EDH+ CAMELLIA: EDH +aRSA: EECDH +aRSA +AESGCM: EECDH +aRSA +SHA256: EECDH: +CAMELLIA128: +AES128: +SSLv3: !aNULL: !eNULL:!LOW:!3DES: !MD5: !EXP: !PSK: !DSS: !RC4: !SEED: !IDEA: !ECDSA: kEDH: CAMELLIA128 -SHA: AES128 -SHA.

More detailed explanation for selecting these ciphers and their compatibilities can be found in (Breyha et al., 2015).

To evaluate these two strings in practice we used two additional tools: Openssl-compare[7], allowing us to test a given cipher string against a certain OpenSSL version and determining the resulting supporting cipher suites.

And O-Saft[8] to test a given cipher string against a server.

3 PRACTICAL EVALUATION

In this section we describe the practical tests performed and all the results and conclusions we derive from these. We start with estimating the practical compatibility of cipher string A and B from (Breyha et al., 2015), and conclude by examining the available cipher suites in different OpenSSL versions.

3.1 Methodology

We first try to estimate the percentage of websites a client with a given cipher string configuration can successfully connect to, taking into account the distribution of SSL/TLS versions as well as the cipher suites described by the cipher string. To do this we use openly available tools as detailed in Section 2.4 as well as our own command line application CiNeg, detailed in the next Subsection.

We then try to verify our estimations by using CiNeg to connect to Alexa's top 1,000/top 25,000 sites and record the negotiated cipher suites.

Finally, we specify a new cipher string and verify its usability and compatibility by performing the scans again.

3.2 Compatibility Estimation for ACH's Cipher Strings

In addition to the tools listed in Section 2.4, we developed the tool CiNeg, which creates an SSL/TLS session for a given list of websites with a chosen cipher string. CiNeg only tries to connect once, which results in small differences of the responding web sites caused by the timeout after 2.5 seconds. CiNeg only uses the given URL and does not try connecting to any sub-domains. The tool uses the installed OpenSSL client, version 1.0.1f in our case. The output shows the negotiated cipher suite and the protocol version used.

We estimate the compatibility of the cipher strings A and B with web servers which support SSL or TLS, using the results from scans with our CiNeg tool given in Table 3.

3.2.1 Compatibility Estimation for Cipher String A

Cipher string A only supports TLS 1.2 cipher suites. This massively reduces the number of compatible sites. According to SSL-Pulse data from December 2014, only 50.1% of SSL/TLS-enabled sites support TLS 1.2. SSL-Pulse uses about 200,000 SSL/TLS websites.

A second scan by Securitypitfalls[9] from November 2014, which uses 441,636 SSL/TLS websites from Alexa's top list[10], indicates that 66.2% support TLS 1.2. The scan by Securitypitfalls shows

[7]https://github.com/azet/openssl-compare

[8]https://www.owasp.org/index.php/O-Saft

[9]https://securitypitfalls.wordpress.com/2014/12/

[10]http://www.alexa.com/topsites

that 56.7% of the surveyed sites support ECDHE and 49.5% support DHE.

Due to the numbers of the previous results and the compatibility values from Table 3 we estimate that the compatibility for this cipher string is between 50% and 60%.

We used CiNeg to verify these values and ran a scan with the top 1,000 and top 25,000 websites according to Alexa's top list. There are about 600 (60%) websites in the top 1,000 that support SSL/TLS and about 14,500 (58%) in the top 25,000.

We first performed a scan with the cipher string `'ALL'` to see how many websites return an error within the handshake, even when the cipher suites are not limited. This value may vary slightly according to the number of no responses on port 443, but this should not severely skew our estimations.

We subtract the number of the errors within the handshakes with all possible cipher suites from the number of errors in the handshake with the limited cipher suites, i.e. we removed cipher suites with limitations such that the errors caused by these cipher suites do not occur any more. The result for the shared cipher suite percentage of cipher string A, when using the top 1,000 websites according to Alexa, is 54.3%. The result for the top 25,000 websites is 56.1%.

These results confirm our initial estimation.

This quite limited compatibility is in line with the goal of cipher string A, to focus on security and not on compatibility with older systems.

The results of the CiNeg scan also include the negotiated cipher suites. Figure 2 shows the distribution of negotiated cipher suites for the top 25,000 websites. This chart shows that ECDHE-RSA-AES256-GCM-SHA384 is used in about 65% of the successful negotiations, although the DHE cipher suites are preferred by the client. Possible reasons for this might be caused by the stronger support of the ECDHE cipher suite or the fact, that - according to the scan by Securitypitfalls - 67% of the websites use their own priority order regarding their ciphers. It is also notable that about 95% of the negotiated cipher suites use AES-GCM.

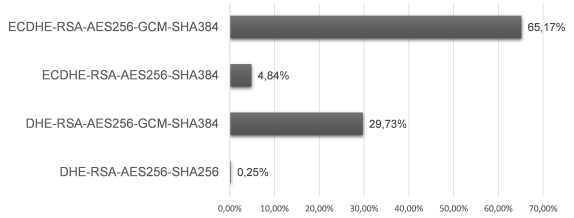

Figure 2: Percentage of cipher suites negotiated using cipher string A with Alexa's top 25,000 websites.

3.2.2 Compatibility Estimation for Cipher String B

Cipher string B offers better compatibility than cipher string A. The main reason for this fact is the support of cipher suites which use RSA for key exchange. This cipher string shares cipher suites with SSLv3, TLSv1, TLSv1.1 and TLS1.2. The results of SSL-Pulse show that nearly all of the websites support at least one of these protocol versions and only 1.3% of the websites support RC4 exclusively.

According to a scan by Securitypitfalls from November 2014, only 0.02% of the SSL/TLS websites from Alexa's top 1 million sites only support SSLv2. This scan used 441,636 SSL/TLS websites. The results of this scan also show that RSA for key exchange is supported by 94.2% of the websites, followed by ECDHE with 56.7% and DHE with 49.5%. AES for encryption is - according to this scan - supported by 93.6%. In this scan there are no different results for AES128 and AES256. A different work (Huang et al., 2014) using older data also found similar results concerning the distribution of AES128/256, so this seems also valid for the newer values.

According to the previous numbers and the compatibility values from Table 3 we estimate that this cipher string is compatible with about 95% of the SSL/TLS websites.

We used CiNeg to verify this estimation in the same way as described in the previous section. The result for the shared cipher suite percentage of cipher string B, when using the top 1,000 websites according to Alexa's list, is 98.3%. The result for the top 25,000 websites is 97.2%, i.e. the compatibility is slightly better regarding the tested websites than estimated.

Figure 3 shows the distribution of negotiated cipher suites for the top 25,000 websites when using cipher string B. This chart shows that about 26.5% of the websites negotiated the cipher suite AES128-SHA, which does not offer perfect forward secrecy (since using RSA as the key exchange method); this cipher suite is the main reason for the better compatibility compared to cipher string A.

3.3 OpenSSL Support for Various Cipher Suites

Openssl-compare, available at https://github.com/azet/openssl-compare, provides the opportunity to install, execute and compare different OpenSSL versions. It offers the functionality to test a cipher string against all OpenSSL versions present on the current host. The command `ciphersuite` returns all the shared cipher suites. To determine which cipher suite

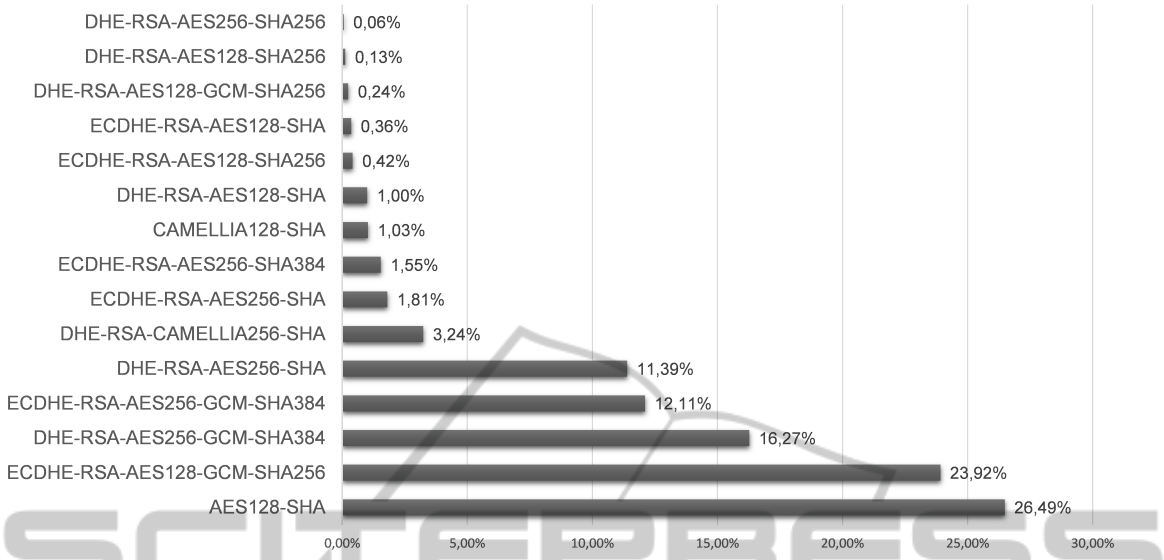

Figure 3: Percentage of cipher suites negotiated using cipher string B with Alexa's top 25,000 websites.

of the shared cipher suites is actually negotiated during a handshake, the command `negotiate` is used.

This tool was used to elaborate which cipher suite is supported by which OpenSSL version. We also used openssl-compare to test which cipher suite is used for the different OpenSSL versions when a handshake is executed with the cipher strings A and B of the ACH project. The shared cipher suites where also identified within these tests.

The following OpenSSL versions and subversions where used:

- 0.9.6 (e,i-m)
- 0.9.7 (a-m)
- 0.9.8 (a-y,za,zb,zc)
- 1.0.0 (a,b,beta1-beta5,c-o)
- 1.0.1 (a,b,beta1-beta3,c-j)
- 1.0.2 (beta1-beta3)

3.3.1 Supported Cipher Suites

To determine the oldest OpenSSL version supporting a specific cipher suite we used the command `openssl-compare ciphersuite -s 'ALL'`. The cipher string `'ALL'` matches all the cipher suites that are supported by default.

Table 3 shows an excerpt of supported cipher suites. The compatibility values are the results of scans using our CiNeg tool with the top 1,000 websites according to Alexa's list. The compatibility percentage is computed as described above.

Table 3 shows that AES is supported since version 0.9.7. It is also notable that ECDH and ECDHE are supported since version 1.0.0. In version 1.0.1 major modifications were made by introducing AES-GCM (128 and 256), SHA256, and SHA384. AES256-GCM is always used in a cipher suite with SHA384. In TLS v1.2 AES256 is usually used in a cipher suite with SHA256, except for cipher suites that include ECDH(E), where SHA384 is used.

3.3.2 OpenSSL Support for Cipher String A

The command `openssl-compare ciphersuite -s` was used to find the OpenSSL versions which share cipher suites with the cipher string A. The result shows that this cipher string shares cipher suites only with OpenSSL versions 1.0.1 and 1.0.2. These shared cipher suites are:

- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-SHA384

The three beta versions of version 1.0.1 do not support the cipher suite DHE-RSA-AES256-SHA256. This cipher string only supports TLSv1.2. To examine which cipher suites are negotiated in a handshake the command `openssl-compare negotiate -s` was used. The negotiated cipher suite for all supported OpenSSL versions is DHE-RSA-AES256-GCM-SHA384.

Table 3: OpenSSL versions' cipher suite support and estimated compatibility for a given suite according to the top 1.000 Alexa websites. "x" denotes support for a cipher suite, "-" lack thereof.

OpenSSL Identifier	0.9.6	0.9.7	0.9.8	1.0.0	1.0.1	1.0.2	Compatibility [%]
RC4-SHA	x	x	x	x	x	x	65.6
DES-CBC3-SHA	x	x	x	x	x	x	87.4
AES128-SHA	-	x	x	x	x	x	96.1
AES256-SHA	-	x	x	x	x	x	94.0
EDH-RSA-DES-CBC3-SHA	x	x	x	x	x	x	27.3
DHE-RSA-AES128-SHA	-	x	x	x	x	x	34.2
DHE-RSA-AES256-SHA	-	x	x	x	x	x	27.5
DHE-RSA-CAMELLIA128-SHA	-	-	-	x	x	x	17.1
DHE-RSA-CAMELLIA256-SHA	-	-	-	x	x	x	17.0
DHE-RSA-AES128-SHA256	-	-	-	-	x	x	16.9
DHE-RSA-AES256-SHA256	-	-	-	-	x	x	16.9
DHE-RSA-AES128-GCM-SHA256	-	-	-	-	x	x	19.0
DHE-RSA-AES256-GCM-SHA384	-	-	-	-	x	x	17.4
ECDHE-RSA-AES128-SHA	-	-	-	x	x	x	60.7
ECDHE-RSA-AES256-SHA	-	-	-	x	x	x	56.2
ECDHE-RSA-AES128-SHA256	-	-	-	-	x	x	51.2
ECDHE-RSA-AES256-SHA384	-	-	-	-	x	x	50.5
ECDHE-RSA-AES128-GCM-SHA256	-	-	-	-	x	x	52.1
ECDHE-RSA-AES256-GCM-SHA384	-	-	-	-	x	x	51.1

3.3.3 OpenSSL Support for Cipher String B

There is no shared cipher suite for any of the 0.9.6 OpenSSL versions. This cipher string causes problems with some of the older versions. In OpenSSL versions 0.9.7 to 0.9.7l the cipher suite AES256-SHA is incorrectly included. This is caused by parsing problems[11] and is fixed in 0.9.7m. 0.9.7m includes the following cipher suites when tested with cipher string B:

- AES128-SHA
- DHE-RSA-AES128-SHA
- DHE-RSA-AES256-SHA

The parsing problems also exist in versions 0.9.8 to 0.9.8d and the shared cipher suites are:

- AES128-SHA
- **AES256-SHA**
- DHE-RSA-AES128-SHA
- DHE-RSA-AES256-SHA
- **ECDHE-RSA-AES128-SHA**
- **ECDHE-RSA-AES256-SHA** (not in 0.9.8(a))
- **ECDH-RSA-AES128-SHA**
- **ECDH-RSA-AES256-SHA**

[11] https://www.openssl.org/news/changelog.html

The five shared cipher suites, indicated in bold in the listing above, should not be included in the shared cipher suites in versions 0.9.8 to 0.9.8d. The ECDHE cipher suites are according to the cipher string, but they are not included by the default lists and should not be used in version 0.9.8.

The parsing problems are fixed in version 0.9.8e. The versions 0.9.8e to 0.9.8zc share the following cipher suites:

- AES128-SHA
- DHE-RSA-AES128-SHA
- DHE-RSA-AES256-SHA

In versions 1.0.0, 1.0.1 and 1.0.2 there are no problems regarding parsing or negotiation of the cipher suite. The 1.0.0 versions share the following cipher suites:

- AES128-SHA
- CAMELLIA128-SHA
- DHE-RSA-AES128-SHA
- DHE-RSA-AES256-SHA
- DHE-RSA-CAMELLIA128-SHA
- DHE-RSA-CAMELLIA256-SHA
- ECDHE-RSA-AES128-SHA
- ECDHE-RSA-AES256-SHA

This list shows that in version 1.0.0 CAMELLIA is also used in DHE cipher suites. The versions 1.0.1

and 1.0.2 also share the cipher suites of the 1.0.0 versions. Additional shared cipher suites in 1.0.1 and 1.0.2 are:

- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-SHA256
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-SHA384

This list shows that AES-GCM, SHA256, and SHA384 since are supported OpenSSL 1.0.1 (used by only about 12% of installations, according to Table 1).

We then again utilized the openssl-compare tool to find the actual cipher suite negotiated when using cipher string B; Table 4 lists our results.

Table 4: Negotiated cipher suite when using cipher string B.

Cipher Suite	OpenSSL
ECDHE-RSA-AES128-SHA	0.9.8(a)
ECDHE-RSA-AES256-SHA	0.9.8b-d
DHE-RSA-AES256-SHA	0.9.8e-zc
DHE-RSA-CAMELLIA256-SHA	1.0.0
DHE-RSA-AES256-GCM-SHA384	1.0.1, 1.0.2

This test shows that the priority settings of the cipher suites does not work properly for the versions 0.9.8 to 0.9.8d, because ECDHE is preferred, but cipher suites including DHE should be preferred instead and ECDHE should not be used with versions 0.9.8.

The following cipher string fixes most of cipher string B's problems:

```
EDH +CAMELLIA: kEDH +aRSA +AES: EECDH
+aRSA +AESGCM: EECDH +aRSA +SHA256:
EECDH: +CAMELLIA128:  +AES128:  +SSLv3:
!aNULL: !eNULL: !LOW: !3DES: !MD5:  !EXP:
!PSK: !DSS: !RC4:  !SEED: !IDEA: !ECDSA:
CAMELLIA128 -SHA: AES128 -SHA.
```

The remaining problem with this cipher string is that the cipher suite AES256-SHA is included in the shared cipher suites for versions 0.9.7h-k and 0.9.8a. This is caused by parsing problems of these versions. The negotiated cipher suites using the modified cipher string B are listed in Table 5. These results are according to the priorities of the cipher suites in the cipher string, i.e. it is working as intended.

If this cipher string should include all the cipher suites mentioned for string B in (Breyha et al., 2015), *CAMELLIA256-SHA:AES256-SHA* has to be added at the end of the cipher string.

Table 5: Negotiated cipher suite when using modified cipher string B.

Cipher Suite	OpenSSL
DHE-RSA-AES256-SHA	0.9.7 - 0.9.8
DHE-RSA-CAMELLIA256-SHA	1.0.0
DHE-RSA-AES256-GCM-SHA384	1.0.1, 1.0.2

3.4 Analyzing Available Cipher Suites Using O-Saft

In order to get all supported cipher suites of web servers we used the tool O-Saft from https://www.owasp.org/index.php/O-Saft. To automate the O-Saft scans we wrote a tool which executes the O-Saft command `cipherraw` for a given list of URLs and a tool which parses the results.

Figure 4 shows the results for the top 100 sites according to Alexa. 58 websites of the 100 support SSL/TLS. The cipher suite with the highest compatibility (96.6%) is AES128-SHA. It is also notable that 67.2% of the sites support RC4-SHA and 44.8% support RC4-MD5. Some of these values differ slightly to results of other scans; this is caused by the small sample size.

4 CIPHER SUITE PROPOSAL

After having evaluated the practical compatibility of cipher strings A and B from (Breyha et al., 2015), our goal was to create a cipher string that offers high security and is compatible with a wide range of websites. Since the two cipher strings by (Breyha et al., 2015) are designed for different requirements — one of them for high security and one for best compatibility — the created cipher string should be placed in the middle between these two.

The first condition for the cipher string is that it only uses cipher suites that support forward secrecy. These are cipher suites using DHE or ECDHE for key exchange. This reduces the compatibility, but is a very important point. (Perfect) Forward Secrecy (Huang et al., 2014) uses the private key of the web server to sign a Diffie-Hellman (DH) key exchange message. When using DH, the server always uses the same key pair: i.e. when the private key gets stolen, all sessions recorded in the past could be decrypted. When using Diffie-Hellman ephemeral (DHE) a new key pair is created for each session and the private key of the session is never stored on the server after the session has terminated. Gaining access to the private keys of a server now only enables decryption of the sessions currently in progress, but not of the ones recorded in the past.

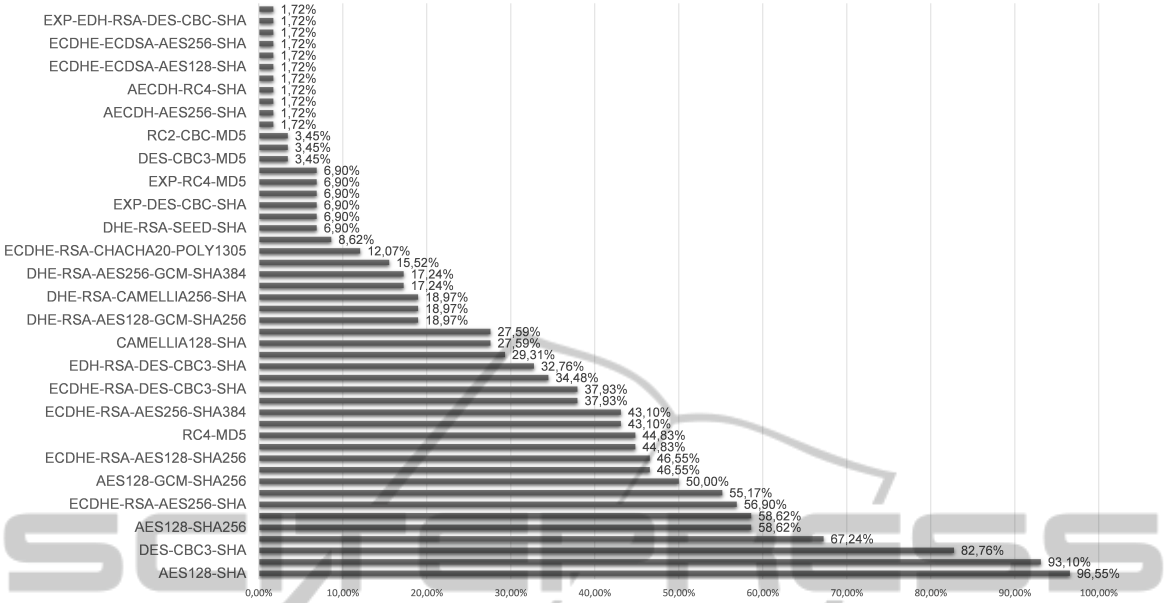

Figure 4: Results of the O-Saft scans, indicating how many percent of Alexa's top 100 websites use a specific cipher suite.

In our cipher string proposal, TLS 1.2 cipher suites are preferred. If the cipher suites have the same protocol version, then ECDHE is preferred, because according to (Huang et al., 2014) only 0.3% of the websites support DH parameters with a size of 2048 bits. 99.3% of the websites support a parameter size of 1024 bits and even 34% support a parameter size of 512 bits.

The chosen encryption methods are AES and CAMELLIA with key lengths of 128 and 256 bits, respectively. These ciphers offer high security and are widely supported; as MAC we only allow hash functions from the SHA family.

The resulting cipher string is:

```
EECDH: kEDH +aRSA +AES: kEDH +aRSA
+CAMELLIA: +SSLv3:  !aNULL: !eNULL:
!3DES: !IDEA: !RC4:  !MD5:  !EXP: !PSK:
!DSS: !ECDSA: !DES
```

The cipher suites shared by OpenSSL from version 1.0.1 and up with this string are:

- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-RSA-AES128-SHA
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- DHE-RSA-AES256-SHA
- DHE-RSA-CAMELLIA128-SHA
- DHE-RSA-CAMELLIA256-SHA
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-SHA256
- ECDHE-RSA-AES128-SHA
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA

Table 6 gives an overview of the cipher suites negotiated by the different OpenSSL versions when using this newly proposed string. When compared to cipher strings A and B from above, our string prefers AES over CAMELLIA and elliptic curve versions of Diffie-Hellman over the integer ones.

Table 6: Negotiated cipher suite when using our proposed cipher string.

Cipher Suite	OpenSSL
DHE-RSA-AES256-SHA	0.9.7 - 0.9.8
ECDHE-RSA-AES256-SHA	1.0.0
ECDHE-RSA-AES256-GCM-SHA384	1.0.1, 1.0.2

CiNeg was used to get an approximate value for the compatibility. The result for the shared cipher suite percentage of our cipher string proposal, when using the top 1,000 websites according to Alexa, is 74.8%. The result for the top 25,000 websites is 77.9%. These results are showing that — in line with the goal of the cipher string proposal — the compatibility is ranked between ACH's cipher string A and B (see also Table 7).

Figure 5 shows the distribution of negotiated cipher suites for the top 25,000 websites when using

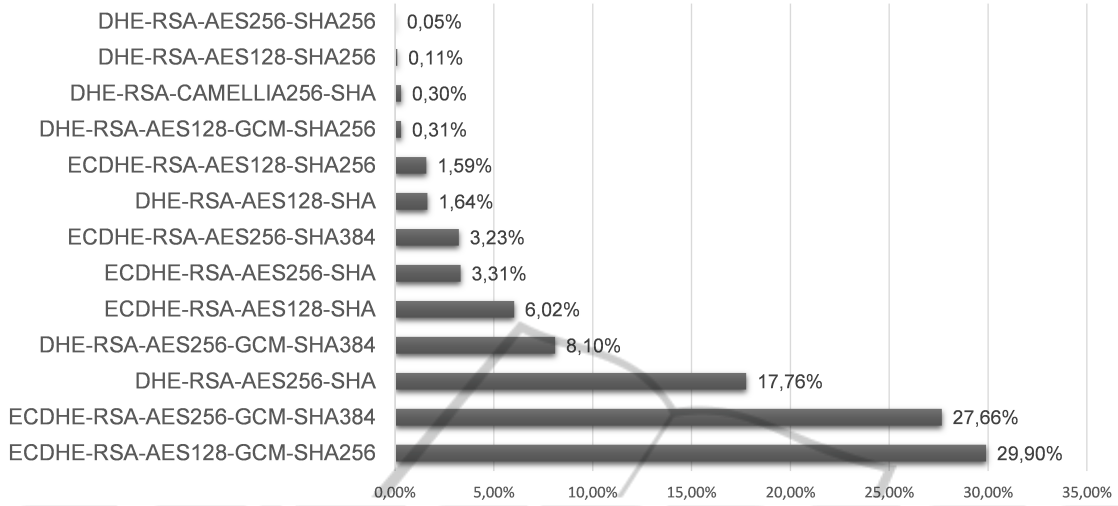

Figure 5: Percentage of cipher suites negotiated using our cipher string proposal with Alexa's top 25,000 websites.

the cipher string proposal. 71% of the negotiated cipher suites are TLS 1.2 cipher suites and this is according to the priorities in the cipher string. It is also notable that 66% of the negotiated cipher suites use AES-GCM.

5 SUMMARY AND CONCLUDING REMARKS

5.1 Comparing the Cipher Strings

Table 7 indicates the percentage of shared cipher suites for the cipher string A, B and our proposal. These are the results from the CiNeg scans with Alexa's top 1,000 and top 25,000 websites, respectively.

Table 7: Cipher Strings A - B - Proposal, comparing the percentage of Alexa's top 1,000/top 25,000 websites sharing at least one cipher suite with the given cipher string (i.e. a TLS handshake with a site would succeed for the respective cipher string)

	Top 1,000	Top 25,000
Cipher String A	54.30%	56.10%
Cipher String B	98.32%	97.17%
Proposed Cipher String	74.75%	77.9%

Cipher string A only supports TLS 1.2 cipher suites which offer perfect forward secrecy, uses hash-functions of the SHA-2 family and AES256. The goal of this cipher string is to use strong cipher suites. This leads to a lower compatibility.

Cipher string B does support some weaker cipher suites in order to offer better compatibility. The two fall back cipher suites are AES128-SHA and CAMELLIA128-SHA; the other cipher suites offer perfect forward secrecy. TLS 1.2 cipher suites are preferred.

Our cipher string proposal supports only cipher suites which offer perfect forward secrecy, but it is not limited to TLS 1.2 cipher suites, i.e. ciphers suites using AES128 and CAMELLIA128 are also supported. This offers a wider range of support than cipher string A, with only a minor reduction in security.

5.2 Conclusion

Bruce Schneier's words "Security is a process, not a product[12]" clearly sums up that there will never be a final solution regarding security issues.

Everything concerning security underlies fast development and rapidly changing requirements; therefore any security issues depend on individual conditions. In this work we evaluated the practical usability and compatibility of two OpenSSL cipher strings publicly proposed by members of academia and industry, using existing tools like O-Saft and Openssl-compare, as well as our newly developed CiNeg tool.

In addition we propose a new cipher string which provides better compatibility than ACH's cipher string A and stronger security than cipher string B, with the caveat of being less secure than A and lacking compatibility compared to B.

This again underlines the problem that there is no single perfect cipher string which meets all possible

[12]https://www.schneier.com/crypto-gram/archives/2000/0515.html

requirements; administrators have to find the right cipher string by balancing security strength and compatibility regarding their individual needs, using the tools available at their disposal.

Considering this complexity as well as recently discovered attack vectors against TLS like FREAK (Beurdouche et al., 2015) and Logjam (Adrian et al., 2015), a point of further research should be to determine if the algorithm variability present in the TLS protocol might in fact be a severe weakness and different approaches on selecting cryptographic primitives could be considered.

ACKNOWLEDGEMENTS

Manuel Koschuch is being supported by the MA23 - Wirtschaft, Arbeit und Statistik - in the course of the funding programme “Stiftungsprofessuren und Kompetenzteams für die Wiener Fachhochschul-Ausbildungen”.

REFERENCES

Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J. A., Heninger, N., Springall, D., Thom, E., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Bguelink, S., and Zimmermann, P. (2015). Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. Technical report, INRIA Paris-Rocquencourt and INRIA Nancy-Grand Est, CNRS and Universit de Lorraine and Microsoft Research and University of Pennsylvania and Johns Hopkins and University of Michigan.

Beurdouche, B., Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P.-Y., and Zinzindohoue, J. K. (2015). A Messy State of the Union: Taming the Composite State Machines of TLS. In *IEEE Security & Privacy 2015, preprint*.

Breyha, W., Durvaux, D., Dussa, T., Kaplan, L. A., Mendel, F., Mock, C., Koschuch, M., Kriegisch, A., Pschl, U., Sabet, R., San, B., Schlatterbeck, R., Schreck, T., Wrstlein, A., Zauner, A., and Zawodsky, P. (2015). Applied Crypto Hardening. Technical report.

Chown, P. (2002). RFC3268 - Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS). Technical report, Network Working Group.

Dierks, T. and Allen, C. (1999). RFC2246 - The TLS Protocol Version 1.0. Technical report, Network Working Group.

Dierks, T. and Rescorla, E. (2006). RFC4346 - The Transport Layer Security (TLS) Protocol Version 1.1. Technical report, Network Working Group.

Dierks, T. and Rescorla, E. (2008). RFC5246 - The Transport Layer Security (TLS) Protocol Version 1.2. Technical report, Network Working Group.

Diffie, W. and Hellman, M. (2006). New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654.

Eronen, P. and Tschofenig, H. (2005). RFC4279 - Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). Technical report, Network Working Group.

Freier, A., Karlton, P., and P.Kocher (2011). RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0. Technical report, Internet Engineering Task Force (IETF).

Huang, L., Adhikarla, S., Boneh, D., and Jackson, C. (2014). An experimental study of TLS forward secrecy deployments. In *IEEE CS Security and Privacy Workshops*.

Lee, H., Yoon, J., and Lee, J. (2005). RFC4162 - Addition of SEED Cipher Suites to Transport Layer Security (TLS). Technical report, Network Working Group.

Medvinsky, A. and Hur, M. (1999). RFC2712 - Addition of Kerberos Cipher Suites to Transport Layer Security (TLS). Technical report, Network Working Group.

Moriai, S., Kato, A., and Kanda, M. (2005). RFC4132 - Addition of Camellia Cipher Suites to Transport Layer Security (TLS). Technical report, Network Working Group.

NIST (2001). Advanced Encryption Standard (AES) (FIPS PUB 197).

NIST (2012). Secure Hash Standard (SHS) (FIPS PUB 180-4).

Popov, A. (2015). RFC7465 - Prohibiting RC4 Cipher Suites. Technical report, Internet Engineering Task Force (IETF).

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

Stallings, W. (2008). *Cryptography and Network Security*, page 539. Pearson, 4th edition.